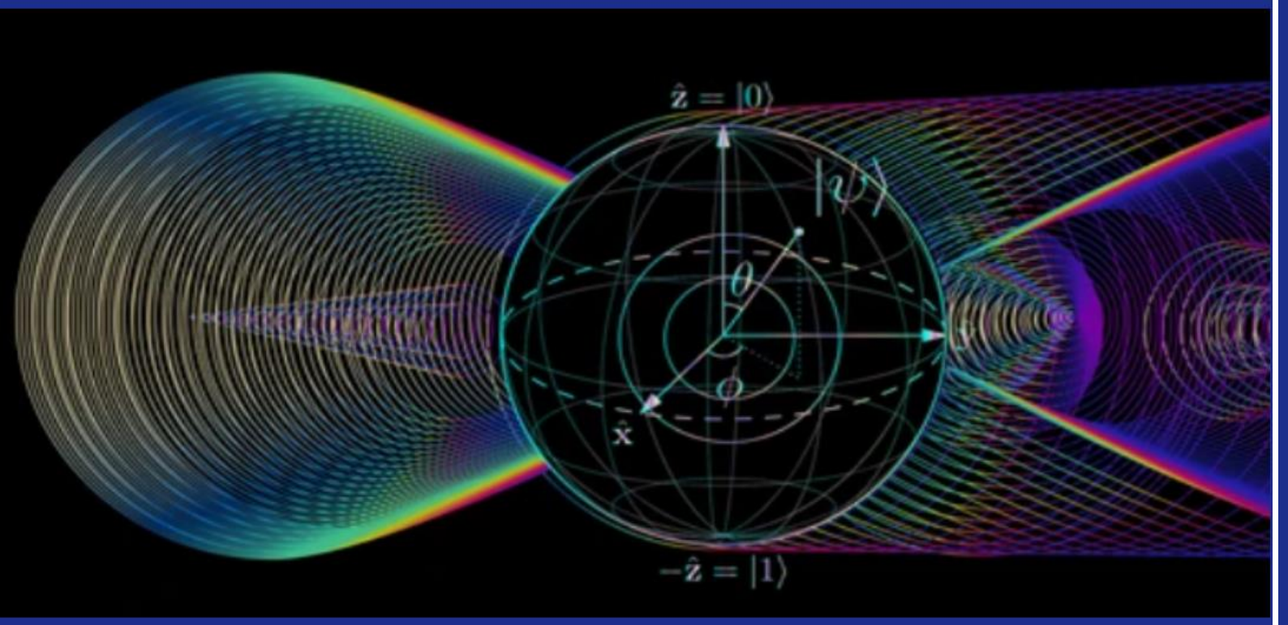


Мухамедиева Дилноз Тулкуновна

КВАНТОВЫЕ МЕТОДЫ АНАЛИЗА И ОБРАБОТКИ ИЗОБРАЖЕНИЙ

Монография

Мухамедиева Дилноз Тулкуновна

КВАНТОВЫЕ МЕТОДЫ АНАЛИЗА И ОБРАБОТКИ ИЗОБРАЖЕНИЙ

Монография

Национальный исследовательский университет «Ташкентский институт инженеров ирригации и механизации сельского хозяйства»

Мухамедиева Дилноз Тулкуновна

КВАНТОВЫЕ МЕТОДЫ АНАЛИЗА И ОБРАБОТКИ ИЗОБРАЖЕНИЙ

Ташкент-2024

Издательство «Fan ziyosi»

УДК 519.71(575.1)

Д.Т.Мухамедиева. «Квантовые методы анализа и обработки изображений». Монография – Т.: Изд. «Fan ziyosi», 2024. 307 с.

В работе рассмотрены актуальные теоретико-методические проблемы разработки методов и алгоритмов анализа и обработки изображений. Квантовый алгоритм, реализованный в работе, позволяет произвести преобразование классического изображения в квантовое состояние, выделения границ и преобразование полутонового изображения в бинарное, показывает возможности квантовой теории информации в интерпретации классических задач. Изложение ведется в достаточно строгой и в то же время доступной форме. Все основные положения и операции иллюстрируются большим числом примеров. Книга рассчитана на широкой круг читателей, включающий специалистов по прикладной математике, инженеров, а также лиц, интересующихся вопросами оптимизации, теории систем и общими вопросами построения гибридных моделей принятия решений.

Рекомендовано к печати НТС

Национально-исследовательского университета «Ташкентский институт инженеров ирригации и механизации сельского хозяйства»

Рецензенты:

Маматов Н.С. - д.т.н., профессор Национально- исследовательского университета «Ташкентский институт инженеров ирригации и механизации сельского хозяйства».

Рахимов Н.О. – д.т.н., профессор Ташкентского университета информационных технологий.

С О Д Е Р Ж А Н И Е

Введение

В современной науке и технике постоянно возникает необходимость в решении таких стратегически важных задач, как предсказание погоды и расчет климатических изменений, создание онкологических препаратов, обработка сигналов из Вселенной для поиска внеземных цивилизаций, обработка символьной информации, криптоанализ [1], опережающий расчет траекторий движущихся воздушных и космических объектов и другие задачи. Практическая реализация перечисленных задач на современных, даже суперкомпьютерных, системах требует недопустимо большого промежутка времени или вообще невозможна. В последнее время наблюдается стремительный рост интереса к квантовым компьютерам [2]. Современные методики по распознаванию объектов имеют ряд существенных не достатков: погрешности поиска в базах данных [3] большой размерности и определения объекта при смене его положения, ухудшение качеств по определению объектов в зависимости от качества освещения [4], средства маскировки [5]. Предполагается использование алгоритмов [6] квантовой природы при определении объектов и образов. В последнее время наблюдается стремительный рост интереса к квантовым компьютерам, особенно после продажи действующих квантовых вычислителей. Использование квантовых компьютеров, позволяет существенно увеличить скорость решения вычислительных задач и, самое главное, экспоненциально увеличить скорость решения NP полных проблем, которые на классических машинах могут решаться за неприемлемое время.

Данная работа посвящена решению задачи исследования и разработки методов функционирования квантовых алгоритмов и моделей квантовых вычислительных устройств. Квантовый алгоритм, реализованный в работе, позволяет произвести преобразование классического изображения в квантовое состояние, выделения границ и преобразование полутонового изображения в бинарное, показывает возможности квантовой теории информации в интерпретации классических задач. Целью работы является компьютерное моделирование квантового алгоритма для решения задачи преобразования классического изображения с использованием квантовых вычислительных средств и методов, изучение существующих алгоритмов распознавания образов и создание

эффективной модели распознавания с помощью свойств и методов квантовых вычислений. Данная работа посвящена решению задачи исследования и разработки методов функционирования квантовых алгоритмов и моделей квантовых вычислительных устройств. Актуальность данных исследований заключается в математическом и программном моделировании и реализации квантового алгоритма для решения классов задач классического характера. Научная новизна данного направления в первую очередь выражается в постоянном обновлении и дополнении поля квантовых исследований по ряду направлений, а компьютерная симуляция квантовых физических явлений и особенностей слабо освещена в мире. В настоящее время во многих передовых странах мира интенсивно ведутся научно-исследовательские работы по разработке и созданию квантовых компьютеров и их программного обеспечения, наблюдается стремительный рост интереса к квантовым компьютерам. Публикуется большое количество статей и монографий. В работе приведены основные теоретические и практические результаты в области квантового компьютинга.

Обработка изображений популярна в нашей повседневной жизни из-за необходимости извлекать важную информацию из нашего трехмерного мира, включая множество приложений в широко разделенных областях, таких как биомедицина, экономика, развлечения и промышленность. Природа визуальной информации, сложность алгоритмов и представление 3D-сцен в 2D-пространствах — все это популярные темы исследований. В частности, быстро растущий объем данных изображений, а также все более сложные вычислительные задачи стали важными движущими силами для дальнейшего повышения эффективности обработки и анализа изображений. С тех пор как концепция квантовых вычислений была предложена Фейнманом в 1982 году, многие достижения показали, что квантовые вычисления значительно повысили эффективность вычислений [1]. Квантовая обработка информации использует квантово-механические свойства, такие как квантовая суперпозиция, запутанность и параллелизм, и эффективно ускоряет многие классические задачи, такие как факторизация больших чисел, поиск в несортированной базе данных, выборка бозонов, квантовое моделирование, решение линейных систем уравнений и машинное обучение. Эти уникальные квантовые свойства также можно

использовать для ускорения обработки сигналов и данных. При квантовой обработке изображений ключевую роль играет представление квантового изображения, которое существенно определяет виды задач обработки и качество их выполнения

Термин «мягкие вычисления» введен Лофти Заде в 1994 году. Это понятие объединяет такие области, как нечеткая логика, нейронные сети, вероятностные рассуждения, сети доверия и эволюционные алгоритмы, которые дополняют друг друга и используются в различных комбинациях или самостоятельно для создания гибридных интеллектуальных систем.

Оптимизационные задачи заключаются в нахождении минимума (максимума) заданной функции. Такую функцию называют целевой. Как правило, целевая функция — сложная функция, зависящая от некоторых входных параметров. В оптимизационной задаче требуется найти значения входных параметров, при которых целевая функция достигает минимального (максимального) значения. Существует целый класс оптимизационных методов. Условно все оптимизационные методы можно разделить на методы, использующие понятие производной (градиентные методы) и стохастические методы (например, методы группы Монте-Карло). С их помощью можно найти экстремальное значение целевой функции, но не всегда можно быть уверенным, что получено значение глобального экстремума. Нахождение локального экстремума вместо глобального называется преждевременной сходимостью. Помимо проблемы преждевременной сходимости существует другая проблема — время процесса вычислений. Зачастую более точные оптимизационные методы работают очень долго.

Для решения поставленных проблем и проводится поиск новых эволюционных алгоритмов. Предложенные сравнительно недавно — в 1975 году — Джоном Холландом генетические алгоритмы (ГА) основаны на принципах естественного отбора Ч. Дарвина. ГА относятся к стохастическим методам. Эти алгоритмы успешно применяются в различных областях деятельности (экономика, физика, технические науки и т.п.). Созданы различные модификации ГА и разработан ряд тестовых функций. Рассмотреть как работают ГА, и какие проблемы остаются неразрешенными — цель данной работы.

Эволюционные алгоритмы относят к области мягких вычислений. Первая схема генетического алгоритма была предложена в 1975 году в Мичиганском университете Джоном Холландом (John Holland) [8], а предпосылками этому послужили работы Ч. Дарвина [2] (теория эволюции) и исследования Л.Дж.Фогеля, А.Дж. Оуэнса, М.Дж.Волша [23] по эволюции простых автоматов, предсказывающих символы в цифровых последовательностях (1966). Новый алгоритм получил название «репродуктивный план Холланда» и в дальнейшем активно использовался в качестве базового алгоритма в эволюционных вычислениях. Идеи Холланда развили его ученики Кеннет Де Йонг (Kenneth De Jong) из университета Джорджа Мейсона (Вирджиния) [22] и Дэвид Голдберг (David E. Goldberg) из лаборатории ГА Иллинойса [6]. Благодаря им, был создан классический ГА, описаны все операторы и исследовано поведение группы тестовых функций (именно алгоритм Голдберга и получил название «генетический алгоритм»).

Генетические алгоритмы — это адаптивные методы поиска, которые в последнее время используются для решения задач оптимизации. В них используются как аналог механизма генетического наследования, так и аналог естественного отбора. При этом сохраняется биологическая терминология в упрощенном виде и основные понятия линейной алгебры.

Целью исследования является решения многокритериальных оптимизационных задач на основе использования мягких вычислений.

Глава 1. КВАНТОВЫЕ ВЫЧИСЛЕНИЯ И ОБРАБОТКА ИЗОБРАЖЕНИЯ

1.1.Понятие квантового бита

Единицей хранения информации является бит. Ячейка памяти классического компьютера объемом в 1 бит может находиться в одном из двух различных состояниях. Эти состояния принято обозначать 0 и 1, саму такую ячейку также принято называть битом. Если бит находится в состоянии 0, то говорят, что он хранит значение 0, если же он находится в состоянии 1, то говорят, что он хранит значение 1 [1-4].

При выполнении вычислений над битами можно совершать различные двоичные операции, например такие:x

– отрицание (NOT): $y = \overline{x}$.

x	**y**
0	**1**
1	**0**

– конъюнкция («И», AND): $y = x_1 \wedge x_2$.

x_1	**x_2**	**y**
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

– дизъюнкция («ИЛИ», OR): $y = x_1 \vee x_2$.

x_1	**x_2**	**y**
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

Данные базовые операции являются основой любых дискретных вычислений. Любая более сложная логическая операция – это определенная комбинация базовых операций. С использованием набора базовых побитовых операций работает обычный компьютер.

Квантовый компьютер – это средство вычислительной техники, в основе которого лежат законы квантовой механики [4–7]. В квантовых компьютерах для вычисления применяются так называемые квантовые алгоритмы, которые используют эффекты квантовой механики, например, квантовый параллелизм и квантовая запутанность [7– 11].

Квантовый компьютер оперирует так называемыми квантовыми битами. Можно определить квантовый бит, или сокращенно ***q-бит (кубит)***, как квантово-механическую систему, имеющую два состояния, обозначаемых, соответственно, как $|0\rangle$ и $|1\rangle$. Однако в отличие от классического случая в квантовой механике эти два состояния могут находиться в ***состоянии суперпозиции***, т.е. наиболее общее состояние квантового бита может быть записано как:

$$\langle\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

где α и β – комплексные коэффициенты. Другими словами, можно сказать, что законы квантовой механики допускают другие значения ***кубита***, которые называются состояниями суперпозиции. Таким образом, состояние суперпозиции представляет собой значения между экстремумами 0 и 1, а квантовый бит может принимать бесконечно много значений.

Например, проведем аналогию с выключателем света. Классический бит может принимать только одно из двух состояний – «включено» или «выключено» (рис. 1.1, а, б). Кубиты похожи на светильник с возможностью регулировки яркости (рис. 1.1, в) [6].

Рис. 1.1. Различие бита и кубита

Кубит можно определить как вектор единичной длины в двумерном гильбертовом пространстве над полем комплексных чисел [1–4]. Состояния $|0\rangle$ и $|1\rangle$ вместе представляют собой базисные вектора. Как и все векторы, они указывают направление и имеют величину. Для записи двух состояний кубитов можно использовать обозначения ***бра*** ($\langle\ |$) и ***кет*** ($|\ \rangle$) – обозначения Дирака. Векторы вида $|\ \rangle$ называются ***кет***-векторами, а вида $\langle\ |$ ***бра***-векторами. Обозначения кет соответствует следующим векторам: Кет-вектора, соответствующие нулевому и единичному состоянию кубита будут иметь следующий вид:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ и } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

При измерении состояния системы с волновой функцией $\langle\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ вероятность обнаружить ее в состоянии $|0\rangle$ равна α^2 , а вероятность обнаружить ее в состоянии $|1\rangle$ равна β^2. Сумма этих вероятностей равна единице:

$$|\alpha|^2 + |\beta|^2 = 1.$$

Данное соотношение называется ***условием нормализации***. То есть формула для волновой функции $|\psi\rangle$ описывает, в какой пропорции бесконечное множество всех вариантов значений квантового состояния $|\psi\rangle$ содержит варианты базисных состояний $|0\rangle$ и $|1\rangle$. Визуализация состояния кубита возможна с помощью специального инструмента, называемого сферой Блоха.

Сфера Блоха – это сфера с единичным радиусом, при этом точка на ее поверхности соответствует состоянию кубита.

Когда кубит находится в суперпозиции $|0\rangle$ и $|1\rangle$, вектор будет располагаться между двумя этими точками на сфере (угол θ). Вращение вокруг оси Z описывается углом φ, и отвечает за изменение фазы кубита. Сфера Блоха показана на рисунке 1.2.

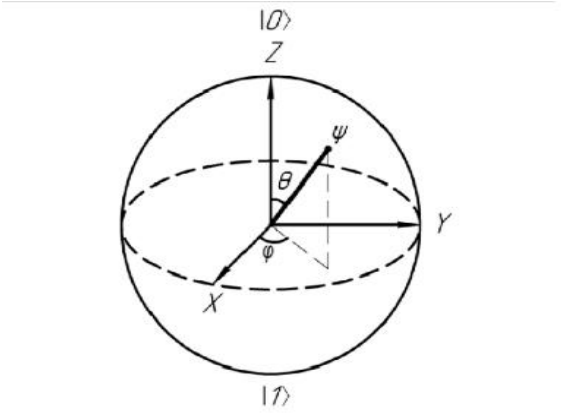


Рис. 1.2. Сфера Блоха. Состояние в верхней части сферы представляет $|0\rangle$, а состояние в нижней части сферы представляет $|1\rangle$

Физические реализации подобной системы с двумя состояниями могут быть разнообразными:

– электрон или ядро со спином 1/2, ориентированным по или против направления магнитного поля;

– атом с двумя различными энергетическими состояниями;

– фотон с горизонтальной или вертикальной поляризацией;

– и т.д.

Понятие квантовой системы

Для квантовых вычислений, как правило, требуется больше одного кубита. Система, состоящая из нескольких кубитов, представляет собой тензорное произведение составляющих ее систем. Такая система называется ***квантовой системой*** [11-14]. Состояние квантовой системы, которая состоит из n кубитов можно представить следующим выражением:

$$|q_1\rangle ... |q_n\rangle = (\alpha_0|0\rangle + \beta|1\rangle) \otimes (\alpha_1|0\rangle + \beta|1\rangle) \otimes ... \otimes (\alpha_{n-1}|0\rangle + \beta_{n-1}|1\rangle)$$

Приведем пример для системы двух кубитов. В данном случае мы имеем четырехмерный вектор единичной длины. Полностью

смешанное состояние системы двух кубитов можно описать следующим образом:

$$|\psi\rangle = (\alpha_0|0\rangle + \beta|1\rangle) \otimes (\alpha_1|0\rangle + \beta|1\rangle) = \alpha_0\alpha_1|00\rangle + \alpha_0\beta_1|01\rangle + \beta_0\alpha_1|10\rangle + \beta_0\beta_1|11\rangle).$$

При этом сумма вероятностей нахождения в том или ином состоянии по-прежнему равна 1.

$$|\alpha_0\alpha_1|^2 + |\alpha_0\beta_1|^2 + |\beta_0\alpha_1|^2 + |\beta_0\beta_1|^2 = 1.$$

Как и в бинарном случае, этот набор возможных результатов называется измерительным базисом, а приводящие к ним комбинации значений – базисными состояниями. Тогда любое системное квантовое состояние мы можем записать как суперпозицию базисных состояний:

$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle.$$

Физический смысл коэффициентов, стоящих в формуле перед базисными состояниями, тот же, что и для одного кубита – это пропорция вариантов значений. Однако, в отличие от одного кубита, это значения не одиночного измерения, а комбинация двух измерений.

Таким образом, квантовая система из двух кубитов может находиться в одном из четырех состояний:

– $|00\rangle$ – оба кубита при измерении дают результат $|0\rangle$.

– $|01\rangle$ – первый кубит при измерении дает $|0\rangle$, второй кубит дает $|1\rangle$.

– $|10\rangle$ – первый кубит при измерении дает $|1\rangle$, второй кубит дает $|0\rangle$.

– $|11\rangle$ – оба кубита при измерении дают результат $|1\rangle$.

Другими словами, вектор состояния n -кубитной системы существует в 2^n -мерном комплексном пространстве и представляет собой сумму 2^n базисных векторов – базисных состояний. Вероятностная природа квантовой механики проявляется в процессе измерений. Измерение является единственным способом извлечения данных, определяющих квантовое состояние. В результате измерения кубит немедленно коллапсирует. Допустим, одномерный кубит находится в состоянии суперпозиции. Если его измерить, то он

примет одно конкретное значение – $|0\rangle$ или $|1\rangle$. После измерения кубита коэффициенты α и β, которыми характеризовалось его предыдущее состояние, будут утеряны. Рассмотрим n-кубит:

$$|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle .$$

Так как длина вектора остается равной единицы, то можно записать

$$\sum_{k=0}^{2^n-1} |\alpha_k|^2 = 1.$$

Когда мы будем выполнять измерение состояния такого кубита, то получим одно из классических значений совокупности n битов от 000...0 до 111...1, где значение k (в двоичном представлении) появится с вероятностью $|\alpha_k|^2$. Следует отметить, что для каких-то состояний вероятность может оказаться нулевой!

Измерение квантовой системы

Чтобы получить информацию о состоянии квантовой системы необходимо выполнить его измерение. Для проведения измерений мы должны активно воздействовать на квантовую систему. В процессе измерения квантовой системы в выбранном базисе мы получим один из векторов этого базиса. В результате сразу же после измерения состояние квантовой системы разрушается, то есть система переходит в состояние, соответствующее наблюденному значению [14, 15].

Приведем пример возможных исходов измерения для одно-кубитной системы: – Если система находилась в состоянии $|0\rangle$, то при ее измерении мы получим тоже $|0\rangle$.

– Если система находилась в состоянии $|1\rangle$, то при ее измерении мы получим тоже $|1\rangle$.

– Если система находилась в состоянии суперпозиции $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, то в результате ее измерения мы получим вектор $|0\rangle$ с вероятностью $|\alpha|^2$, а вектор $|1\rangle$, с вероятностью $|\beta|^2$.

Следует отметить, что вероятность получения в процессе измерения конкретного вектора выбранного базиса ($|0\rangle$ или $|1\rangle$) равна квадрату модуля скалярного произведения вектора данной системы

на этот вектор [1-4]. При этом в результате измерений невозможно определить значения α и β, а также невозможно измерить повторно ту же самую систему, потому что после измерения состояние квантовой системы разрушается. Таким образом, для получения максимально достоверной информации о состоянии системы при ее измерении необходимо выбирать базис, один из векторов которого наиболее близок с измеряемым кубитом [4–7].

В общем случае вероятностный процесс измерения квантовой системы происходит следующим образом:

– Пусть у нас имеется квантовое состояние системы из n кубитов ψ и стандартный базис пространства состояний системы $k = \left\{ |0\rangle, ..., |2^n - 1\rangle \right\}$

$$|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle .$$

– При измерении состояния системы по отношению к базису k с вероятностью $|\alpha_i|^2$ результат измерения будет – $|i\rangle$.

– После измерения квантовая система будет находиться в состоянии – $|i\rangle$.

– После измерений амплитуды состояний отличных от $|i\rangle$ будут равны нулю: $\alpha_k = 0$ для $k \neq i$. Рассмотрим пример измерения 2-кубита.

Пример 1.1: Выполним измерение следующего 2-кубита

$$|\psi\rangle = \alpha_0 |00\rangle + \alpha_1 |01\rangle + \alpha_2 |10\rangle + \alpha_3 |11\rangle ,$$

$$\alpha_0 = 0.3;\ \ \alpha_1 = 0.1;\ \alpha_2 = 0.9;\ \alpha_3 = 0.3;$$

При измерении данного 2-кубита возможны четыре варианта:

– С вероятностью 0.09 получится значение 00 и 2-кубит перейдет в состояние $|00\rangle$.

– С вероятностью 0.01 получится значение 01 и 2-кубит перейдет в состояние $|01\rangle$.

– С вероятностью 0.81 получится значение 10 и 2-кубит перейдет в состояние $|10\rangle$.

– С вероятностью 0.09 получится значение 11 и 2-кубит перейдет в состояние $|11\rangle$.

В некоторых задачах требуется измерить состояние лишь некоторых кубитов в большом n-кубите. Приведем пример подобной ситуации для 3-кубита.

Пример 1.2: Выполним измерение первых двух битов кубита, но не будем трогать третий бит.

$$|\psi\rangle = \alpha_0|000\rangle + \alpha_1|001\rangle + \alpha_2|010\rangle + \alpha_3|011\rangle + \alpha_5|101\rangle + \alpha_6|110\rangle,$$

$$\alpha_0 = 0.3;\ \alpha_1 = -0.6;\ \alpha_2 = -0.1;\ \alpha_3 = -0.7;\ \alpha_5 = 0.1;\ \alpha_6 = -0.2;$$

Видно, что в заданном нами примере отсутствуют состояния $|100\rangle$ и $|111\rangle$, т.е. их коэффициенты α_4 и α_7 равны 0. Так как нам необходимо измерить только два первых кубита, то возможны четыре варианта исхода: 00, 01, 10, 11. После измерения первые два кубита получат определенные значения, а значение третьего кубита не будет фиксировано.

– Вероятность наблюдения значения 00 будет определяться как $\alpha_0^2 + \alpha_1^2 = 0.09 + 0.36 = 0.45$, при этом новое состояние системы будет следующим:

$$|\psi\rangle = \frac{1}{\sqrt{\alpha_0^2 + \alpha_1^2}}(\alpha_0|000\rangle + \alpha_1|001\rangle),$$

$$\alpha_0 = 0.3;\ \alpha_1 = -0.6;$$

– Вероятность наблюдения значения 01 будет определяться как $\alpha_2^2 + \alpha_3^2 = 0.01 + 0.49 = 0.5$, при этом новое состояние системы будет следующим:

$$|\psi\rangle = \frac{1}{\sqrt{\alpha_2^2 + \alpha_3^2}}(\alpha_2|010\rangle + \alpha_3|011\rangle),$$

$$\alpha_2 = -0.1;\ \alpha_3 = -0.7;$$

Стоит отметить, что для нормализации состояния третьего ненаблюдаемого кубита необходимо коэффициенты этих двух состояний разделить на корень квадратный из вероятности появления данного исхода.

– Вероятность наблюдения значения 10 будет определяться как $\alpha_5^2 = 0.01$, при этом новое состояние системы будет $|101\rangle$.

– Вероятность наблюдения значения 11 будет определяться как $\alpha_6^2 = 0.04$, при этом новое состояние системы будет $|110\rangle$. Заметьте,

учитывая особенности нашего 3-кубита (отсутствие состояний $|101\rangle$ и $|110\rangle$), в последних двух случаях вероятность того, что третий бит соответственно принимает значения 1 и 0, равна 1.

Пример 1.3: Выполним измерение последних двух кубитов, но не будем трогать первый кубит.

$$|\psi\rangle = \alpha_0|000\rangle + \alpha_1|001\rangle + \alpha_2|010\rangle + \alpha_3|011\rangle + \alpha_5|101\rangle + \alpha_6|110\rangle,$$

$$\alpha_0 = 0.3;\ \alpha_1 = -0.6;\ \alpha_2 = -0.1;\ \alpha_3 = -0.7;\ \alpha_5 = 0.1;\ \alpha_6 = -0.2;$$

– Вероятность наблюдения значения 00 будет определяться как $|\alpha_0|^2 = 0.09$, при этом новое состояние системы будет $|000\rangle$.

– Вероятность наблюдения значения 01 будет определяться как $\alpha_1^2 + \alpha_5^2 = 0.36 + 0.01 = 0.37$, при этом новое состояние системы будет следующим:

$$|\psi\rangle = \frac{1}{\sqrt{\alpha_1^2 + \alpha_5^2}}(\alpha_1|001\rangle + \alpha_5|101\rangle),$$

$$\alpha_5 = 0.1;\ \alpha_1 = -0.6;$$

– Вероятность наблюдения значения 10 будет определяться как $\alpha_2^2 + \alpha_6^2 = 0.01 + 0.04 = 0.05$, при этом новое состояние системы будет следующим:

$$|\psi\rangle = \frac{1}{\sqrt{\alpha_2^2 + \alpha_6^2}}(\alpha_2|010\rangle + \alpha_6|110\rangle),$$

$$\alpha_2 = 0.1;\ \alpha_6 = -0.2;$$

– Вероятность наблюдения значения 11 будет определяться как $\alpha_6^2 = 0.04$, при этом новое состояние системы будет $|011\rangle$.

1.2. Квантовые гейты

Классические дискретные цепи представляют собой совокупность проводников и набора логических гейтов (совокупности полупроводниковых приборов). Логические гейты осуществляют преобразование информации, поступающей на их входы. В квантовых компьютерах преобразование информации осуществляется с использованием так называемых квантовых гейтов [14, 15].

1.2.1. Гейт Адамара

Одиночный кубит по определению является суперпозицией двух квантовых состояний $|0\rangle$ и $|1\rangle$, каждое из которых может рассматриваться как носитель одного бита классической информации $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Чтобы создать состояния суперпозиции, используется гейт, называемый ***гейт Адамара (H). Гейт Адамара*** является одним из наиболее полезных квантовых гейтов. Он задается матрицей:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Учитывая, что состояния кубита могут быть представлены в виде:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix};\ |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix};\ |\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

Запишем выражения показывающие действия оператора Адамара на кубиты.

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle,$$

$$H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} \alpha+\beta \\ \alpha-\beta \end{pmatrix} = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle.$$

Таким образом, действие оператора H на произвольный кубит можно описать формулой:3

$$H|\psi\rangle = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle.$$

В геометрической интерпретации кубита на сфере Блоха можно заметить, что в результате действия гейта Адамара на кубит в состоянии $|0\rangle$ переводит его в положение между состояниями $|0\rangle$ и $|1\rangle$, т.е. в состояние $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. Соответственно, в результате действия гейта Адамара состояние кубита $|1\rangle$ переводит его в положение между

состояниями $|0\rangle$ и $|1\rangle$, только в другой полусфере, т.е. в состояние $\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle$.

Действие гейта Адамара на сфере Блоха показано на рисунке 1.3.

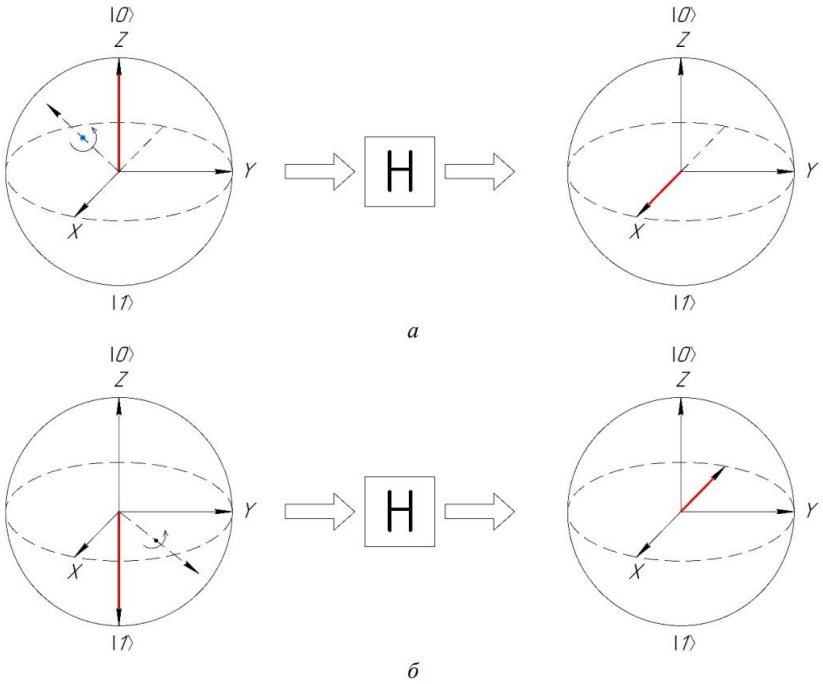


Рис. 1.3. Действие гейта Адамара: а) начальное состояние кубита $|0\rangle$; б) начальное состояние кубита $|1\rangle$

Оператор H является самосопряженным, а, следовательно, обратным к себе самому, его повторное применение к базису Адамара вернет нам обычный базис. Другими словами, алгебраические вычисления дают $H^2=1$. То есть двукратное применение гейта H возвращает систему в исходное положение. Покажем самосопряженность гейта Адамара на примерах.

Пример 1.1. Для кубита в состоянии $|0\rangle$:

$$H|0\rangle=\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}=\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle,$$

$$HH|0\rangle=\frac{1}{\sqrt{2}}\begin{pmatrix}1&1\\1&-1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}=\left(\frac{1}{\sqrt{2}}\right)^2\begin{pmatrix}2\\0\end{pmatrix}=\begin{pmatrix}1\\0\end{pmatrix}=|0\rangle.$$

Пример 1.2. Для кубита в состоянии $|1\rangle$:

$$H|1\rangle=\frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}=\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle,$$

$$HH|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix} = \left(\frac{1}{\sqrt{2}}\right)^2\begin{pmatrix}0\\ 2\end{pmatrix} = \begin{pmatrix}0\\ 1\end{pmatrix} = |1\rangle.$$

Пример 1.3. Для кубита в состоянии $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$:

$$H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle,$$

$$HH|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} = \left(\frac{1}{\sqrt{2}}\right)^2\begin{pmatrix}2\alpha\\ 2\beta\end{pmatrix} = \begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha|0\rangle + \beta|1\rangle.$$

Если у нас имеется система из n кубитов, то применив ***оператор Адамара*** ко всем кубитам индивидуально, мы получим суперпозицию всех 2^n состояний:

$$(H \otimes H \otimes H \otimes H \ldots \otimes H \otimes H)|0000\ldots00\rangle =$$

$$= \frac{1}{\sqrt{2^n}}((|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \ldots \otimes (|0\rangle + |1\rangle)) = \frac{1}{\sqrt{2^n}}\sum_{z=0}^{2^n-1}|z\rangle$$

1.2.2. Оператор NOT (Гейт X)

В классических дискретных вычислениях оператор NOT – это оператор инверсии. В соответствии с этим определением классического оператора NOT, квантовый гейт X (т.е. гейт преобразующий информацию внутри кубита) может быть определен по аналогии:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \Rightarrow NOT|\psi\rangle = \alpha|1\rangle + \beta|0\rangle.$$

Квантовым аналогом классического оператора NOT является матрица вида:

$$X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}.$$

Покажем действия гейта X на кубит в различных состояниях.

$$X|0\rangle = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}0\\ 1\end{pmatrix} = |1\rangle,$$

$$X|1\rangle = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix} = \begin{pmatrix}1\\ 0\end{pmatrix} = |0\rangle,$$

$$X|\psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} = \beta|0\rangle + \alpha|1\rangle .$$

С точки зрения интерпретации действия данного гейта на состояние ***кубита*** с помощью сферы Блоха можно заметить, что происходит поворот вектора состояния на 180 градусов вокруг оси X (рис. 1.4).

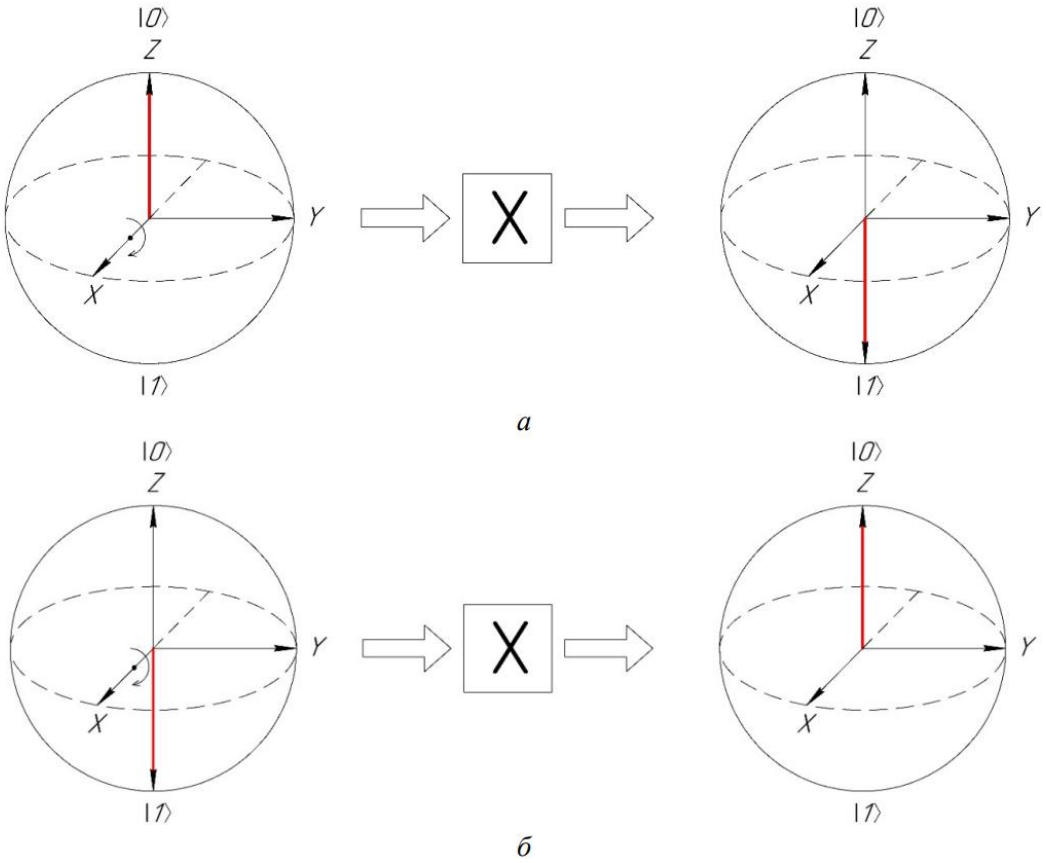


Рис. 1.4. Действие гейта X: а) начальное состояние кубита $|0\rangle$; б) начальное состояние кубита $|1\rangle$

1.2.3. Гейт Z

Определим сначала действие ***гейта*** Z на базисные вектора. Потребуем, чтобы он не изменял 0, а 1 переводил в –1 [11-14]:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \Rightarrow Z|\psi\rangle = \alpha|0\rangle - \beta|1\rangle .$$

Тогда действию данного ***гейта*** Z отвечает матрица:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Покажем действия ***гейта*** Z на кубит в различных состояниях.

$$Z|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle,$$

$$Z|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -|1\rangle,$$

$$Z|\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix} = \alpha|0\rangle - \beta|1\rangle.$$

На сфере Блоха действие гейта Z соответствует повороту вектора вокруг оси Z на угол π (рис. 1.5).

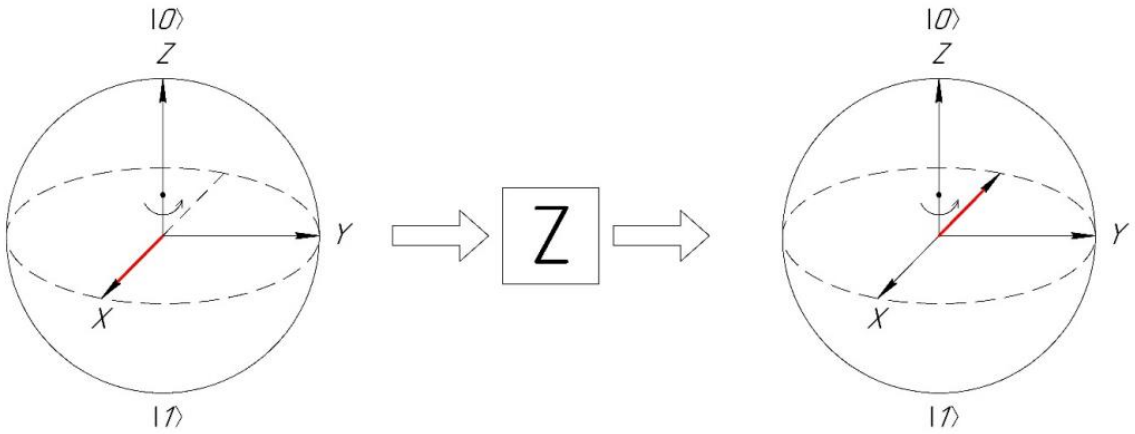


Рис. 1.5. Действие ***гейта*** Z

Отметим следующие два свойства гейтов X и Z**:**

$$HXH = Z,$$

$$HZH = X.$$

Приведем примеры показывающие данные свойства.

Пример 1.4. Для кубита в состоянии $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ применим операции HXH.

$$H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} \alpha+\beta \\ \alpha-\beta \end{pmatrix} = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle,$$

$$XH|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} \alpha+\beta \\ \alpha-\beta \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} \alpha+\beta \\ \alpha-\beta \end{pmatrix} = \frac{\alpha-\beta}{\sqrt{2}}|0\rangle + \frac{\alpha+\beta}{\sqrt{2}}|1\rangle,$$

$$HZH|\psi\rangle = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha+\beta\\ \beta-\alpha\end{pmatrix} = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\begin{pmatrix}2\beta\\ 2\alpha\end{pmatrix} = \beta|0\rangle + \alpha|1\rangle .$$

Как было показано ранее $X|\psi\rangle = \beta|0\rangle + \alpha|1\rangle$, а, следовательно, можно заметить, что $HZH = X$.

1.2.4. Гейт Y

Гейт Y задается матрицей:

$$Y = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}.$$

В отличие от предыдущих рассмотренных нами элементов, ***гейт*** Y является комплексным. Покажем действия ***гейта*** Y на кубит в различных состояниях [14-17].

$$Y|0\rangle = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}0\\ i\end{pmatrix} = i|1\rangle, \quad Y|1\rangle = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix} = \begin{pmatrix}-i\\ 0\end{pmatrix} = -i|1\rangle,$$

$$Y|\psi\rangle = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}-i\beta\\ i\alpha\end{pmatrix} = -i\beta|0\rangle + i\alpha|1\rangle .$$

На сфере Блоха действие ***гейта*** Y соответствует повороту вектора вокруг оси Y на угол π (рис. 1.6).

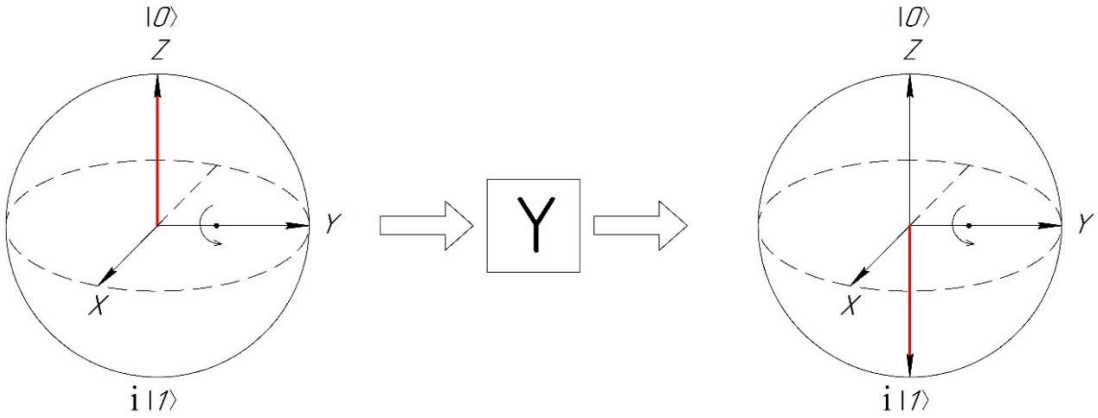


Рис. 1.6. Действие ***гейта*** Y

Отметим следующее свойства ***гейта*** Y:

$$HYH = -Y .$$

Приведем пример, показывающий данное свойство.

Пример 1.5. Для кубита в состоянии $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ применим операции HYH.

$$H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle,$$

$$YH|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\begin{pmatrix} \alpha+\beta \\ \alpha-\beta \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} -i(\alpha-\beta) \\ i(\alpha+\beta) \end{pmatrix} = \frac{-i(\alpha-\beta)}{\sqrt{2}}|0\rangle + \frac{i(\alpha+\beta)}{\sqrt{2}}|1\rangle,$$

$$HYH|\psi\rangle = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} -i(\alpha-\beta) \\ i(\alpha+\beta) \end{pmatrix} = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\begin{pmatrix} 2i\beta \\ -2i\alpha \end{pmatrix} = i\beta|0\rangle - i\alpha|1\rangle.$$

Как было показано ранее $Y|\psi\rangle = -i\beta|0\rangle + i\alpha|1\rangle$, а, следовательно, можно заметить, что $HYH = -Y$.

1.2.5. Гейт S

Гейт S задается матрицей:

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

Покажем действия гейта S на кубит в различных состояниях.

$$S|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = i|0\rangle,$$

$$S|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ i \end{pmatrix} = i|1\rangle,$$

$$S|\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ i\beta \end{pmatrix} = \alpha|0\rangle + i\beta|1\rangle.$$

Покажем действие ***гейта*** S на сфере Блоха (рис. 1.7).

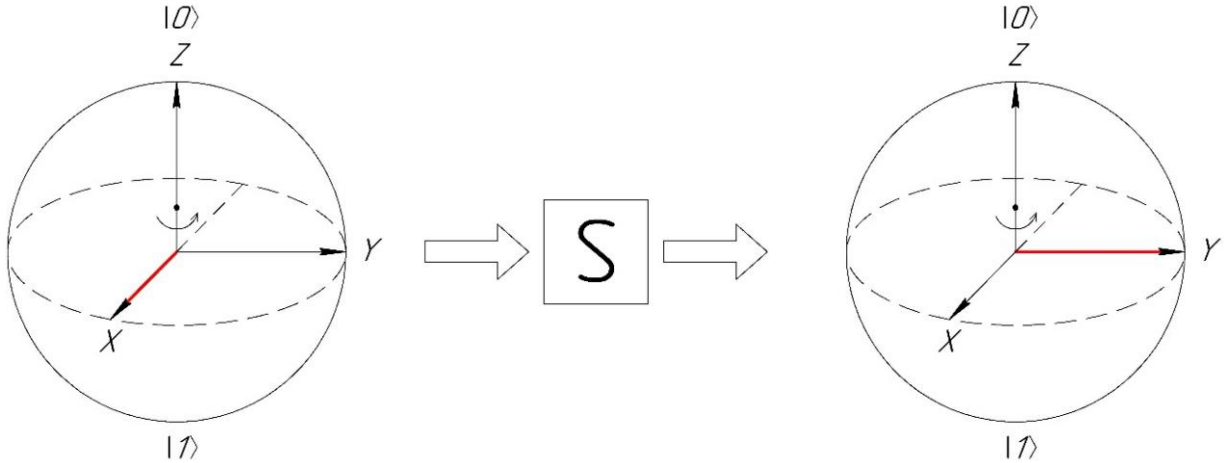


Рис. 1.7 Действие гейта S

1.2.6. Гейт T

Рассмотрим еще один комплексный логический ***гейт*** T, который часто обозначается как $\frac{\pi}{4}$. ***Гейт*** T задается матрицей [18-20]:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}.$$

Название ***гейта*** T определяется историческими причинами и возможностью представления матрицы этого ***гейта*** с точностью до общего фазового множителя $e^{i\frac{\pi}{8}}$ в виде.

$$T = e^{i\frac{\pi}{8}} \begin{pmatrix} e^{-i\frac{\pi}{8}} & 0 \\ 0 & e^{i\frac{\pi}{8}} \end{pmatrix}.$$

Покажем действия ***гейта*** T на ***кубит*** в различных состояниях.

$$T|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = i|0\rangle,$$

$$T|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ e^{i\frac{\pi}{4}} \end{pmatrix} = e^{i\frac{\pi}{4}}|1\rangle,$$

$$T|\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ e^{i\frac{\pi}{4}}\beta \end{pmatrix} = \alpha|0\rangle + e^{i\frac{\pi}{4}}\beta|1\rangle.$$

Покажем действие ***гейта*** T на сфере Блоха (рис. 1.8).

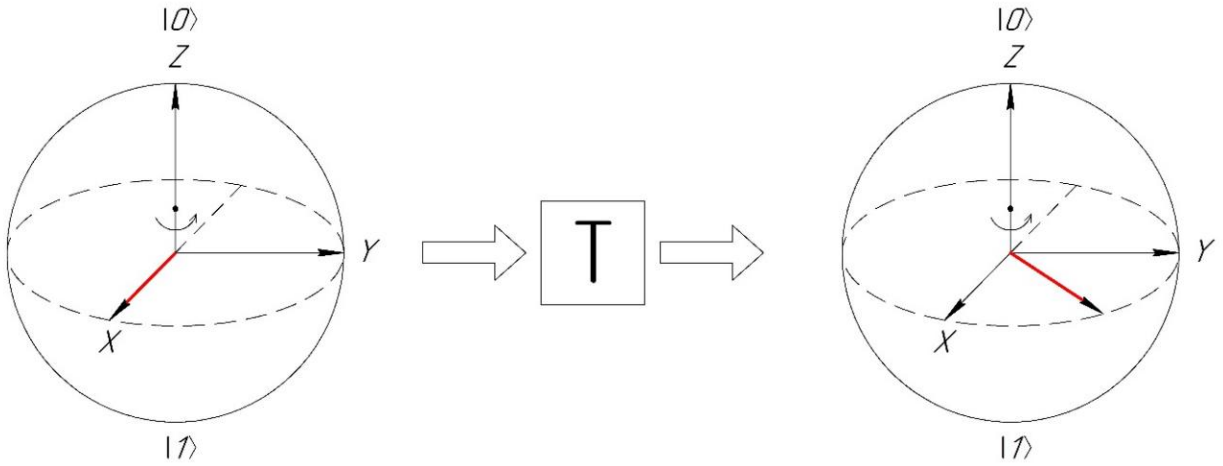


Рис. 1.8. Действие ***гейта*** T

Таким образом, ***гейт*** T не изменяет коэффициент при базисном векторе 0 и меняет фазу коэффициента при базисном векторе 1.

1.2.7. Гейты поворота Rx, Ry, Rz

Гейт Rx на сфере Блоха соответствует вращению кубита вокруг оси X на заданный угол. В матричном виде данный ***гейт*** можно записать как [1, 2, 6]:

$$Rx(\theta) = e^{-i\frac{\theta}{2}X} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\cdot\sin\left(\frac{\theta}{2}\right) \\ -i\cdot\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}.$$

По аналогии с ***гейтом*** Rx, ***гейты*** Ry и Rz соответствуют вращению ***кубита*** вокруг осей Y и Z. При этом их можно определить как:

$$Ry(\theta) = e^{-i\frac{\theta}{2}Y} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix},$$

$$Rz(\theta) = e^{-i\frac{\theta}{2}Z} = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}.$$

1.2.8. Произвольные однокубитные унитарные гейты U

Произвольный однокубитовый унитарный оператор может быть записан в виде [34,35, 40]:

$$U = e^{i\alpha} R_{\vec{n}}(\theta),$$

где $R_{\vec{n}}(\theta)$ – оператор поворота на угол θ вокруг оси, определенной единичным вектором $\vec{n}$, α и θ – действительные числа.

Гейт U1 осуществляет вращение одного кубита вокруг оси Z. В матричном виде данный гейт можно записать как:

$$U1(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}.$$

В зависимости от значения угла λ данный гейт является эквивалентом других гейтов:

$$U1(\pi) = Z,$$

$$U1(\pi/2) = S,$$

$$U1(\pi/4) = T\,.$$

Гейт U2 осуществляет вращение одного кубита вокруг $X+Y$ осей. Согласно теореме $X-Y$ разложения для однокубитового гейта, данный оператор определяется как:

$$U2(\varphi,\lambda) = Rz\left(\varphi+\frac{\pi}{2}\right)Rx\left(\frac{\pi}{2}\right)Rz\left(\lambda-\frac{\pi}{2}\right).$$

В матричном виде данный гейт можно записать как:

$$U2(\varphi,\lambda) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\varphi} & e^{i(\varphi+\lambda)} \end{pmatrix}.$$

Можно заметить, что $U2(0,\pi) = H$.

Гейт U3 – это универсальный однокубитный поворотный затвор с тремя углами Эйлера. Данный гейт определяется как:

$$U3(\theta,\varphi,\lambda) = Rz\left(\varphi-\frac{\pi}{2}\right)Rx\left(\frac{\pi}{2}\right)Rz\left(\pi-0\right)Rx\left(\frac{\pi}{2}\right)Rz\left(\lambda-\frac{\pi}{2}\right),$$

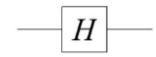
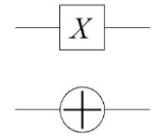
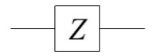
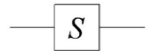
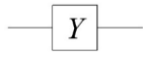
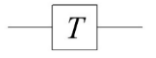
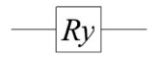
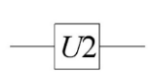
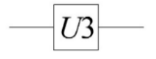
$$U3(\theta,\varphi,\lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\varphi}\sin\left(\frac{\theta}{2}\right) & e^{i(\varphi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}.$$

Можно заметить, что $U3(\theta,-\frac{\pi}{2},\frac{\pi}{2}) = Rx(\theta)$ и $U3(\theta,0,0) = Ry(\theta)$. В таблице 1.1 представлены графические обозначение однобитовых гейтов и результат их воздействия на произвольный кубит $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

Используя гейты поворота R и унитарные гейты U, можно получать произвольные состояния суперпозиции $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Приведем примеры получения различных состояний суперпозиции кубита [12-15].

Таблица 1.1.

Однокубитные гейты

Название	**Графическое обозначение**	**Матрица**	**Результат воздействия**
Гейт **H**		$H=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$	$\frac{\alpha+\beta}{\sqrt{2}}\|0\rangle+\frac{\alpha-\beta}{\sqrt{2}}\|1\rangle$
Гейт **X**		$X=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$	$\beta\|0\rangle+\alpha\|1\rangle$
Гейт **Z**		$Z=\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$	$\alpha\|0\rangle-\beta\|1\rangle$
Гейт **S**		$S=\begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$	$\alpha\|0\rangle+i\beta\|1\rangle$
Гейт **Y**		$Y=\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$	$-i\beta\|0\rangle+i\alpha\|1\rangle$
Гейт **T**		$T=\begin{pmatrix}1 & 0\\ 0 & e^{i\frac{\pi}{4}}\end{pmatrix}$	$\alpha\|0\rangle+e^{i\frac{\pi}{4}}\beta\|1\rangle$
Гейт **Rx**		$Rx(\theta)=\begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & -i\cdot\sin\left(\frac{\theta}{2}\right)\\ -i\cdot\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{pmatrix}$	$\left(\alpha\cos\left(\frac{\theta}{2}\right)-i\beta\sin\left(\frac{\theta}{2}\right)\right)\|0\rangle+$ $+\left(\beta\cos\left(\frac{\theta}{2}\right)+i\alpha\sin\left(\frac{\theta}{2}\right)\right)\|1\rangle$
Гейт **Ry**		$Ry(\theta)=\begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right)\\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{pmatrix}$	$\left(\alpha\cos\left(\frac{\theta}{2}\right)-\beta\sin\left(\frac{\theta}{2}\right)\right)\|0\rangle+$ $+\left(\alpha\sin\left(\frac{\theta}{2}\right)+\beta\cos\left(\frac{\theta}{2}\right)\right)\|1\rangle$
Гейт **Rz**		$Rz(\theta)=\begin{pmatrix}e^{-i\frac{\theta}{2}} & 0\\ 0 & e^{i\frac{\theta}{2}}\end{pmatrix}$	$\alpha e^{-i\frac{\theta}{2}}\|0\rangle+\beta e^{i\frac{\theta}{2}}\|1\rangle$
Гейт **U1**		$U1(\lambda)=\begin{pmatrix}1 & 0\\ 0 & e^{i\lambda}\end{pmatrix}$	$\alpha\|0\rangle+\beta e^{i\lambda}\|1\rangle$
Гейт **U2**		$U2(\varphi,\lambda)=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & -e^{i\lambda}\\ e^{i\varphi} & e^{i(\varphi+\lambda)}\end{pmatrix}$	$\frac{\left(\alpha-\beta e^{i\lambda}\right)}{\sqrt{2}}\|0\rangle+\frac{\left(\alpha e^{i\varphi}+\beta e^{i(\varphi+\lambda)}\right)}{\sqrt{2}}\|1\rangle$
Гейт **U3**		$U3(\theta,\varphi,\lambda)=\begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right)\\ e^{i\varphi}\sin\left(\frac{\theta}{2}\right) & e^{i(\varphi+\lambda)}\cos\left(\frac{\theta}{2}\right)\end{pmatrix}$	$\left(\alpha\cos\left(\frac{\theta}{2}\right)-\beta e^{i\lambda}\sin\left(\frac{\theta}{2}\right)\right)\|0\rangle+$ $+\left(\alpha e^{i\varphi}\sin\left(\frac{\theta}{2}\right)+\beta e^{i(\varphi+\lambda)}\cos\left(\frac{\theta}{2}\right)\right)\|1\rangle$

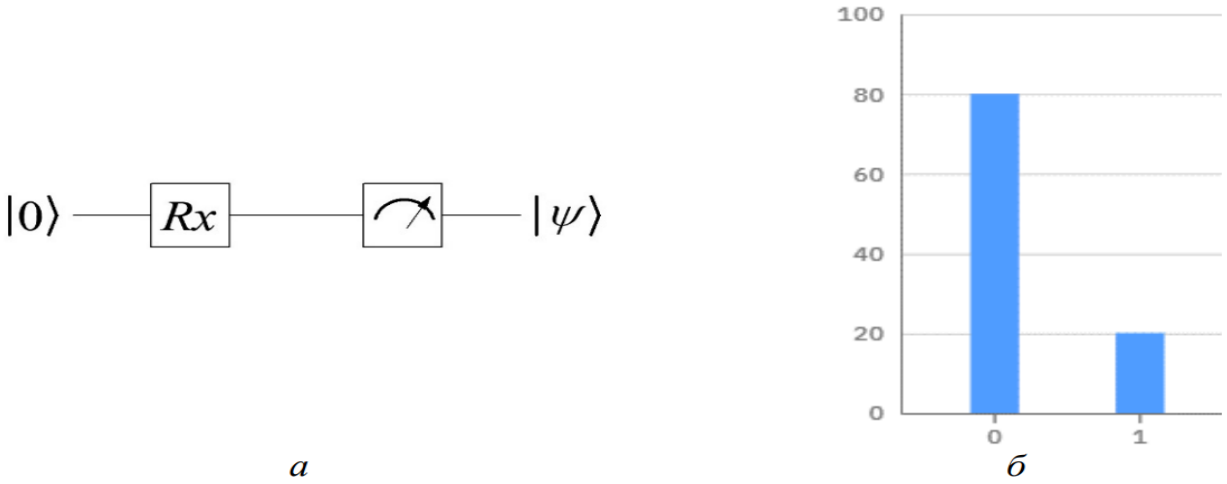
Пример 1.6. Получим кубит в состоянии суперпозиции $|\psi\rangle = \sqrt{0.8}|0\rangle + \sqrt{0.2}|1\rangle$, применив воздействие гейта Rx к ***кубиту***, который находится в исходном состоянии $|0\rangle$:

$$Rx(\theta)|0\rangle = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\cdot\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ -i\cdot\sin\left(\frac{\theta}{2}\right) \end{pmatrix} = \cos\left(\frac{\theta}{2}\right)|0\rangle - i\cdot\sin\left(\frac{\theta}{2}\right)|1\rangle .$$

Таким образом, для получения необходимого состояния нужно рассчитать правильный угол поворота θ . Для этого решим уравнение:

$$\cos\left(\frac{\theta}{2}\right)^2 = 0.8 \Rightarrow \theta \approx 0.93 рад .$$

На рисунке 1.9 представлена квантовая схема, реализующая подобное состояние суперпозиции и результат измерений состояния ***кубита*** с использованием системы **IBM** Quantum Experience.



а *б*

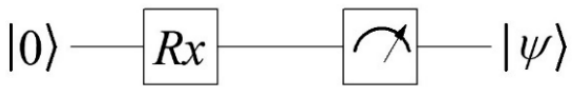
Рис. 1.9 Квантовая схема получения кубита в состоянии суперпозиции $|\psi\rangle = \sqrt{0.8}|0\rangle + \sqrt{0.2}|1\rangle$ (а) и результат симуляции (б)

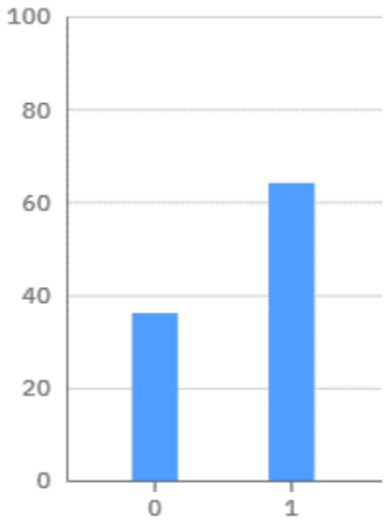
Пример 1.7. Получим кубит в состоянии суперпозиции $|\psi\rangle = 0,6|0\rangle + 0,8|1\rangle$, применив воздействие гейта Rx к кубиту, который находится в исходном состоянии $|0\rangle$. Состояние $|\psi\rangle = 0,6|0\rangle + 0,8|1\rangle$ соответствует тому, что данный кубит будет принимать значение $|0\rangle$ с вероятностью 36% и значение $|1\rangle$ с вероятностью 64%.

Решим уравнение:

$$\cos\left(\frac{\theta}{2}\right)^2 = 0.36 \Rightarrow \theta \approx 1,85 рад .$$

На рисунке 1.10 представлена квантовая схема, реализующая подобное состояние суперпозиции и результат измерений состояния ***кубита*** с использованием системы **IBM** Quantum Experience.





а *б*

Рис. 1.10. Квантовая схема получения кубита в состоянии суперпозиции $|\psi\rangle = 0,6|0\rangle + 0,8|1\rangle$ (а) и результат симуляции (б)

1.2.9. Контролируемые гейты

Для квантовых вычислений, как правило, требуется больше одного кубита [15-17]:

$$|q_1\rangle ... |q_n\rangle = (\alpha_0|0\rangle + \beta|1\rangle) \otimes (\alpha_1|0\rangle + \beta_1|1\rangle) \otimes ... \otimes (\alpha_{n-1}|0\rangle + \beta_{n-1}|1\rangle) .$$

Например, система двух кубитов описывается как:

$$|\psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle = (\alpha_0|0\rangle + \beta_0|1\rangle) \otimes (\alpha_1|0\rangle + \beta_1|1\rangle) =$$
$$= \alpha_0\alpha_1|00\rangle + \alpha_0\beta_1|01\rangle + \beta_0\alpha_1|10\rangle + \beta_0\beta_1|11\rangle$$

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \lambda|10\rangle + \delta|11\rangle .$$

Простейшим двукубитным контролируемым гейтом является ***гейт*** *CNOT*. Данный гейт имеет два кубита на входе и два кубита на выходе. При этом один из кубитов называется контролирующим, а второй – контролируемым. Процесс выполнения ***гейта*** *CNOT*

является следующим: если контролирующий кубит находится в состоянии $|1\rangle$, тогда контролируемый кубит подвергается квантовой операции NOT, если контролирующий кубит находится в состоянии $|0\rangle$, тогда контролируемый кубит остается без изменения. Графическое обозначение квантового ***гейта*** $CNOT$ представлено на рисунке 1.11.

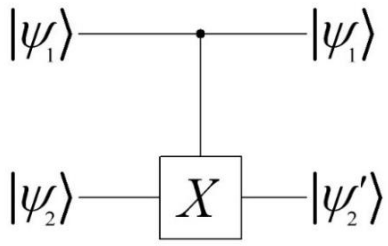


Рис. 1.11. Графическое обозначение квантового ***гейта*** $CNOT$

Для пары кубитов $|\psi_1\rangle$ и $|\psi_2\rangle$ в качестве базисных можно выбрать вектора, являющиеся произведением базисных векторов отдельных кубитов [4, 5]:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \qquad |01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix},$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \qquad |11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}.$$

В общем случае можно записать:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \lambda|10\rangle + \delta|11\rangle = \begin{pmatrix}\alpha\\\beta\\\lambda\\\delta\end{pmatrix}.$$

Исходя из этого, можно определить матрицу ***гейта*** $CNOT$:

$$CNOT = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}.$$

В данном случае для состояния $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \lambda|10\rangle + \delta|11\rangle$ первый (слева) кубит будет контролирующим, а второй кубит контролируемым. При этом действия ***гейта*** *CNOT* будет следующим:

$$CNOT|00\rangle \Rightarrow |00\rangle,$$

$$CNOT|01\rangle \Rightarrow |01\rangle,$$

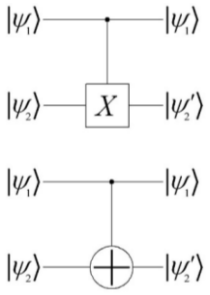
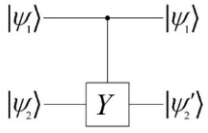
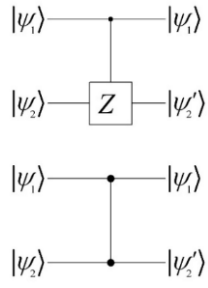
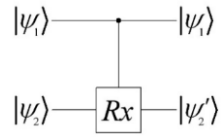
$$CNOT|10\rangle \Rightarrow |10\rangle,$$

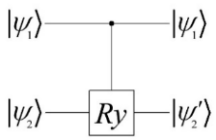
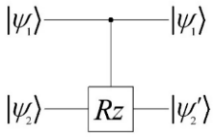
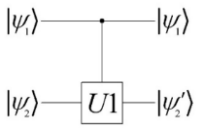
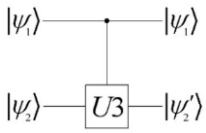
$$CNOT|11\rangle \Rightarrow |11\rangle.$$

Аналогичным образом может быть определен произвольный контролируемый унитарный оператор (табл. 1.2).

Таблица 1.2.

Контролируемые квантовые гейты

Название	**Графическое обозначение**	**Матрица**
CXgate CNOT		$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
CYgate		$CY = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{pmatrix}$
CZgate		$CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$
CRxgate		$CRx = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\theta}{2}\right) & -i\cdot\sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & -i\cdot\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$

CRygate		$CRy = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$
CRzgate		$CRz = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{-i\frac{\lambda}{2}} & 0 \\ 0 & 0 & 0 & e^{i\frac{\lambda}{2}} \end{pmatrix}$
CU1gate		$CU1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\lambda} \end{pmatrix}$
CU3gate		$CU3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & e^{i\varphi}\sin\left(\frac{\theta}{2}\right) & e^{i(\varphi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}$

1.3 Квантовые алгоритмы

Квантовый параллелизм – это фундаментальное свойство квантовых вычислений. Данное свойство позволяет квантовым компьютерам вычислять функцию $f(x)$ для различных значений x одновременно. Как отмечалось ранее, в классических компьютерах состояние бита представляется либо 0, либо 1. В квантовом же компьютере состояние кубита можно представить не только как 0 или 1, а как комбинацию двух этих значений. Поэтому применение какой-либо операции кубиту происходит сразу со всеми его значениями одновременно. В качестве примера рассмотрим действие ***гейта*** x. Допустим, изначально кубит находился в состоянии $|\psi\rangle = 0.8|1\rangle + 0.6|0\rangle$, после действия гейта инверсии кубит будет находиться в состоянии $|\psi\rangle = 0.8|1\rangle + 0.6|0\rangle$. Таким образом, одна операция повлияла сразу на оба значения кубита. Это и есть квантовый параллелизм. Однако для получения информации, которая хранится в суперпозиционном состоянии, нужно выполнить операцию измерения состояний кубитов. При этом измерение кубита всегда дает только один вариант из всего множества возможных вариантов. Квантовый параллелизм

лежит в основе всех квантовых алгоритмов. Благодаря возможности воздействовать сразу на все состояния кубитов системы эффективность квантовых алгоритмов значительно выше, чем у классических компьютерных алгоритмов.

Перепутанные состояния двух кубитов. Базис Белла

Рассмотрим два незапутанных кубита. Незапутанность двух кубитов подразумевает, что измерение первого кубита не влияет на результат измерения второго кубита. Зададим их состояния:

$$|\psi_1\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle,$$

$$|\psi_2\rangle = \beta_0|0\rangle + \beta_1|1\rangle.$$

Состояние пары кубитов получается перемножение одиночных состояний:

$$|\psi\rangle = |\psi_1\psi_2\rangle = |\psi_1\rangle|\psi_2\rangle = (\alpha_0|0\rangle + \alpha_1|1\rangle)(\beta_0|0\rangle + \beta_1|1\rangle),$$

$$|\psi\rangle = |\psi_1\psi_2\rangle = \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle),$$

$$|\psi\rangle = |\psi_1\psi_2\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle).$$

В данном случае, как отмечалось в главе 1, вероятность событий $|00\rangle$, $|01\rangle$, $|10\rangle$ и $|11\rangle$. равна произведению вероятностей его составляющих – $\alpha_0\beta_0$, $\alpha_0\beta_1$, $\alpha_1\beta_0$ и $\alpha_1\beta_1$ соответственно. Перепутанное состояние двух кубитов – это состояние, при котором измерение одного из кубитов однозначно определяет состояние второго кубита. Например, если в результате измерения одного кубита получилось, что состояние $|0\rangle$, то измерение другого кубита однозначно даст значение $|0\rangle$, а если в результате измерения одного кубита получилось, что состояние $|1\rangle$, то измерение другого кубита также даст значение $|1\rangle$.

В этом примере амплитуды вероятности групп $|01\rangle$ и $|10\rangle$ равны нулю. Такое состояние системы двух кубитов можно записать следующим образом [4, 5]:

$$|\psi\rangle = |\psi_1\psi_2\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right),$$

$$|\psi\rangle = |\psi_1\psi_2\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right),$$

$$|\psi\rangle = |\psi_1\psi_2\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right),$$

$$|\psi\rangle = |\psi_1\psi_2\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right).$$

Для понимания специфики подобных состояний можно подробно рассмотреть процессы измерения, например, для состояний:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle,$$

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle).$$

Сначала рассмотрим процесс измерения двухкубитного состояния $|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$. Допустим, что сначала мы проводим измерение первого кубита. В ходе проведения измерения первого кубита мы получим состояние $|0\rangle$, в том случае если мы попадем в группы $|00\rangle$ и $|01\rangle$. Вероятность такого события будет составлять $|\alpha_{00}|^2 + |\alpha_{01}|^2$. При этом первый кубит перешел в состояние $|0\rangle$, а состояния $|10\rangle$ и $|11\rangle$ становятся нереализуемы.

После получения этого результата двухкубитное состояние выглядит так:

$$|\psi\rangle = \frac{\alpha_{00}}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}|00\rangle + \frac{\alpha_{01}}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}|01\rangle,$$

$$|\psi_1\rangle = |0\rangle; \quad |\psi\rangle = \frac{\alpha_{00}}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}|0\rangle + \frac{\alpha_{01}}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}|1\rangle.$$

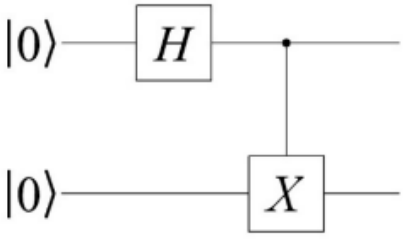
Теперь рассмотрим процесс измерения двухкубитного состояния $|\psi\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle)$.

В данном случае при измерении реализуемы только два состояния: $|01\rangle$ и $|10\rangle$. Если мы проведем измерение первого кубита, то реализуемой остается только одна группа: $|01\rangle$ или $|10\rangle$. Отсюда следует, что состояние второго кубита тоже определится, хотя его мы пока не провели его измерение. Например, если при измерении первого кубита мы получим состояние $|\psi_1\rangle = |0\rangle$ тогда реализуемой остается только группа $|01\rangle$, т.е. второй кубит переходит в

определенной состояние – $|\psi_2\rangle = |1\rangle$. Если же при измерении первого кубита мы получим состояние $|\psi_1\rangle = |1\rangle$ тогда реализуемой остается только группа $|10\rangle$, то есть второй кубит переходит в определенной состояние – $|\psi_2\rangle = |0\rangle$. Также можно описать процессы измерения других перепутанных состояний.

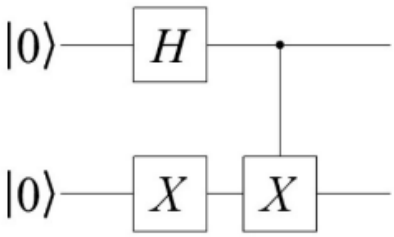
Аналогично обстоит дело и с более сложными системами: трехкубиными, четырехкубитными и так далее. Измерение одного кубита всегда выводит его из запутанности с остальными кубитами и приводит в одно из двух чистых базисных состояний. Квантовое состояние оставшейся части системы кубитов скачкообразно изменяется строго определенным образом. Если же измеряемый кубит не запутан с прочими кубитами системы, то при его измерении с остальными кубитами ничего не происходит.

На рисунке 1.12 представлены способы реализации состояний Белла системы двух кубитов.



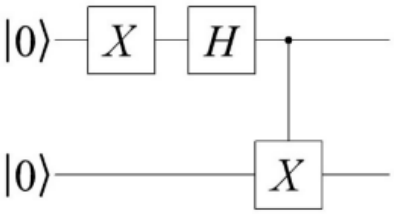
$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

а



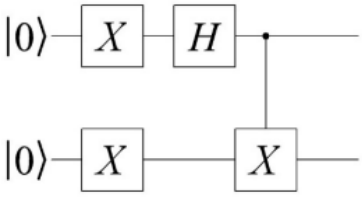
$$|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$

б



$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$

в



$$|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

г

Рис. 1.12. Примеры реализации состояний Белла

1.3.1.Сверхплотное кодирование

Приведем пример применения перепутанности квантовых состояний. Допустим, Алиса хочет передать Бобу одну из цифр от 0 до 3. Для организации передачи информации между Бобом и Алисой каждому из них пересылается один из двух кубитов приготовленных в запутанном состоянии, например, $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ Пусть Алиса получает первый кубит, а Боб второй. При этом Алиса может осуществлять преобразование на своем кубите, а Боб на своем[2].

Алгоритм обмена информации будет следующим: 1. Алиса получает извне два классических бита, которые кодируют цифры от 0 до 3. В зависимости от значения числа Алиса совершает одно из преобразований I , X , Y или Z : Для цифры 0:

$$|\psi_0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix};$$

$$0 \Rightarrow (I \otimes I)|\psi_0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

Таким образом, получаем: $0 \Rightarrow (I \otimes I)|\psi_0\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle)$.

Для цифры 1:

$$0 \Rightarrow (X \otimes I)|\psi_0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

Таким образом, получаем: $1 \Rightarrow (X \otimes I)|\psi_0\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle).$

Аналогично можно получить преобразования для цифр 2 и 3:

$$2 \Rightarrow (Y \otimes I)|\psi_0\rangle = \frac{1}{\sqrt{2}}(-|10\rangle + |01\rangle);$$

$$3 \Rightarrow (Z \otimes I)|\psi_0\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle).$$

2. Далее Алиса передает свой кубит Бобу.

3. Боб применяет ***гейт*** $CNOT$ к двум запутанным кубитам:

$$0 \Rightarrow \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \Rightarrow CNOT \Rightarrow \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle);$$

$$1 \Rightarrow \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) \Rightarrow CNOT \Rightarrow \frac{1}{\sqrt{2}}(|11\rangle + |01\rangle);$$

$$2 \Rightarrow \frac{1}{\sqrt{2}}(-|10\rangle + |01\rangle) \Rightarrow CNOT \Rightarrow \frac{1}{\sqrt{2}}(-|11\rangle + |01\rangle);$$

$$3 \Rightarrow \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \Rightarrow CNOT \Rightarrow \frac{1}{\sqrt{2}}(|00\rangle - |10\rangle).$$

1. Далее Боб производит измерение второго кубита. В результате измерений он получит состояние $|0\rangle$ для цифр 0 и 3, состояние $|1\rangle$ для цифр 1 и 2:

$$0 \Rightarrow \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \Rightarrow \begin{cases} \textit{Первый кубит } \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ \textit{Второй кубит } |0\rangle \end{cases};$$

$$1 \Rightarrow \frac{1}{\sqrt{2}}(|11\rangle + |01\rangle) \Rightarrow \begin{cases} \textit{Первый кубит } \frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) \\ \textit{Второй кубит } |1\rangle \end{cases};$$

$$2 \Rightarrow \frac{1}{\sqrt{2}}\left(-|11\rangle + |01\rangle\right) \Rightarrow \begin{cases} \textit{Первый кубит } \frac{1}{\sqrt{2}}(-|1\rangle + |0\rangle) \\ \textit{Второй кубит } |1\rangle \end{cases};$$

$$3 \Rightarrow \frac{1}{\sqrt{2}}\left(|00\rangle - |10\rangle\right) \Rightarrow \begin{cases} \textit{Первый кубит } \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ \textit{Второй кубит } |0\rangle \end{cases}.$$

Результат измерения второго кубита:

$$|0\rangle \Rightarrow \begin{cases} \textit{Либо } 0 \\ \textit{Либо } 3 \end{cases} \qquad |1\rangle \Rightarrow \begin{cases} \textit{Либо } 1 \\ \textit{Либо } 2 \end{cases}$$

2. Далее Боб применяет ***преобразование Адамара*** к первому кубиту и измеряет его:

$$|0\rangle \Rightarrow \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) + \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)\right) = |0\rangle;$$

$$|1\rangle \Rightarrow \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) + \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)\right) = |0\rangle;$$

$$|2\rangle \Rightarrow \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\left(-|0\rangle + |1\rangle\right) + \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)\right) = |1\rangle;$$

$$|3\rangle \Rightarrow \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) + \frac{1}{\sqrt{2}}\left(-|0\rangle + |1\rangle\right)\right) = |1\rangle.$$

3. Таким образом, выполнив преобразования и измерив два ***кубита***, Боб понимает, какую цифру передавала Алиса:

$$\left.\begin{matrix} \textit{Первый кубит} \to |0\rangle \\ \textit{Второй кубит} \to |0\rangle \end{matrix}\right\} \to \textit{цифра } 0;$$

$$\left.\begin{matrix} \textit{Первый кубит} \to |0\rangle \\ \textit{Второй кубит} \to |1\rangle \end{matrix}\right\} \to \textit{цифра } 1;$$

$$\left.\begin{matrix} \textit{Первый кубит} \to |1\rangle \\ \textit{Второй кубит} \to |1\rangle \end{matrix}\right\} \to \textit{цифра } 2;$$

$$\left.\begin{matrix} \textit{Первый кубит} \to |1\rangle \\ \textit{Второй кубит} \to |0\rangle \end{matrix}\right\} \to \textit{цифра } 3.$$

Исходя из представленного выше описания сврехплотного кодирования, можно построить квантовую схему (рис. 3.2).

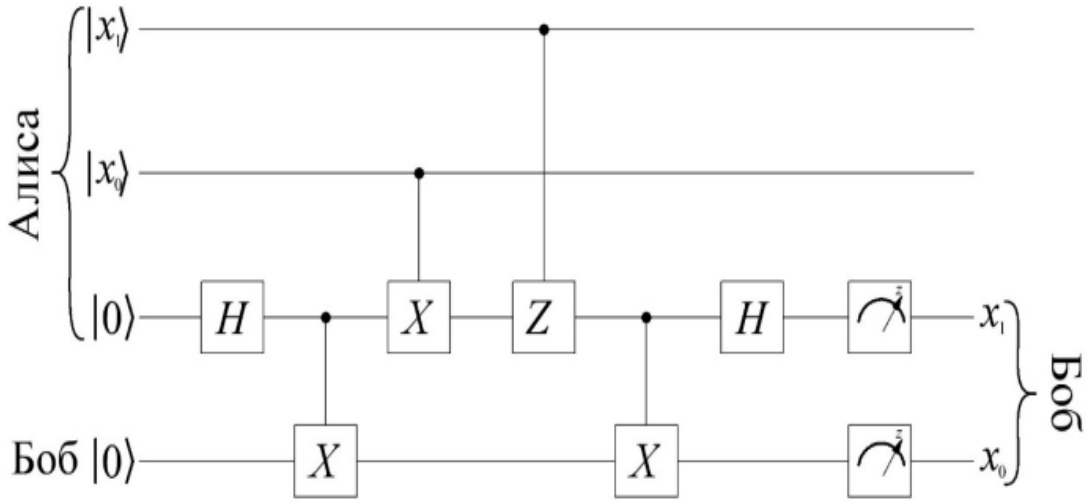


Рис. 1.13. Квантовая схема сверхплотного кодирования

На схеме, представленной на рисунке 1.13, передаваемая цифра кодируется в двоичном коде 1 x_1x_0. При этом один из запутанных ***кубитов*** принадлежит Алисе и в зависимости от передаваемой цифры Алиса применяет к нему соответствующее преобразование. После преобразования своего кубита Боб применяет операцию распутывания кубитов посредствам операции контролируемого *NOT* и гейта Адамара H. После этого производится операция измерения двух кубитов.

1.3.3. Квантовая телепортация

Задача квантовой телепортации заключается в переносе неизвестного квантового состояния с одной системы на другую с использованием квантового канала связи. Так как квантовое состояние не может быть скопировано (теорема о неклонируемости), то в процессе передачи исходное состояние кубита будет утеряно. При осуществлении квантовой телепортации исходное состояние кубита будет утеряно, но при этом оно будет воссоздано у получателя [14, 15, 20].

Рассмотрим пример квантовой телепортации. Допустим, Алиса хочет передать Бобу кубит в состоянии суперпозиции $|\psi_1\rangle = \alpha|0\rangle + \beta|1\rangle$. Кубит в данном состоянии находится у Алисы. Для организации передачи информации между Бобом и Алисой каждому из них пересылается один из двух кубитов приготовленных в запутанном состоянии, например, $|\psi_{23}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

На рисунке 1.14 показано распределение кубитов между Бобом и Алисой.

$$|\psi_1\rangle = \alpha|0\rangle + \beta|1\rangle \rightarrow Первый\ бит - Алиса$$

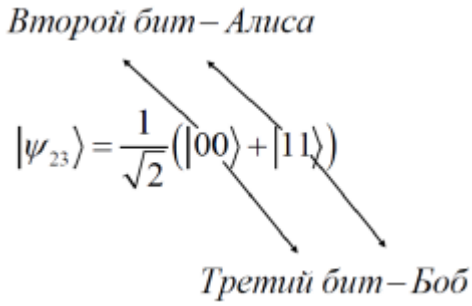


Рис. 1.14 Кубиты Алисы и Боба для квантовой телепортации

Таким образом, можно говорить, что мы имеем трехкубитовую систему:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle)$$

Далее Алиса применяет операцию контролируемой инверсии, причем передаваемый кубит 1 $|\psi_1\rangle = \alpha|0\rangle + \beta|1\rangle$ будет являться контролирующим. Тогда трехбитовая система будет иметь вид:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|110\rangle + \beta|101\rangle).$$

После операции *CNOT* Алиса выполняет преобразование Адамара с передаваемым кубитом. Для дальнейшего описания системы вспомним, что $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ и $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Следовательно, состояние системы после преобразования Адамара можно представить в виде:

$$\alpha|000\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|100\rangle),$$

$$\alpha|011\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(\alpha|011\rangle + \alpha|111\rangle),$$

$$\beta|110\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(\beta|010\rangle - \beta|110\rangle),$$

$$\beta|110\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(\beta|010\rangle - \beta|110\rangle),$$

$$|\psi\rangle = \frac{1}{2}\left(\alpha|000\rangle + \alpha|100\rangle + \alpha|011\rangle + \alpha|111\rangle + \beta|010\rangle - \beta|110\rangle + \beta|001\rangle - \beta|101\rangle\right).$$

Напомним, что в данной системе два кубита принадлежат Алисе и один кубит – Бобу. Если выделить в волновой функции системы кубиты Алисы, то ее можно записать следующим образом:

$$|\psi\rangle = \frac{1}{2}\left(|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |11\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)\right).$$

Из представленной выше функции можно заметить, что кубит, который принадлежит Бобу, может быть преобразован в исходный (передаваемый) кубит. Причем необходимое преобразование зависит от значений двух кубитов Алисы. Распишем необходимые преобразования для получения Бобом исходного кубита:

1. Если кубиты Алисы при измерении принимают значения $|00\rangle$, то в данном случае Бобу не нужно выполнять никаких преобразований и его кубит будет иметь волновую функцию $\alpha|0\rangle + \beta|1\rangle$.

2. Если кубиты Алисы при измерении принимают значения $|01\rangle$, то в данном случае кубит, который принадлежит Бобу, находится в состоянии $\alpha|1\rangle + \beta|0\rangle$. Можно заметить, что для получения исходного кубита Бобу необходимо выполнить операцию инверсии:

$$\alpha|1\rangle + \beta|0\rangle \xrightarrow{X} \alpha|0\rangle + \beta|1\rangle.$$

3. Если кубиты Алисы при измерении принимают значения $|10\rangle$, то в данном случае кубит, который принадлежит Бобу, находится в состоянии $\alpha|0\rangle - \beta|1\rangle$. Можно заметить, что для получения исходного кубита Бобу необходимо применить гейт Z:

$$\alpha|0\rangle - \beta|1\rangle \xrightarrow{Z} \alpha|0\rangle + \beta|1\rangle.$$

4. Если кубиты Алисы при измерении принимают значения $|11\rangle$, то в данном случае кубит, который принадлежит Бобу, находится в состоянии $\alpha|1\rangle - \beta|0\rangle$. Можно заметить, что для получения исходного кубита Бобу необходимо применить гейты Z и X:

$$\alpha|1\rangle - \beta|0\rangle \xrightarrow{X} \alpha|0\rangle - \beta|1\rangle,$$
$$\alpha|0\rangle - \beta|1\rangle \xrightarrow{Z} \alpha|0\rangle + \beta|1\rangle.$$

Исходя из алгоритма телепортации, можно построить квантовые схемы ее реализующие (рис. 1.15).

В представленной на рисунке 1.15 схеме передаваемым кубитом является $|\psi_1\rangle$. При разработке схемы телепортации данный кубит должен находиться в состоянии суперпозиции. Получить подобное состояние можно с использованием гейтов поворота U или R.

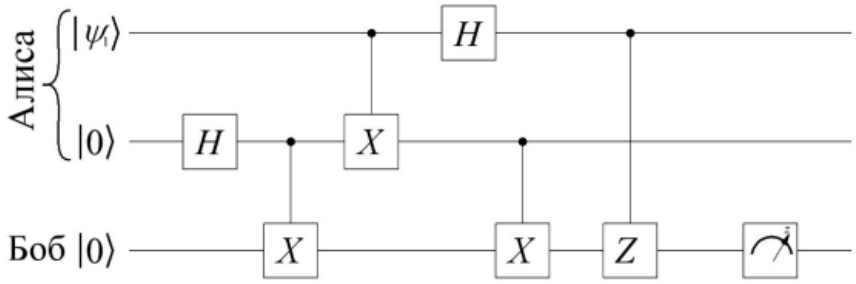


Рис. 1.15. Схема алгоритма квантовой телепортации

1.3.4. Алгоритм Дойча. Задача Дойча-Джозы

Принцип работы квантового компьютера и преимущества квантовых вычислений наиболее просто показать на примере простейшего квантового алгоритма – алгоритма Дойча [34, 35, 40]. Опишем задачу, которую решает данный алгоритм. Предположим, что у нас имеется «черный ящик», который вычисляет неизвестную нам функцию одной переменной – $f(x)$. Так как это функция одной переменной, то на вход можно подать 0 или 1 и на выходе также можно получить или 0, или 1. Функции одной переменной можно разделить на две группы: константные и сбалансированные. На выходе первых мы всегда будем получать постоянное значение 0 или 1 не зависимо от того, что подано на вход. Константных функций одной переменной являются функции вида: $f(x)=0$ и $f(x)=1$. Сбалансированными функциями одной переменной являются функции тождественного равенства и инверсии: $f(x)=x$ и $f(x)=\overline{x}$.

Задача Дойча состоит в том, чтобы определить к какой из двух групп (константная или сбалансированная) относится функция, реализуемая «черным ящиком». Решение подобной задачи с использованием классического компьютера сводится к тому, что необходимо на вход схемы подать сначала 0, а потом 1. То есть на классическом компьютере нам необходимо будет два раза вызвать функцию и измерить выходную реакцию. После двух данных операций мы однозначно можем идентифицировать не только к какой

группе относится функция, но и вид данной функции. Если же мы на классическом компьютере вызовем функцию один раз, то не сможем определить даже группу, к которой она относится. Однако использование квантового компьютера позволит определить группу за один вызов функции. В квантовой системе в качестве функции в «черном ящике» должен быть реализован унитарный оператор U_f. Данный оператор называется квантовым оракулом. Квантовый оракул – это многокубитный гейт, который соответствует бинарной функции, содержит в себе информацию о функции $f(x)$ и позволяет одновременно вызвать ее на всех возможных аргументах. Квантовый оракул определяется как преобразование $U_f|x\rangle|y\rangle=|x\rangle|y\oplus f(x)\rangle$, где, в общем случае, множество кубитов $|x\rangle$ несут в себе информацию об аргументах функции (остаются неизменными), а кубит $|y\rangle$ – результат. При этом размерность квантового оракула будет составлять $n+1$при размерности функции $f(x)-n$.

Определив квантовый оракул как преобразование $U_f|x\rangle|y\rangle=|x\rangle|y\oplus f(x)\rangle$ можно показать его воздействие на состояни $\frac{1}{\sqrt{2}}|x\rangle(|0\rangle-|1\rangle)$:

$$U_f\frac{1}{\sqrt{2}}|x\rangle(|0\rangle-|1\rangle)=\frac{1}{\sqrt{2}}|x\rangle\left(|0\oplus f(x)\rangle-|1\oplus f(x)\rangle\right)=\frac{1}{\sqrt{2}}(-1)^{f(x)}|x\rangle\left(|0\rangle-|1\rangle\right).$$

Давайте получим матрицы и схемы оракулов для всех бинарных функций одной переменной:

$$f(x)=0,\ f(x)=1,\ f(x)=x \text{ и } f(x)=\bar{x}.$$

1. Функция $f(x)=0$. В соответствии с воздействием $|x\rangle|y\oplus f(x)\rangle$ рассмотрим в какие состояния должны перейти все возможные базисные состояния регистра:

$$|00\rangle=|0\rangle|0\rangle\xrightarrow{U_f}|0\rangle|0\oplus f(0)\rangle=|0\rangle|0\oplus 0\rangle=|00\rangle,$$
$$|01\rangle=|0\rangle|1\rangle\xrightarrow{U_f}|0\rangle|1\oplus f(0)\rangle=|0\rangle|1\oplus 0\rangle=|01\rangle,$$
$$|10\rangle=|1\rangle|0\rangle\xrightarrow{U_f}|1\rangle|0\oplus f(1)\rangle=|1\rangle|0\oplus 0\rangle=|10\rangle,$$
$$|11\rangle=|1\rangle|1\rangle\xrightarrow{U_f}|1\rangle|1\oplus f(0)\rangle=|1\rangle|1\oplus 0\rangle=|11\rangle.$$

Таким образом, матрицу оракула U_f можно представить в виде:

$$U_f=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}.$$

Исходя из полученной матрицы оракула, можно представить его квантовую схему (рис. 1.16).

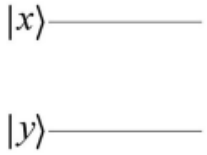


Рис. 1.16. Квантовая схема оракула функции $f(x)=0$

2. Функция $f(x)=1$. Рассмотрим, в какие состояния должны перейти все возможные базисные состояния регистра

$$|00\rangle=|0\rangle|0\rangle\xrightarrow{U_f}|0\rangle|0\oplus f(0)\rangle=|0\rangle|0\oplus 1\rangle=|01\rangle,$$
$$|00\rangle=|0\rangle|1\rangle\xrightarrow{U_f}|0\rangle|1\oplus f(0)\rangle=|0\rangle|1\oplus 1\rangle=|00\rangle,$$
$$|10\rangle=|1\rangle|0\rangle\xrightarrow{U_f}|1\rangle|0\oplus f(1)\rangle=|1\rangle|0\oplus 1\rangle=|11\rangle,$$
$$|11\rangle=|1\rangle|1\rangle\xrightarrow{U_f}|1\rangle|1\oplus f(0)\rangle=|1\rangle|1\oplus 1\rangle=|10\rangle.$$

Таким образом, матрицу оракула U_f можно представить в виде:

$$U_f=\begin{pmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}.$$

Как видно из полученной матрицы оракула, – в данном случае схема будет представлять собой инверсию $|y\rangle$ (рис. 1.17).

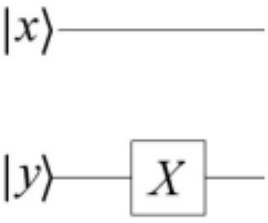


Рис. 1.17. Квантовая схема оракула функции $f(x)=1$

3. Функция $f(x)=x$. Рассмотрим, в какие состояния должны перейти все возможные базисные состояния регистра:

$$|00\rangle=|0\rangle|0\rangle\xrightarrow{U_f}|0\rangle|0\oplus f(0)\rangle=|0\rangle|0\oplus 0\rangle=|00\rangle,$$
$$|01\rangle=|0\rangle|1\rangle\xrightarrow{U_f}|0\rangle|1\oplus f(0)\rangle=|0\rangle|1\oplus 0\rangle=|01\rangle,$$
$$|10\rangle=|1\rangle|0\rangle\xrightarrow{U_f}|1\rangle|0\oplus f(1)\rangle=|1\rangle|0\oplus 1\rangle=|11\rangle,$$
$$|11\rangle=|1\rangle|1\rangle\xrightarrow{U_f}|1\rangle|1\oplus f(0)\rangle=|1\rangle|1\oplus 1\rangle=|10\rangle.$$

Таким образом, матрицу оракула U_f можно представить в виде:

$$U_f = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

В данном случае квантовый оракул представляет собой контролируемый гейт x, причем кубит $|x\rangle$ является контролирующим (рис. 1.18).

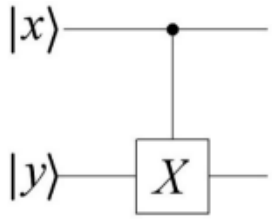


Рис. 1.18. Квантовая схема оракула функции $f(x) = x$

4. Функция $f(x) = \bar{x}$. Рассмотрим, в какие состояния должны перейти все возможные базисные состояния регистра:

$$|00\rangle = |0\rangle|0\rangle \xrightarrow{U_f} |0\rangle|0 \oplus f(0)\rangle = |0\rangle|0 \oplus 1\rangle = |01\rangle,$$
$$|01\rangle = |0\rangle|1\rangle \xrightarrow{U_f} |0\rangle|1 \oplus f(0)\rangle = |0\rangle|1 \oplus 1\rangle = |00\rangle,$$
$$|10\rangle = |1\rangle|0\rangle \xrightarrow{U_f} |1\rangle|0 \oplus f(1)\rangle = |1\rangle|0 \oplus 0\rangle = |10\rangle,$$
$$|11\rangle = |1\rangle|1\rangle \xrightarrow{U_f} |1\rangle|1 \oplus f(0)\rangle = |1\rangle|1 \oplus 0\rangle = |11\rangle.$$

Таким образом, матрицу оракула U_f можно представить в виде:

$$U_f = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Подобный оператор можно реализовать различными схемами (рис. 1.19).

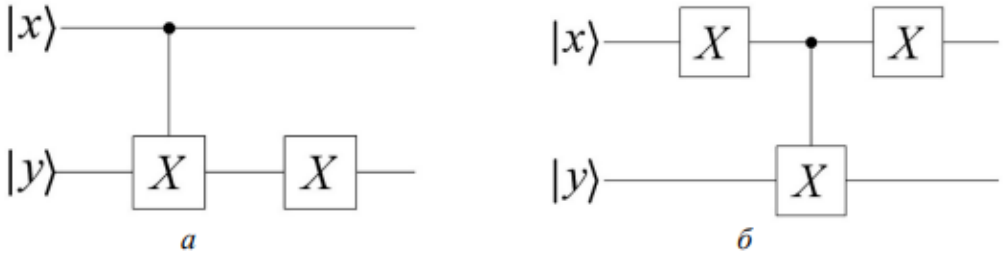


Рис. 1.19. Квантовая схема оракула функции $f(x) = \bar{x}$

Итак, описав оракулы, мы можем вернуться к разработке алгоритма, реализующего задачу Дойча. Схема данного алгоритма представлена на рисунке 1.20.

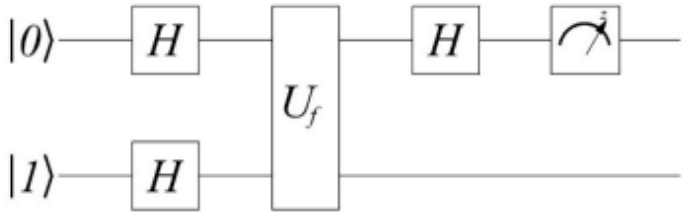


Рис. 1.20. Схема алгоритма Дойча

Приведем описание, как данная схема работает. Применим преобразование Адамара к каждому кубиту двукубитовой системы:

$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}\left(|0\rangle+|1\rangle\right);$$

$$|1\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}\left(|0\rangle-|1\rangle\right);$$

$$|xy\rangle = |01\rangle \xrightarrow{H} \frac{1}{2}\left(|0\rangle+|1\rangle\right)\left(|0\rangle-|1\rangle\right) = \frac{1}{2}|0\rangle\left(|0\rangle-|1\rangle\right)+\frac{1}{2}|1\rangle\left(|0\rangle-|1\rangle\right).$$

Далее на систему действует оператор U_f.

$$U_f \frac{1}{\sqrt{2}}|x\rangle\left(|0\rangle-|1\rangle\right) = \frac{1}{\sqrt{2}}(-1)^{f(x)}|x\rangle\left(|0\rangle-|1\rangle\right);$$

$$\frac{1}{2}|0\rangle\left(|0\rangle-|1\rangle\right)+\frac{1}{2}|1\rangle\left(|0\rangle-|1\rangle\right) \xrightarrow{U_f} \frac{1}{2}\left((-1)^{f(0)}|0\rangle+(-1)^{f(1)}|1\rangle\right)\left(|0\rangle-|1\rangle\right).$$

Далее по алгоритму применяется преобразование Адамара к первому кубиту, и после этого производится его измерение. Рассмотрим оба случая, т.е. когда функция является константной или сбалансированной.

- В случае если функция является константной, то $f(0)=f(1)$. В данном случае после воздействия оператора U_f первый кубит будет находиться либо в состоянии $\frac{1}{\sqrt{2}}\left(|0\rangle+|1\rangle\right)$, либо в состоянии $-\frac{1}{\sqrt{2}}\left(|0\rangle+|1\rangle\right)$. При этом воздействие оператора Адамара приведет его в состояния – $|0\rangle$ или $-|0\rangle$ соответственно. В любом случае измерение покажет, что кубит находится в нулевом состоянии.
- В случае если функция является сбалансированной, то $f(0)\neq f(1)$. После воздействия оператора U_f первый кубит будет находиться либо в состоянии $\frac{1}{\sqrt{2}}\left(|0\rangle-|1\rangle\right)$, либо в состоянии $-\frac{1}{\sqrt{2}}\left(|0\rangle-|1\rangle\right)$. Последующее воздействие оператора Адамара приведет его в состояния $|1\rangle$ или $-|1\rangle$, соответственно, и измерение покажет, что кубит находится в единичном состоянии.

Далее рассмотрим с вами решения задачи Дойча-Джозы, которая является обобщенной задачей Дойча. Данная задача формулируется следующим образом: пусть имеется бинарная функция: $f:\{0,1\}^n \to \{0,1\}$, и известно, что данная функция может быть константной или сбалансированной. Функция с n входными переменными будет сбалансированной, если она принимает значение 0 на 2^{n-1} входных наборах и значение 1 на 2^{n-1} входных наборах. Задача Дойча-Джозы, как и задача Дойча, заключается в том, что необходимо определить, является ли функция в «черном ящике» сбалансированной или константной. По аналогии с предыдущей задачей, схема алгоритма Дойча-Джозы будет иметь вид, представленный на рисунке 1.21.

Выполним описание данного алгоритма. Для этого рассмотрим действие оператора Адамара на произвольный квантовый регистр $|x\rangle^n$:

$$|x\rangle^n \to |x_1 x_2 ... x_n\rangle \xrightarrow{H} H|x_1\rangle \otimes H|x_2\rangle \otimes ... \otimes H|x_n\rangle =$$

$$= \frac{1}{\sqrt{2^n}} \left(\sum_{y_1 \in \{0,1\}} (-1)^{x_1 y_1} |y_1\rangle \otimes \sum_{y_2 \in \{0,1\}} (-1)^{x_2 y_2} |y_2\rangle \otimes ... \otimes \sum_{y_n \in \{0,1\}} (-1)^{x_n y_n} |y_n\rangle \right) =$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y \in B_n} (-1)^{x_1 y_1 \otimes x_2 y_2 \otimes ... \otimes x_n y_n} |y\rangle^n$$

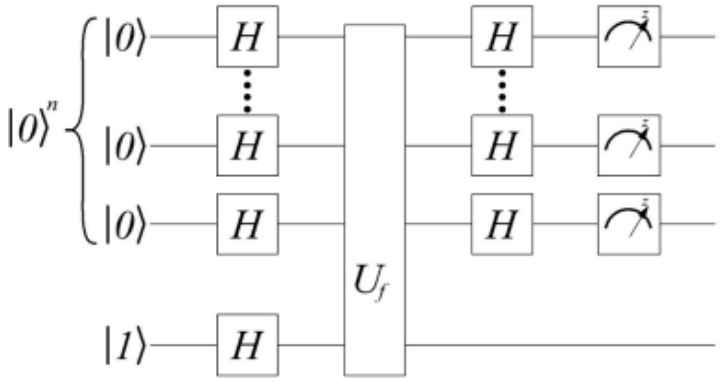


Рис. 1.21. Схема алгоритма Дойча-Джозы

В начальный момент времени система (рис. 3.10) находится в состоянии $|0\rangle^n|1\rangle$. Если применить операцию H_n к состоянию $|0\rangle^n$, то мы получим следующий результат $H_n|0\rangle^n = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$, следовательно, для состояния $|0\rangle^n|1\rangle$ получим:

$$|0\rangle^n|1\rangle \xrightarrow{H_{n+1}} = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle(|0\rangle - |1\rangle).$$

Воздействие оракула U_f переведет систему в состояние:

$$\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle(|0\rangle-|1\rangle)\xrightarrow{U_f}\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle(|0\rangle-|1\rangle).$$

Если функция $f(x)$ является константной и, учитывая, что $H_n|0\rangle^n=\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle$, то первые n кубитов находятся в состоянии

$$\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle=(-1)^{f(x)}H_n|0\rangle^n.$$

Повторное применение оператора Адамара к состоянию $H_n|0\rangle^n$ приведет к тому, что при измерении (рис. 1.10) с вероятностью 100% будет получено состояние $|0\rangle^n$.

Если же функция $f(x)$ является сбалансированной, то $(-1)^{f(x)}-$ нельзя вынести из под суммы и состояние первых n кубитов можно представить как [1].

$$\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle=\frac{1}{\sqrt{2^n}}\left(\sum_{x:f(x)=0}|x\rangle-\sum_{x:f(x)=1}|x\rangle\right).$$

Отметим, что

$$H_n|x\rangle^n=\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{x_1y_1\otimes x_2y_2\otimes\ldots\otimes x_ny_n}|y\rangle=\frac{1}{\sqrt{2^n}}\left(|0\rangle^n+\sum_{y=1}^{2^n-1}(-1)^{x_1y_1\otimes x_2y_2\otimes\ldots\otimes x_ny_n}|y\rangle\right).$$

Если функция $f(x)$ сбалансирована, то число слагаемых в суммах выражения: $\frac{1}{\sqrt{2^n}}\left(\sum_{x:f(x)=0}|x\rangle-\sum_{x:f(x)=1}|x\rangle\right)$ одинаково, и при повторно применении оператора Адамара векторы $|0\rangle^n$ будут отсутствовать. Таким образом, в случае сбалансированной функции при измерении системы вероятность получения состояния $|0\rangle^n$ будет равно нулю.

1.3.5. Алгоритм Гровера

Алгоритм Гровера направлен на решение следующей задачи: в базе неупорядоченных данных, состоящей из n элементов, требуется найти элемент с заданными свойствами [4, 5, 10]. Для решения данной задачи на классическом компьютере в среднем потребуется перебрать $\frac{n}{2}$ элементов базы, а в наихудшем случае – $n-1$ элементов. Квантовый алгоритм Гровера позволит выполнить данную задачу всего за $\sqrt{n}$ операций [15]. Задачу поиска можно сформулировать

следующим образом: пусть имеется бинарная функция: $f:\{0,1\}^n \to \{0,1\}$, и известно, что данная функция принимает значение 0 на всех входных наборах, кроме x_0, и задача состоит в том, чтобы найти x_0. Очевидно, что для реализации алгоритма поиска необходимо получит оракул данной функции:

$$U_f|x\rangle|y\rangle = |x\rangle|y \otimes f(x)\rangle,$$

$$|x\rangle - \textit{входной регистр},$$

$$|y\rangle - \textit{входной регистр}.$$

Вычисление функции $f(x)$ для всевозможных входных наборов $|x\rangle$ за счет применения оператора U_f, при начальном значении выходного регистра равному 0, приведет к суперпозиции $\frac{1}{\sqrt{2^n}}\sum_x |x\rangle|f(x)\rangle$. Для входного набора x_0, при котором $f(x_0)=1$, состояние $|x_0\rangle|1\rangle$ будет иметь вероятность 2^{-n}. Суть алгоритма Гровера состоит в том, чтобы максимально повысить амплитуду состояния $|x_0\rangle|1\rangle$ и уменьшить амплитуды состояний $|x_0\rangle|0\rangle$ [2]. Представим последовательность реализации алгоритма Гровера:

1. Подготавливаем входной регистр $|x\rangle^n$, который будет содержать суперпозицию всех возможных значений входных наборов.

2. Вычисляем функцию $f(x)$.

3. Изменяем знак коэффициентов для тех значений x при которых функция $f(x)$ равна 1.

4. Далее необходимо увеличить амплитуду коэффициентов тех значений x, при которых функция $f(x)$ равна 1. Увеличение амплитуд возможно за счет применения операции, которая носит название «инверсия относительно среднего». После применения данной операции амплитуды входных значений, при которых функция равна 1 вырастут, а амплитуды входных значений, при которых функция равно 0 – уменьшаться.

5. Далее пункты 2–4 повторяются $\frac{\pi}{4}\sqrt{2^n}$ раз. Изменение знака коэффициентов выполняется следующим образом. Возьмем значение выходного регистра, равное 1, и применим преобразование Адамара:

$$|0\rangle^n|1\rangle \xrightarrow{H_{n+1}} = \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle\left(|0\rangle-|1\rangle\right).$$

Затем к полученному состоянию суперпозиции применяется преобразование U_f и, как и в алгоритме Дойча, это приводит к следующему результату:

$$\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle\left(|0\rangle-|1\rangle\right)\xrightarrow{U}\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle\left(|0\rangle-|1\rangle\right).$$

Отсюда можно заметить, что в полученном состоянии суперпозиции амплитуды значений, при которых функция равна 1, будут иметь отрицательное значение. Инверсия относительно среднего – это унитарное преобразование, которое воздействует на систему следующим образом:

$$\sum_i \alpha_i|x_i\rangle \to \sum_{x=0}^{2^n-1}(2A-\alpha_i)|x_i\rangle,$$

где A – среднее значение амплитуды. По сути амплитуда осуществляется путем переворачивания вероятностей относительно их среднего. Если число выше среднего, оно переворачивается и становится ниже среднего и наоборот. Преобразование инверсии относительно среднего описывается матрицей:

$$D=\begin{pmatrix} \frac{2}{N}-1 & \frac{2}{N} & \dots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N}-1 & \dots & \frac{2}{N} \\ \dots & \dots & \dots & \dots \\ \frac{2}{N} & \frac{2}{N} & \dots & \frac{2}{N}-1 \end{pmatrix},$$

где $T=2^n$.

Пример матрицы преобразования «инверсия относительно среднего» для $N=4$, т.е. для $n=2$, будет выглядеть следующим образом:

$$D=\begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}=\frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}.$$

Пример схемы, реализующей данное преобразование, представлен на рисунке 1.22 [14, 15].

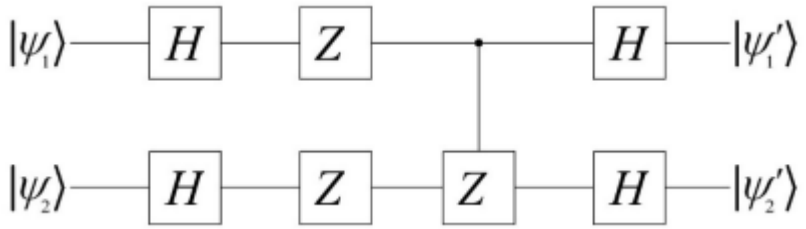


Рис. 1.22. Квантовая схема инверсии относительно среднего

Покажем, что данная схема реализует преобразование «инверсия относительно среднего:

$$H\otimes H=\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}=\frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix},$$

$$Z\otimes Z=\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}=\frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$(H\otimes H)(Z\otimes Z)=\frac{1}{2}\begin{pmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix},$$

$$CZ=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix},$$

$$(H\otimes H)(Z\otimes Z)CZ=\frac{1}{2}\begin{pmatrix}1&-1&-1&-1\\1&1&-1&1\\1&-1&1&1\\1&1&1&-1\end{pmatrix},$$

$$(H\otimes H)(Z\otimes Z)CZ(H\otimes H)=\frac{1}{4}\begin{pmatrix}-2&2&2&2\\2&-2&2&2\\2&2&-2&2\\2&2&2&-2\end{pmatrix}=\frac{1}{4}\begin{pmatrix}-1&1&1&1\\1&-1&1&1\\1&1&-1&1\\1&1&1&-1\end{pmatrix}.$$

В общем виде пример схемы алгоритма Гровера можно представить, как показано на рисунке 1.23 [14].

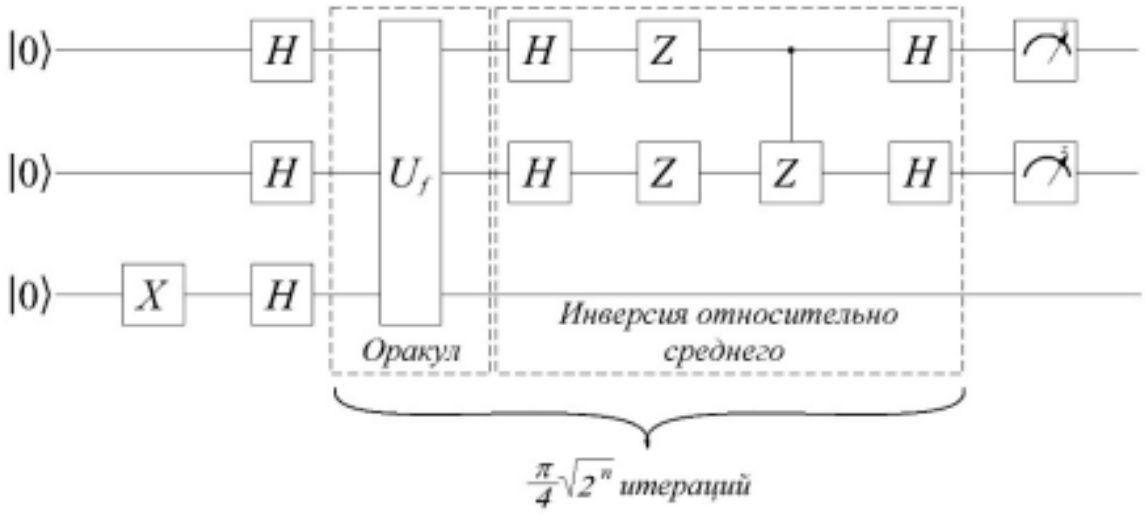


Рис. 1.23 Схема алгоритма Гровера

1.3.6. Квантовое преобразование Фурье

Квантовое преобразование Фурье (КПФ) – это аналог дискретного преобразования Фурье (ДПФ), которое применяется к вектору амплитуд квантовых состояний. Квантовое преобразование Фурье широко применяется во многих квантовых алгоритмах, в частности, в алгоритме Шора. Квантовое преобразование Фурье осуществляет преобразование квантового состояния следующим образом:

$$\sum_x f(x)|x\rangle=\sum_y F(y)|y\rangle\,.$$

При измерении состояния после преобразования Фурье с вероятностью $|F(y)|^2$ мы получим состояние $|y\rangle$. ДПФ осуществляет преобразование функции с периодам r в функцию, которая имеет

нулевые значения на всех частотах не кратных $\frac{1}{r}$. Применив квантовое преобразование Фурье к функции $f(x)$ с периодом r, мы получим функцию $F(y)$, которая будет равна нулю, кроме значений кратных $\frac{N}{r}$. То есть проведя измерение, мы получим результат кратный $\frac{N}{r}$ [35].

Квантовое преобразование Фурье определяется следующим образом:

$$|x\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i y x}{2^n}} |y\rangle .$$

В общем случае квантовая схема квантового преобразования Фурье представлена на рисунке 1.24.

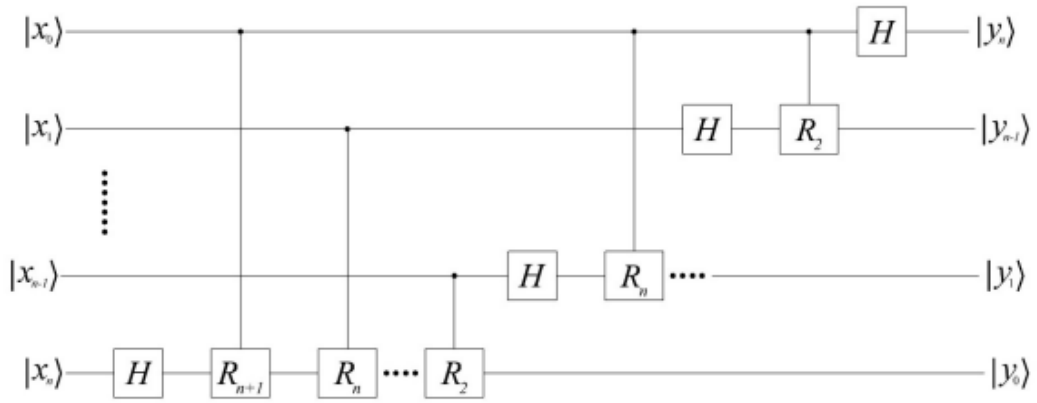


Рис. 1.24. Схема квантового преобразования Фурье

Следует отметить, что схема меняет порядок кубитов. В схеме, представленной на рисунке 1.24, используются унитарные операторы поворота R_k, которые можно определить как:

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{pmatrix}.$$

В общем случае выражение для n-й линии можно выражением [14]:

$$|x_n\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i \left(\frac{x_n}{2} + \ldots + \frac{x}{2^n} + \frac{x_0}{2^{n+1}} \right)} |1\rangle \right).$$

1.3.7. Алгоритм Шора

Алгоритм Шора является одним из наиболее распространенных алгоритмов и направлен на решение задачи факторизации. Факторизация – это разложение чисел на множители. Факторизация широко применяется в современной теории чисел и криптоанализа. Следует отметить, что эта задача весьма сложная, так как трудоемкость классических алгоритмов растет экспоненциально с увеличением размера факторизуемых чисел. Так, при фактризации числа N путем простого перебора необходимо будет перебрать все варианты от 2 до $\sqrt{N}$, а это весьма сложно, если мы имеем дело с очень большими числами [14, 15, 20].

Пример использования факторизации – это алгоритмы шифрования с открытым ключом (RSA, El Gamal и т.д.). В данном случае ключ представляет собой пару больших чисел, а взлом шифра, который заключается в нахождении по открытому ключу приватного и наоборот, требует решения задачи факторизации. Алгоритм Шора позволяет выполнить решение задачи факторизации за полиномиальное время. Это осуществляется за счет использования свойства квантового параллелизма и сведения задачи к поиску периода функции. Допустим нам необходимо разложить на множители некоторое число N. Изначально выберем произвольное число $\alpha < N$ и рассматриваем функцию $f_a(x) = a^x \bmod N$.

Функция $f_a(x)$ является периодической с периодом r. Период r является порядком числа $a : a^r = 1 \bmod N$ и $\forall r_1 < r \to a^{r_1} \neq 1 \bmod N$. Если число N простое, то период r будет равно $N-1$. Этот случай весьма простой и легко реализуется проверкой на простоту классическими методами.

В общем случае $f_a(x+r) = f_a(x)$. Если период r известен, то разложение на множители числа N легко можно определить классическими методами. В частности, если период r является четным числом, то из соотношения $a^r - 1 = 0 \bmod N$ можно записать:

$$\left(\alpha^{\frac{r}{2}} - 1\right)\left(\alpha^{\frac{r}{2}} + 1\right) = 0 \bmod N.$$

Так как $\left(\alpha^{\frac{r}{2}} - 1\right)\left(\alpha^{\frac{r}{2}} + 1\right)$ делится на N, то оба сомножителя имеют общие с N делители. Эти делители можно определить классическим алгоритмом Евклида по поиску наибольшего общего делителя. Если

же период r является нечетным или $\left(\alpha^{\frac{r}{2}}-1\right)\left(\alpha^{\frac{r}{2}}+1\right)$ вырождается в ноль, то следует выбрать другое число α. Для больших чисел N это случается редко.

Квантовый алгоритм Шора предназначен для быстрого поиска периода r. Для реализации алгоритма необходимо использовать квантовый компьютер с двумя квантовыми регистрами размера n. Причем размер должен быть таким, что $M = 2^n > N$, $M \approx N^2$. Алгоритм определения периода r будет следующим:

1. На первом этапе необходимо приготовить два входных регистра в состоянии $|0\rangle$.

2. Далее воздействием на каждый кубит входного регистра преобразованием Адамара:

$$|0\rangle^n |0\rangle^n \xrightarrow{H_n} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle^n .$$

3. Применим к обоим регистрам квантовую схему, реализующую функцию $f_a(x)$. Теперь регистры перейдут в состояние:

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle^n \xrightarrow{f_a(x)} \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |f_a(x)\rangle .$$

Таким образом, в выходном регистре будет суперпозиция всех возможных значений функции $f_a(x)$.

4. Далее производится измерение выходного регистра. В результате измерений мы получи

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |f_a(x)\rangle \xrightarrow{\text{измерение}} \frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} |k + jr\rangle |f_a(k)\rangle .$$

После измерения мы получим значение функции $f_a(k)$ при некотором случайном значении k, а выходной регистр перейдет в состояние $|f_a(k)\rangle$. Измеренное значение выходного регистра нас не интересует, но важно, что состояние входного регистра редуцируется до комбинации только тех состояний, которые совместимы с измеренным на выходе значением: $f_a(x) = f_a(k)$, $x = k$, $x = k + r$, $x = k + 2r$ и т.д.

5. Для извлечения периодичности в выходном регистре необходимо выполнить квантовое преобразование Фурье и измерение результата преобразования.

$$\frac{1}{\sqrt{M}}\sum_{i=0}^{M-1}\left|k+jr\right\rangle \xrightarrow{QFT} \frac{1}{\sqrt{MN}}\sum_{i=0}^{M-1}\sum_{y=0}^{2^n-1}e^{2\pi i\frac{ky}{N}}e^{2\pi i\frac{jry}{N}}\left|y\right\rangle = \frac{1}{\sqrt{MN}}\sum_{y=0}^{2^n-1}e^{2\pi i\frac{ky}{N}}\sum_{y=0}^{M-1}e^{2\pi i\frac{jry}{N}}\left|y\right\rangle .$$

При преобразовании Фурье период r будет преобразован в $\frac{M}{r}$ и измерение приведет к тому, что с высокой долей вероятности мы получим $\frac{jM}{r}$ для $j=1,2,...$ Далее, применив классический алгоритм разложения в непрерывную дробь (алгоритм Евклида), можно извлечь из полученного результата период r.

На рисунке 1.25 представлена структурная схема алгоритма Шора [1].

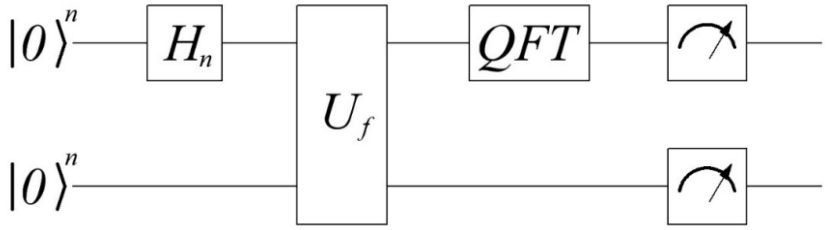


Рис. 1.25. Схема алгоритма Шора

Приведем классический пример реализации алгоритма Шора [1].

Пример 1.11: Допустим, нам необходимо выполнить факторизацию числа $N=pq=3\cdot 5=15$. Используя функцию Эйлера, можно определить функцию $f_a(x)$:

$$\Phi(N)=(q-1)(p-1)=(5-1)(3-1)=8\rightarrow a=7\rightarrow f_a(x)=7^x \bmod 15.$$

Функцию $f_a(x)$ можно записать в следующем виде:

$$f_a(x)=\left(7^8\right)^{x_3}\cdot\left(7^4\right)^{x_2}\cdot\left(7^2\right)^{x_1}\cdot\left(7^1\right)^{x_0} \bmod 15.$$

Поскольку $\left(7^8\right)^{x_3} \bmod 15=1$, $\left(7^4\right)^{x_2} \bmod 15=1$ и $\left(7^2\right)^{x_1} \bmod 15=(4)^{x_1} \bmod 15$, то можно переопределить функцию как:

$$f_a(x)=(4)^{x_1}\cdot(7)^{x_0} \bmod 15.$$

На рисунке 1.26 представлена квантовая схема, реализующая алгоритм Шора для данного примера.

На рисунке 1.26 блок под номером **1** реализует умножение регистра $|y\rangle$ на 7, а блок под номером 2 – умножение на 4. Далее выполняется квантовое преобразование Фурье (блок QFT) и измерение результата преобразования.

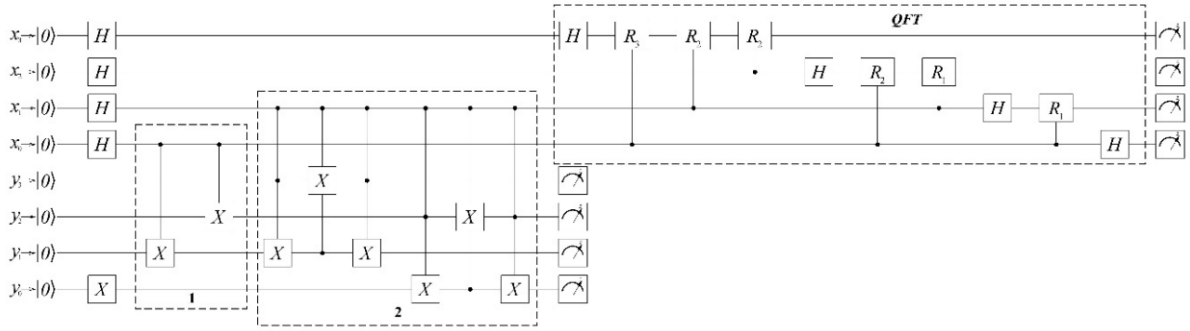


Рис. 1.26. Квантовая схема алгоритма Шора для $N=15$.

Выполним моделирование данной схемы в системе IBM Quantum Experience. Результаты серии измерений представлены на рисунке 1.27.

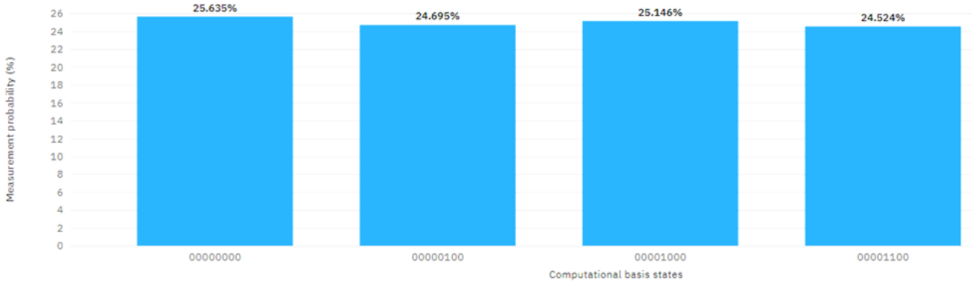


Рис. 1.27. Результаты симуляции алгоритма Шора для $N=15$.

Из рисунка 1.27 видно, что в результате измерений мы получили 0, 4, 8 и 12. Отбросив нулевой результат и вспомнив, что преобразование Фурье приведут к тому, что с высокой долей вероятности мы получим $\frac{jM}{r}$ для $j=1,2,...,$ можно вычислить период r:

$$\frac{jM}{r} \to \frac{j2^n}{r} \to \begin{cases} j=1 \to \dfrac{2^4}{r}=4 \to r=4 \\ j=2 \to \dfrac{2\cdot 2^4}{r}=8 \to r=4 \\ j=3 \to \dfrac{3\cdot 2^4}{r}=4 \to r=4 \end{cases}$$

Таким образом, мы определили период $r = 4$. Зная период, мы можем определить числа p и q:

$$НОД\left(7^{\frac{r}{2}} + 1{,}15\right) = НОД\left(50{,}15\right) = 5,$$

$$НОД\left(7^{\frac{r}{2}} - 1{,}15\right) = НОД\left(49{,}15\right) = 3.$$

1.4. Реализация алгоритма преобразования классического изображения в квантовое состояние

1.4.1. Процесс формирования набора кубитов

На рис. 1.28 представлен вычислительный процесс при моделировании запутанных квантовых вычислений в области квантовых алгоритмов, что подразумевает в первую очередь применение описанных в работе квантовых вычислений различного рода в работе квантовых алгоритмов и формирование набора кубитов для состояния управляющих сигналов нормализации в конкретный момент времени.

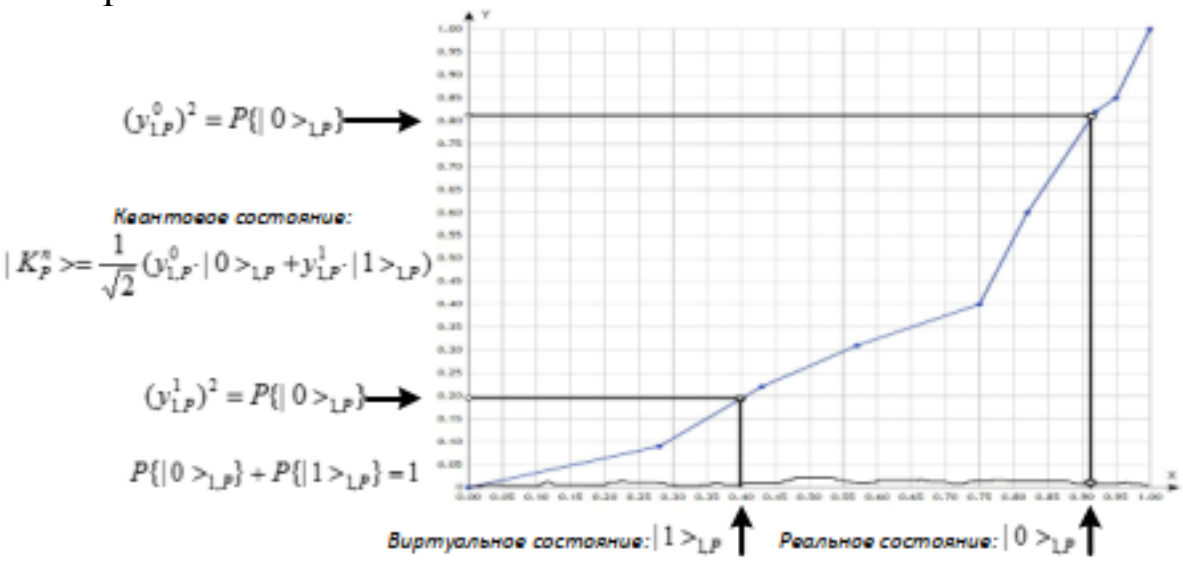


Рис. 1.28. Процесс формирования набора кубитов

Применяя тензорное произведение между преобразованиями Адамара, получим члены вида $K^n_P \otimes K^{n_2}_D$ и аналогичные комбинации коэффициентов усиления. Пример, описанный выше, свидетельствует о существовании шестнадцати вероятностных состояний, которые описывают вариации среди корреляций в соответствии с их типом и видом.

1.4.2. Выполнение преобразования классического изображения в состояние квантовой суперпозиции

Рассмотрим квантовый метод [18], направленный на представление и обработку цветной пиксельной фотографии или изображения. Предполагается преобразование каждого пикселя изображения $x(i, j)$ в некое квантовое состояние [19] $|q(i, j)>:|q(i, j)>=c_0|0>+c_1|>1$. Произведем кодирование цветовой палитры пиксельного набора в рамках комплексных амплитудных квантовых состояний [20]:

$$\delta: R^3 \to C_1^2, (x_1, x_2, x_3) \to \left(r_1 e^{i\phi_1}, r_1 e^{i\phi_2}\right), \quad (1.1)$$

где x_1, x_2, x_3 – значения системы цветов RGB, $r_1 := \sqrt{1-x_3^2}$, $r_2 := x_3$, $\phi_1 := \arcsin(2x_1 - 1)$, $\phi_2 := \arcsin(2x_2 - 1)$. Пусть $z_1 = r_1 e^{i\varphi_1}$, $z_2 = r_2 e^{i\varphi_2}$, $z_2 = r_2 e^{i\varphi_2}$, далее получим пиксельный набор цветов в форме $|q_1\rangle = z_1|0> + z_2|1>$. Следующим этапом является процесс кодирования [11] координат цветовой палитры пиксельного набора:

$$|k\rangle = |x\rangle|y\rangle = |x_{n-1}x_{n-2}...x_0\rangle|y_{n-1}y_{n-2}...y_0\rangle, x_i, y_i \in \{0,1\}, \quad (1.2)$$

где $|x\rangle$, $|y\rangle$ – квантовые состояния, кодирующие пиксельную координатную сетку. В зависимости от используемого преобразующего алгоритма выборка амплитудных [12] значений вероятности кодирующие пиксельные цвета фотографии, могут существенно различаться. Создадим суперпозицию вычислительного процесса. Пиксельная суперпозиция состояний квантовой системы фотографии или изображения на входе:

$$|I\rangle = \frac{1}{2^n} \sum_{k=0}^{2^{2n}-1} |q_k\rangle \otimes |k\rangle. \quad (1.3)$$

Рассмотренный метод трансляции классического изображения в квантовое состояние суперпозиции характеризует фотоизображение, которое состоит из пиксельного набора, в форме унифицированной суперпозиции с характеристиками всего набора пикселей. Вектор состояния и вероятностная амплитуда [13] сохраняются в системе отдельными значениями, поскольку речь идет о разработанной модели квантового вычислительного устройства [14] и алгоритма, входящего в состав модели, а также хранении пиксельного набора в вычислительном устройстве классического образца.

При получении на входе модели квантового вычислительного устройства и алгоритма серого изображения и трансляции его в бинарное состояние необходимым и достаточным является

предварительная обработка пиксельного набора средствами классических методов, средств и алгоритмов, направленных на распознавание объектов. Данные процесс повторяется многократно на нескольких этапах вычислительного процесса алгоритма. Область применения данного алгоритма – задачи сжатия исходного набора пикселей с помощью устранения информации избыточного характера. Преимуществом изображения бинарного природы – наименьшее количество требуемой памяти и времени процессора с точки зрения процесса обработки. Алгоритм требует выполнения ряда этапов:

Этап 1. Пусть имеем изображение размерности $M \times N$ с полутоновым набором пикселей. Произведем трансляцию всего набора пикселей поступающей на вход фотографии $x(i, j)$ в состояние $|q(i, j)\rangle$ квантовой системы. При интенсивности всего пиксельного набора, тогда вероятностные распределения $|c_0|^2$ и $|c_1|^2$ вычисляются с помощью определения сумм s_1 и s_2 так:

$$s_1 = \sum_{k=-1}^{1} \sum_{l=0}^{1} x(i-k, j-1) + x(i, j+1) + x(i, j+2), \quad (1.4)$$

$$s_2 = x(i-1, j-1) + x(i-1, j) + x(i, j-1). \quad (1.5)$$

Если $P = (s_1 + s_2)/5$, то $|c_0|^2 = f(P)$, $|c_1|^2 = 1 - f(P)$, где $f(P) = \frac{1}{1 + e^{\frac{P+a}{b}}}$. Квантовые состояния $|0\rangle, |1\rangle$ соотносятся с векторами $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$. В соответствии с этим кубиту $|q(i, j)\rangle$ сопоставляется вектор $\begin{pmatrix} 1 - f(P) \\ f(P) \end{pmatrix}$, таким образом, интенсивность пиксельного набора фотографии представляется в двумерном пространстве.

Этап 2. С помощью измерения кубита $|q(i, j)\rangle$ всего пиксельного набора фотографии, подающейся на вход, происходит формирование матрицы пиксельной системы [15]. При соответствии квантовых состояний $|0\rangle, |1\rangle$ показателям 0, 1 выходного пиксельного набора фотографии на выходе получим бинарный объект. На рис. 2 изображена разработанная модель квантового вычислительного устройства и результат выполнения квантового алгоритма преобразования набора пикселей.

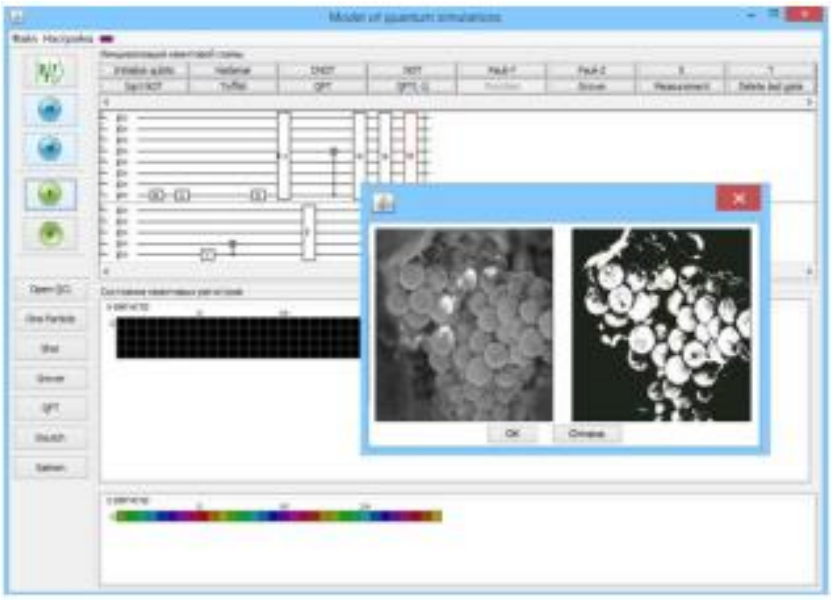


Рис. 1.29. Модель квантового вычислительного устройства

Процесс измерения заключается в разыгрывании числа случайного характера из промежутка $[0,1]$. При попадании значения в интервал $\left[0,|c_1|^2\right]$ выходным значением измерительного процесса является базисное состояние бра вектора 0, в противном случае (интервал $\left[|c_1|^2,1\right]$ – квантовое состояние бра вектора 1.

3. Вычислительный эксперимент. В настоящее время существует огромное количество методов и алгоритмов, направленных на распознавание объектов и лиц на требуемом изображении. Рассмотрим ряд классических методов и произведем сравнительный анализ с разработанным квантовым алгоритмом:

Метод главных компонент. Метод помогает в решении задач по уменьшению размера входных данных и потере наименьшего количества информации, позволяет выделять характерные признаки объекта и использовать их для реконструкции и восстановления. Данный метод основан на преобразовании Карунена–Лоэва.

Факторный анализ. Факторный анализ (ФА) базируется на гипотезе о том, что наблюдаемые переменные являются косвенными проявления относительно небольшого числа неких скрытых факторов. Цель ФА – получение модели изображения объекта, с помощью которой можно провести оценку близости тестового изображения к изображению объекта.

Метод опорных векторов. Метод опорных векторов позволяет построить классификатор, минимизирующий верхнюю оценку

ожидаемой ошибки классификации. Применение метода опорных векторов к задаче обнаружения лица заключается в поиске гиперплоскости в признаковом пространстве, отделяющий класс изображений лиц от изображений "не лиц".

Искусственные нейронные сети. ИНС – это математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма.

Скрытые марковские модели. Скрытые марковские модели являются одним из способов получения математической модели (описания свойств) некоторого наблюдаемого сигнала. Скрытые марковские модели относятся к классу стохастических моделей. Стохастические модели пытаются охарактеризовать только статистические свойства сигнала, не обладая информацией о его специфических свойствах.

В процессе вычислительного эксперимента было получено преимущество квантового алгоритма в сравнении с описанными выше классическими аналогами как по времени, так и по качеству распознавания необходимого лица или объекта на изображении. Для чистоты эксперимента было использовано единое изображение и весь вычислительный процесс проходил на одной вычислительной машине.

В табл. 1.3 показаны сравнительные характеристики и результаты всего эксперимента.

Таблица 1.3

Сравнительный анализ методов распознавания образов

	Метод главных компонент	**Факторный анализ**	**Метод опорных векторов**	**Искусственные нейронные сети**	**Квантовый алгоритм**	**Скрытые марковские модели**
Время выполнения процесса распознавания, с						
Качество распознавания, %						

Как видно из значений табл. 1.3 разработанный квантовый алгоритм в значительной мере выигрывает в скорости выполнения операция по распознаванию объектов на изображении. Что касается качества распознавания, то здесь выигрыш алгоритма квантовой природы минимален. Данный выигрыш обусловлен квантовыми преимуществами, а именно параллелизмом всего вычислительного процесса, позволяющим производить ряд вычислений по распознаванию объектов на изображении в асинхронном режиме в несколько потоков.

Стремление повысить вычислительную мощность компьютеров и обеспечить непревзойденные масштабы решаемых задач является одним из определяющих факторов развития суперкомпьютерных технологий.

Основными преимуществами использования вычислений квантового характера [16] в области определения объектов и изображений: ускорение вычислительного процесса с помощью квантовых компонентов, устойчивость при различном ракурсе объекта, его движении / статике, обеспечение криптографической помехоустойчивости. Важное значение придается разработке фундаментально новых физических принципов вычислений, где наиболее перспективным направлением является квантовый компьютинг.

1.5. Квантовая обработка изображений

Классическое представление изображений

Изображения в классических компьютерах определяются как матрицы чисел, представляющие дискретные значения цвета или интенсивности, присутствующие в каждом пикселе изображения. Каждое изображение рассматривается как входные данные, отображаемые различными способами, будь то массивы значений пикселей или многомерные графики, представляющие распределение интенсивностей пикселей. Изображения могут визуализироваться в цвете с наложением трех каналов (синий, зеленый и красный), в оттенках серого со значениями пикселей от 0 (черный) до 255 (белый) и в двоичном изображении, отображающем только значения черного или белого (0 или 1).

Представление квантового изображения (QIR)

Квантовая обработка изображений — один из наиболее привлекательных и многообещающих инструментов в наборе инструментов квантовых технологий. Представление изображения на квантовом компьютере в виде нормализованных состояний облегчает решение многих задач обработки изображений [22].

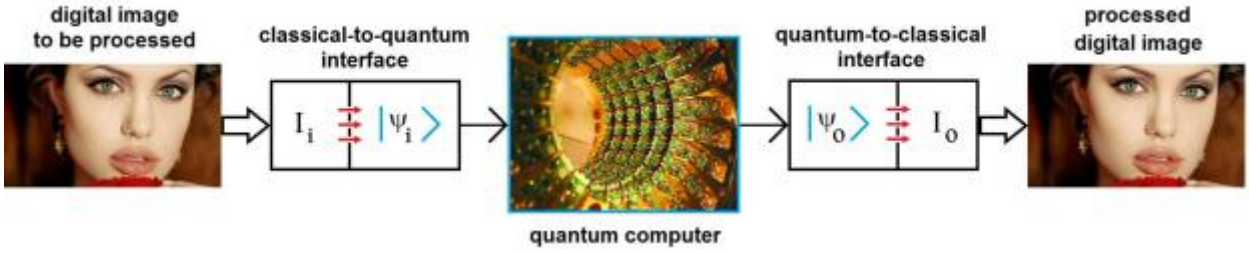


Рис. 1.30. Схема квантовой обработки изображений (QImP)

Для QIR были предложены следующие методы:

1) Решетка кубитов. В 2003 году Венегас-Андрака и Бозе предложили модель решетки кубитов [23] для отображения пространственной информации изображения с амплитудой одного кубита без использования квантовых свойств, поэтому требуется такое же количество кубитов, как пикселей. Это квантово-аналоговое представление классических изображений. Значение пикселя i^{th} строки и j^{th} столбца можно сохранить как:

$$\left| pixel_{i,j} \right\rangle = \left(\cos\left(\frac{\theta_{i,j}}{2} \right) |0\rangle + \sin\left(\frac{\theta_{i,j}}{2} \right) |1\rangle \right). \qquad (1.6)$$

2) Гибкое представление квантовых изображений (FRQI):

FRQI, предложенный Le et al. [24] сопоставляет значение шкалы серого каждого пикселя с амплитудой, а также фиксирует соответствующие положения на изображении и интегрирует их в квантовое состояние. Представление FRQI выражается как:

$$\left| I(\theta) \right\rangle = \frac{1}{2^n} \sum_{i=0}^{2n-1} \left(\sin(\theta_i) |0\rangle + \cos(\theta_i) |1\rangle \right) |i\rangle \ , \qquad (1.7)$$

где θ_i кодирует значение пикселя соответствующей позиции $|i\rangle$. Представление FRQI поддерживает нормализованное состояние, а пространство представления уменьшается экспоненциально по сравнению с классическим изображением из-за эффекта суперпозиции квантовых состояний. FRQCI и IFRQI — это два

улучшенных гибких представления квантовых изображений, о которых сообщили Ли и др. [25] и Хан и др. [26] соответственно.

3) Новое расширенное квантовое представление (NEQR):

Чжан и др. [27] сообщили о представлении, которое использует базовое состояние последовательности кубитов для хранения значения оттенков серого каждого пикселя вместо амплитуды вероятности, закодированной в кубите, как в FRQI. Изображения сохраняются путем переплетения информации о цвете, представленной $|f(y,x)\rangle$, с информацией о местоположении, представленной $|yx\rangle$. Представление NEQR для изображения $2^n \times 2^n$ выражается как:

$$|I\rangle = \frac{1}{2^n} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} |f(y,x)\rangle |yx\rangle, \qquad (1.8)$$

где $f(y,x)$ относится к интенсивности пикселя в точке $f(y,x)$. NEQR может выполнять сложные и сложные операции с цветом более удобно, чем FRQI. NEQR может добиться квадратичного ускорения подготовки квантовых изображений и точно извлекать цифровые изображения из квантовых изображений. Однако представление NEQR использует больше кубитов для кодирования квантового изображения. Исследователи также предложили несколько вариантов NEQR: улучшенный NEQR (INEQR) [28], обобщенную модель NEQR (GNEQR) [29] и CQIR [30].

4) Нормальное произвольное состояние квантовой суперпозиции (NAQSS):

NAQSS было предложено для решения многомерной обработки цветных изображений Ли и др. [31]. NAQSS — это $a(n+1)$ -кубитное квантовое представление, которое можно использовать для представления многомерных цветных изображений. N кубитов представляют цвета и координаты 2^n пикселей, а оставшийся 1 кубит представляет информацию о сегментации изображения для улучшения качества изображения.

NAQSS соответствует цвету и углу изображения один к одному, отображая информацию о цвете в определенное значение в интервале $[0, \pi/2]$. NAQSS можно представить как:

$$|I\rangle = \sum_{i=0}^{2^n-1} \theta_i |\upsilon_1\rangle |\upsilon_2\rangle ... |\upsilon_k\rangle \otimes |\chi_i\rangle, \qquad (1.9)$$

где

$$\left|\chi_i\right\rangle = \cos\gamma_i \left|0\right\rangle + \sin\gamma_i \left|1\right\rangle \tag{1.10}$$

представляет информацию о сегментации. NAQSS может повысить эффективность и точность сегментации изображений. Однако он не может точно измерить пиксели изображения.

5) Квантово-вероятностное кодирование изображений (QPIE). Представление QPIE Яо и др. [32] имеет непосредственное преимущество перед другими ранее обсуждавшимися методами кодирования, заключающееся в том, что оно позволяет кодировать прямоугольные изображения размером $r \times c$. Более того, количество кубитов, необходимых для кодирования изображения, дополнительно сокращается. Представление QPIE выражается как:

$$\left|I\right\rangle = \sum_{i=0}^{2^{2n}-1} c_i \left|i\right\rangle, n = \left\lceil \log_2 (rc) \right\rceil, \tag{1.11}$$

где

$$I' = \left(I_{1,1}, I_{2,1}, ..., I_{r,1}, I_{r,2}, ..., I_{r,c}\right)^T \tag{1.12}$$

и

$$c_i = \frac{I'(i)}{\left\|I'\right\|} \tag{1.13}$$

— нормализованное значение i^{th} элемента вектора.

Этот метод позволяет кодировать изображения произвольного размера $r \times c$. Однако это кодирование имеет проблему извлечения точного исходного изображения из его квантовой схемы кодирования. Это связано с тем, что значение пикселя сохраняется в амплитуде состояния, поэтому его приблизительное значение можно восстановить только путем нескольких измерений.

6) Кодирование данных на основе квантовой карты характеристик:

Классические данные кодируются в пространство квантовых состояний с использованием карты квантовых признаков. Выбор используемой карты объектов важен и зависит от набора данных, который необходимо классифицировать. Карта квантовых признаков использует классический вектор признаков для кодирования в квантовое состояние, что включает в себя применение унитарной операции к начальному состоянию.

$$\upsilon_{\Phi(X)} = \prod_d U_{\Phi(X)} H^{\otimes n} ,$$

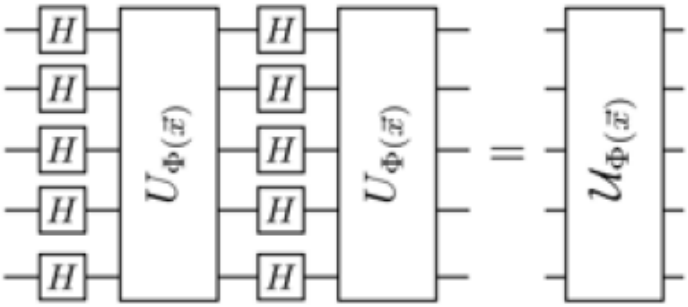
где

$$U_{\Phi(\vec{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{i \in S} Z_i \right).$$

Блоки запутанности, $U_{\Phi(x)} : P_i \in \{1, X, Y, Z\}$ обозначают матрицы Паули, S обозначают связь между различными кубитами/точками данных $S \in \left\{ \begin{matrix} n \\ k \end{matrix} \text{ комбинации, где } k \right\}$.

Карта признаков N-кубитов, сгенерированная унитарной функцией из работы Гавличека и др. [33].

Карта объектов содержит слои ворот Адамара с запутанными блоками, закодированными как:



Глубина цепи=2

Рис. 1.31. Карта признаков на N-кубитах, порожденная унитарной функцией из работы Гавличека и др. [33]

Здесь для кодирования данных используются карты ZZFeature Map, функции Паули и квантовое ядро SVM на основе карты Zfeature, как это представлено в работе Havlicek et al. [33]. При подготовке матриц квантового ядра для обучения и тестирования карта признаков сначала применяется к каждой паре точек данных обучения, а затем к точкам данных тестирования. На более позднем этапе обучение и проверка матриц квантового ядра используются для машины опорных векторов классификации.

Классическое обнаружение краев и углов

1) Обнаружение краев. Обнаружение краев — это метод обработки изображений, позволяющий найти границы объекта на данном изображении. Края — это часть изображения, которая представляет границу или форму объекта на изображении. Общий метод обнаружения края состоит в том, чтобы изучить изменения одного пикселя изображения в серой области и использовать

изменение края, соседнего первого или второго порядка, для обнаружения края.

Существуют различные методы обнаружения границ, и ниже приведены некоторые из наиболее часто используемых методов:

1) **Обнаружение границ Превитта** [34]: этот метод обычно используется для обнаружения границ на основе последовательного применения горизонтального и вертикального фильтра. Этот детектор границ на основе градиента оценивается в окрестности 3×3 для восьми направлений. Рассчитываются все восемь масок свертки.

2) **Обнаружение ребер Собеля** [35]: обнаружение ребер Собеля находит края с использованием аппроксимации Собеля к производной. Он предшествует краям в тех точках, где градиент самый высокий. Это один из наиболее часто испо льзуемых детекторов границ, который помогает снизить шум и обеспечивает дифференцирование, одновременно давая реакцию на края.

3) **Обнаружение ребер Лапласа** [36]: этот метод использует только один фильтр (также называемый ядром). За один проход обнаружение края Лапласа выполняет производные второго порядка и, следовательно, чувствительно к шуму. Лапласиан обычно используется для определения того, находится ли пиксель на темной или светлой стороне края.

4) **Обнаружение границ Канни** [37]: это наиболее часто используемый, высокоэффективный и сложный по сравнению со многими другими методами. Это лучший метод, который не нарушает впоследствии особенности краев изображения и применяет тенденцию нахождения краев и серьезное значение порога. Алгоритмические действия следующие:

А) Сверните изображение $f(r,c)$ с помощью функции Гаусса, чтобы получить гладкое изображение $f(r,c).f(r,c) = f(r,c) * G(r,c,6)$.

б) Примените первый оператор градиента разности для вычисления силы края, затем величина и направление края получаются, как и раньше.

в) Применить немаксимальное или критическое подавление к величине градиента.

d) Примените порог к изображению с немаксимальным подавлением.

2) Обнаружение углов. Обнаружение углов работает по принципу: если над изображением расположено небольшое окно, то это окно размещается в углу. Если это окно сдвинуть в любом направлении, произойдет значительное изменение интенсивности.

1) Обнаружение угла Моравца [38]: это один из самых ранних алгоритмов обнаружения угла, который определяет угол как точку с низким самоподобием. Алгоритм проверяет каждый пиксель изображения на предмет наличия угла, учитывая, насколько участок, введенный в пиксель, похож на близлежащие, в значительной степени перекрывающиеся участки.

2) Детектор углов Харриса [39]: этот метод был разработан для определения внутренних углов изображения и реализуется путем расчета градиента каждого пикселя. Он основан на функции локальной автокорреляции сигнала, которая измеряет локальные изменения сигнала с участками, сдвинутыми на небольшую величину в разных направлениях.

3) Детектор угла Форстнера [40]: этот метод находит точку, ближайшую ко всем касательным линиям угла в данном окне, и представляет собой решение методом наименьших квадратов для достижения приближенного решения. Алгоритм основан на том факте, что в идеальном углу касательные пересекаются в одной точке.

4) Угловой детектор SUSAN (наименьший однозначный сегмент, ассимилирующий ядро) [41]: этот метод основан на сравнении яркости и реализуется с помощью круговой маски. Если яркость каждого пикселя внутри маски сравнить с яркостью ядра этой маски, то можно определить область маски, имеющую ту же (или подобную) яркость, что и ядро. Площадь Сьюзен достигнет минимума, пока ядро находится в угловой точке.

5) Нечеткая система [42]: мера «угла» каждого пикселя изображения вычисляется с помощью нечетких правил (представленных в виде шаблонов), которые применяются к набору пикселей, принадлежащих прямоугольному окну. Возможная неопределенность, содержащаяся в окрестности окна, обрабатывается с использованием соответствующей базы правил (набора шаблонов). Нечеткая система проста в реализации и по-прежнему быстро выполняет вычисления по сравнению с некоторыми существующими

нечеткими методами. Кроме того, его можно легко расширить для обнаружения других функций.

Квантовые методы обнаружения краев

В известных классических алгоритмах выделения ребер при обработке типичного изображения размером $2^n \times 2^n$ с вычислительной сложностью менее $O\left(2^{2n}\right)$ такую задачу решить невозможно [43]. Напротив, квантовая обработка изображений использует характеристики квантовой механики суперпозиции и запутанности для одновременного расчета всех пикселей изображения, тем самым реализуя ускорение алгоритма. По сути, методы обнаружения квантовых границ представляют собой комбинацию методов QIR и классических методов края.

1) QSobel и другие методы, основанные на модели квантового изображения:

В начале 2014 г. Чжан и др. сообщил о новом алгоритме выделения границ квантового изображения под названием QSobel[44]. Он разработан на основе FRQI и знаменитого алгоритма выделения ребер Sobel. Фан и др. в 2019 году предложил метод обнаружения границ изображения, основанный на операторе Лапласа и Собеле [45]. В 2020 году Xu и др. предложил алгоритм квантовой обработки изображений с использованием выделения границ на основе оператора Кирша [43]. Алгоритм квантового обнаружения границ, основанный на улучшенном восьминаправленном операторе Собеля, был предложен Ма и др. в 2020 году [46]. Блок-схема показана на рисунке 4, которая более конкретно разделена на 4 этапа. Классическое цифровое изображение сначала квантуется в модель квантового изображения NEQR. Затем преобразования X-Shift и Y-Shift используются для получения сдвинутого набора изображений.

После этого градиент самого пикселя далее рассчитывается по маске Собеля с использованием одновременно набора смещенных изображений. Наконец, край исходного изображения извлекается с помощью пороговой операции U_T.

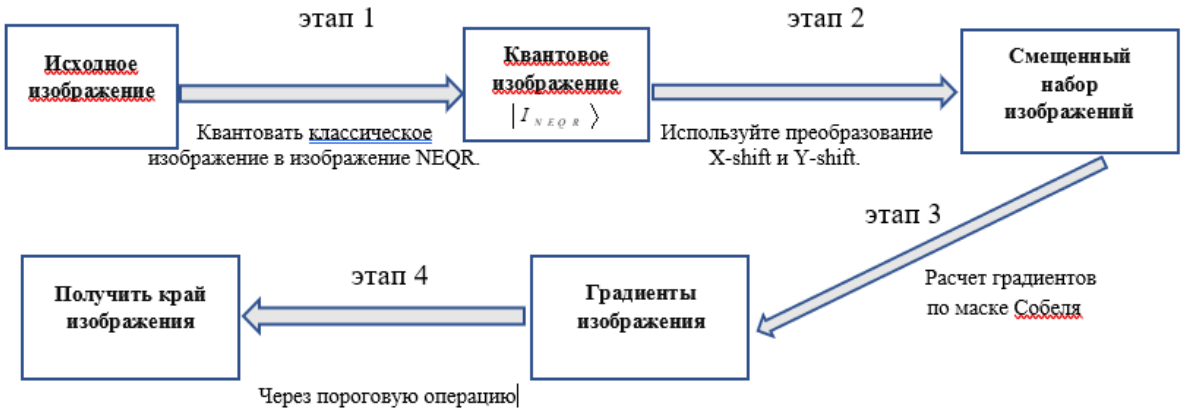


Рис. 1.32. Основной рабочий процесс выделения границ квантового изображения, изображение из работы Fan et al. [45]

2) Квантовое обнаружение границ Адамара (QHED). Другой вид алгоритма квантового обнаружения краев рассчитывает разницу между соседними пикселями в квантовом изображении с помощью преобразования Адамара. Этот метод называется квантовым обнаружением краев Адамара (QHED), о котором сообщает Яо и др., в 2017 году [47]. Этот алгоритм QHED кодирует значения пикселей изображения в амплитудах вероятности и положения пикселей в состояниях вычислительного базиса. В N-пиксельном изображении пиксели изображения нумеруются в виде $|b_1 b_2 ... b_{n-1} 0\rangle$, где $b_i = 0$ или 1.

Положения любой пары соседних пикселей в столбце изображения задаются двоичными последовательностями $|b_1 b_2 ... b_{n-1} 0\rangle$ и $|b_1 b_2 ... b_{n-1} 1\rangle$, а соответствующие значения интенсивности пикселей сохраняются как $c_{b_1 b_2 ... b_{n-1} 0}$ и $c_{b_1 b_2 ... b_{n-1} 1}$. Применяя преобразование Адамара к младшему биту квантового регистра произвольного размера. Тогда общая операция равна

$$I_{2^{n-1}} \otimes H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & & 0 & 0 \\ 1 & -1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & 1 & \ldots & 0 & 0 \\ 0 & 0 & 1 & -1 & & 0 & 0\ 0 \\ & & \ldots & & \ldots & & \ldots \\ 0 & 0 & 0 & 0 & \ldots & 1 & 1 \\ 0 & 0 & 0 & 0 & & 1 & -1 \end{bmatrix}$$

где $I_{2^{n-1}}$ — единичная матрица размером $2^{n-1} X\ 2^{n-1}$.

Применение этого унитарного метода к квантовому регистру, содержащему значения пикселей, закодированные с использованием представления QPIE, как

$$\left(I_{2^{n-1}} \otimes H\right).\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ \dots \\ c_{N-2} \\ c_{N-1} \end{bmatrix} \to \frac{1}{\sqrt{2}}\begin{bmatrix} c_0 + c_1 \\ c_0 - c_1 \\ c_2 + c_3 \\ c_2 - c_3 \\ \dots \\ c_{N-2} + c_{N-1} \\ c_{N-2} - c_{N-1} \end{bmatrix}.$$

Очевидно, что теперь у нас есть доступ к градиенту между интенсивностями соседних пикселей в виде $c_i - c_{i+1}$. Логика здесь такова: когда два пикселя принадлежат к одной и той же области, их значения интенсивности идентичны, и разница исчезает, в противном случае их разница не исчезает, что указывает на границу области. Этот процесс приводит к обнаружению горизонтальных границ между парами четных пикселей 0/1, 2/3 и так далее.

Для обнаружения горизонтальных границ между парами нечетных пикселей 1/2, 3/4 и т.д. мы можем выполнить перестановку амплитуды в квантовом регистре, чтобы преобразовать вектор амплитуды $(c_0, c_1, c_2, ..., c_{N-1})^T$ в $(c_0, c_1, c_2, ..., c_{N-1}, c_0)^T$, а затем применить H-элемент и измерить квантовый регистр, обусловленный тем, что LSB является $|1\rangle$.

Классификация изображений.

1) Машина опорных векторов (SVM) и квантовая SVM:

Классификация SVM по сути представляет собой метод бинарной (двухклассовой) классификации, и методы ядра могут обеспечить большую сложность классической SVM. Подобно машинам опорных векторов, алгоритм Quantum SVM применяется к задачам классификации, требующим карты признаков, неявно заданной ядром. Quantum SVM использует квантовую подпрограмму минимизации GroverOptim и обеспечивает сходимость к глобальному оптимуму для невыпуклой функции стоимости. Квантовая SVM и ядра могут эффективно использовать многомерное пространство и генерировать карты признаков и последующие границы решений, которые трудно сопоставить классическим функциям ядра.

Как упоминалось в работе Парка и др. [48], Quantum SVM потенциально может обеспечить более высокую производительность, если базовая граница является сложной и плохо улавливается традиционными классическими ядрами.

2) K-средние и квантовые K-средние: алгоритм K-средних принадлежит к семейству алгоритмов машинного обучения без учителя, которые группируют n наблюдений в K кластеры, обеспечивая минимизацию внутрикластерной дисперсии. Наиболее ресурсоёмкой операцией для алгоритма k-средних является вычисление расстояния между векторами. Квантовая версия алгоритма K-средних обеспечивает экспоненциальное ускорение для входных векторов очень большой размерности за счет того, что для загрузки N-мерных входных векторов с использованием амплитудного кодирования требуется только log N кубитов [49].

3) Convolutional Neural Network (CNN) и квантовая CNN: CNN продемонстрировала очень высокую производительность в области компьютерного зрения с преимуществом эффективного использования корреляционной информации данных. Однако CNN сложно эффективно обучать, если заданное измерение данных или модели становится слишком большим. Конг расширяет слой свертки и уровень объединения (основные особенности CNN) на квантовые системы в качестве модели квантовой CNN [50]. Схема свертки найдет скрытое состояние, применяя несколько кубитовых вентилей между соседними кубитами, а схема объединения уменьшит размер квантовой системы, наблюдая за долей кубитов или применяя двухкубитные вентили, такие как вентили CNOT. После повторения схемы свертки и схемы объединения до тех пор, пока размер системы не станет достаточно маленьким, полностью связная схема может предсказать результат классификации. Построение квантового сверточного слоя можно разделить на четыре этапа: (1) Процесс кодирования сохраняет пиксельные данные, соответствующие размеру фильтра, в кубитах; (2) Обучаемые квантовые схемы применяют фильтры, которые могут найти скрытое состояние из входного состояния; (3) Процесс декодирования получает новые классические данные путем измерения; (4) Повторите шаги (1)–(3), чтобы завершить новую карту объектов [51] .

Наборы данных

MNIST Dataset

Этот набор данных [52] представляет собой обширный набор из 60 000 рукописных цифр, каждое изображение имеет размер 28x28 пикселей. Наряду с обучающими данными он также предоставляет 10000 дополнительных наборов тестовых данных, состоящих из равномерно распределенных цифр от 0 до 9.

Fashion MNIST Dataset

Этот набор данных [53] состоит из 60 000 изображений одежды, 50 000 обучающих данных и 10 000 тестовых наборов данных с метками из 10 различных классов, каждое изображение имеет размер 28x28 пикселей.

Здесь для квантовой двоичной классификации небольшой набор данных используется для обучения и последующего тестирования трех одинаково распределенных классов данных одежды, обозначенных как:

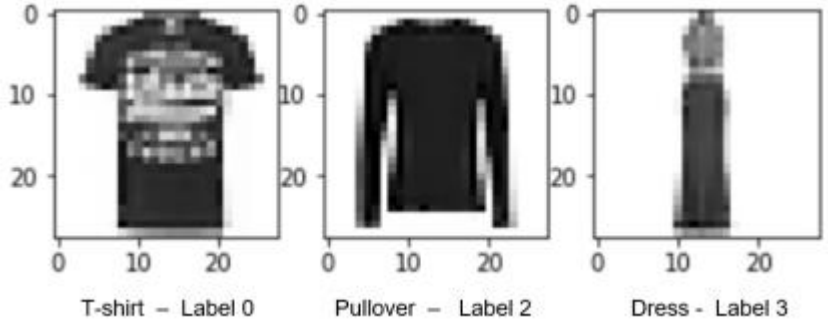


Рис. 1.33. Данные Fashion MNIST с метками 0, 2, 3.

Набор данных CIFAR-10 Этот набор данных (Изучение нескольких уровней функций из крошечных изображений) [34] состоит из 60 000 изображений, равномерно распределенных по 10 различным классам, каждое изображение имеет размер 32x32 пикселя, из которых 10 000 изображений используются в качестве набора тестовых данных. Этот набор данных разделен на 6 пакетов: 5 пакетов используются для обучения модели, а 1 пакет используется для оценки модели в качестве тестового набора данных.

Синтетические наборы данных. Современные квантовые устройства (NISQ) имеют ограниченное количество кубитов, что ограничивает исследование изображений больших размеров для создания синтетических наборов данных для оценки квантовых алгоритмов. Кроме того, мы также планируем применять методы уменьшения размерности, такие как анализ главных компонентов

(PCA), к общедоступным наборам данных перед их передачей через квантовые устройства.

Полученные результаты.

Классические методы

1) Обнаружение краев с помощью набора данных MNIST:

• Этап 1: Нормализация и сглаживание изображения. Пример этого этапа показан на рисунке 1.34.

• Этап 2: Получение информации, связанной с фронтом, с использованием фильтра Гаусса, Лапласа и Собеля, как показано на рисунке 1.35.

• Этап 3: угловой детектор Харриса для обнаружения границ цифр, как показано на рисунке 1.37.

• Этап 4: Преобразование в представление «мешком слов».

• Этап 5: Представление гистограммы.

• Этап 6: Классификация с использованием SVM.

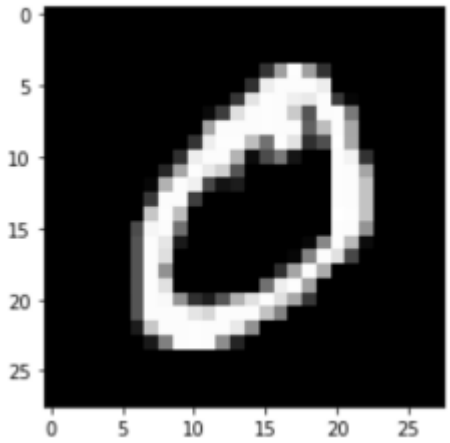


Рис. 1.34. Этап-1: Нормализация и сглаживание изображения

2) Модель глубокого обучения для набора данных MNIST: мы используем архитектуру с блоками Resnet [55], пакетной нормализацией и активацией ReLU для классификации набора данных MNIST. Архитектура обеспечивает точность 99,6% всего за 12 эпох.

Мы использовали средство поиска скорости обучения, предоставленное библиотекой FastAI [56], и обнаружили, что лучшая скорость обучения для нашей архитектуры составляет 0,05. Мы использовали стохастический градиентный спуск (SGD) в качестве оптимизатора и функцию потерь в качестве перекрестной энтропийной потери, поскольку MNIST представляет собой задачу классификации нескольких классов.

3) Модель глубокого обучения для набора данных CIFAR-10: здесь мы использовали функцию активации SOFTMAX с 10 единицами в выходных слоях. При составлении модели мы учли функцию потерь, используя разреженную категориально-кросс-энтропию, поскольку классы совершенно различны. Мы использовали Adam Optimizer для адаптации скорости обучения, поскольку он обновляет веса после каждой итерации.

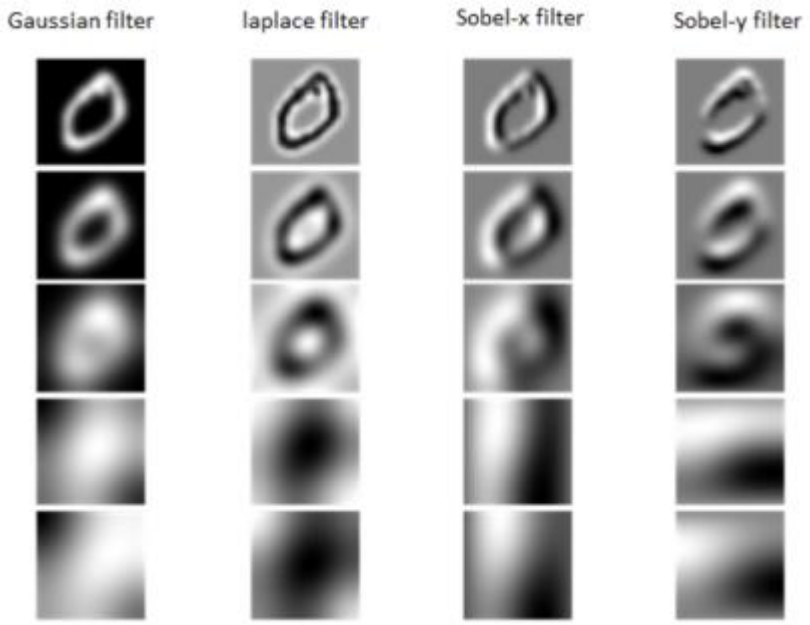


Рис. 1.35. Этап-2: Получение информации, связанной с фронтом, с использованием фильтров Гаусса, Лапласа и Собеля.

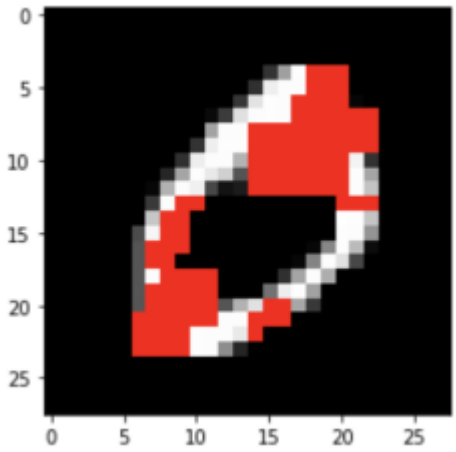


Рис. 1.36. Этап-3: угловой детектор Харриса для обнаружения краев цифр

```
(array([[575,   0,   0,   0,   0, 405,   0,   0,   0,   0],
        [  0, 698,   0,   0,   0, 437,   0,   0,   0,   0],
        [  0,   0, 645,   0,   0, 387,   0,   0,   0,   0],
        [  0,   0,   0, 607,   0, 403,   0,   0,   0,   0],
        [  0,   0,   0,   0, 622, 360,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0, 892,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0, 365, 593,   0,   0,   0],
        [  0,   0,   0,   0,   0, 418,   0, 610,   0,   0],
        [  0,   0,   0,   0,   0, 404,   0,   0, 570,   0],
        [  0,   0,   0,   0,   0, 383,   0,   0,   0, 626]]), 0.6438)
```

Рис. 1.37. Матрица ошибок и точность прогноза на тестовом наборе данных MNIST

После запуска 15 эпох окончательные результаты получены с точностью 77,5%.

Квантовые методы

1) Бинарная классификация квантового изображения набора данных MNIST:

Здесь лабораторная платформа IBM Quantum используется для двоичной классификации данных модных изображений MNIST [57]. В этом методе сначала классические данные были разделены на выборочные, проверочные и тестовые данные и метки, с последующей предварительной обработкой классического набора данных с размерностью признаков 5 с использованием стандартизации, анализа главных компонентов и нормализации. Для многоклассовой классификации мы использовали три бинарных классификатора на основе квантовых машин опорных векторов с подходом «один против остальных», классифицирующим конкретную метку, например, метку 0 как положительную (1), а остальное как отрицательную (0).

На втором этапе классические данные были закодированы в квантовое состояние с использованием различных карт квантовых признаков, после чего была подготовлена матрица квантового ядра с использованием симулятора квантового вектора состояния для набора данных для обучения и проверки. Наконец, точность матриц квантового ядра оценивалась на основе набора проверочных и тестовых данных с вероятностью предсказания конкретной метки для тестовых данных.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 128)               524416
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 591,274
Trainable params: 591,274
Non-trainable params: 0
_________________________________________________________________
```

Рис. 1.38. Краткое описание модели глубокого обучения (CNN)

```
[ ] test_loss, test_accuracy = cifar10_model.evaluate(x_test, y_test)

    313/313 [==============================] - 2s 6ms/step - loss: 0.7296 - sparse_categorical_accuracy: 0.7754

[ ] print("Test accuracy: {}".format(test_accuracy))

    Test accuracy: 0.7753999829292297
```

Рис. 1.39. Результаты модели глубокого обучения (CNN)

Наконец, результат был получен с использованием квантовых ядер на основе различных квантовых карт признаков для правильной идентификации и прогнозирования метки в тестовых данных на основе наибольшей вероятности, соответствующей конкретной метке. Классификация квантовых бинарных изображений была выполнена на основе различных карт признаков, т.е.

Результаты карт ZZFeature, Pauli и ZFeature сравнивались на предмет полученной точности классификации различных меток изображений. Квантовое ядро, основанное на карте свойств ZZ, включает в себя запутанные блоки в различной конфигурации, т.е. линейную, круговую и полную запутанность для кодирования классических данных. Это сравнение показывает точность прогнозирования в карте объектов ZZ с различной конфигурацией запутанности меток данных тестового изображения, что помогло

реализовать подходящую конфигурацию для 3-классовой классификации изображений с высокой точностью.

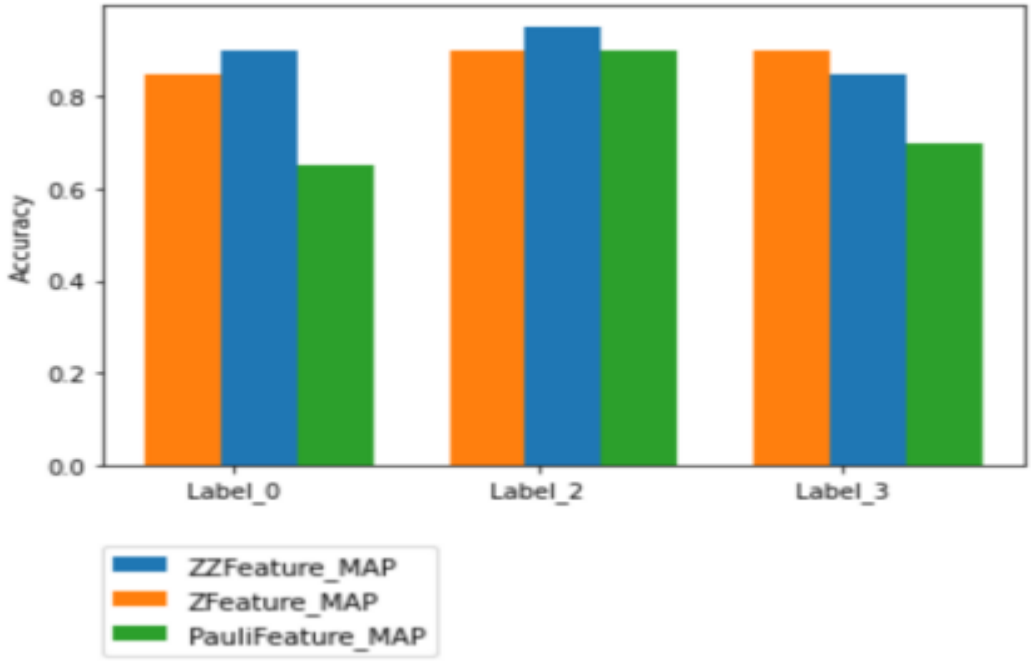


Рис. 1.40. Точность предсказания квантового ядра на основе различных карт признаков

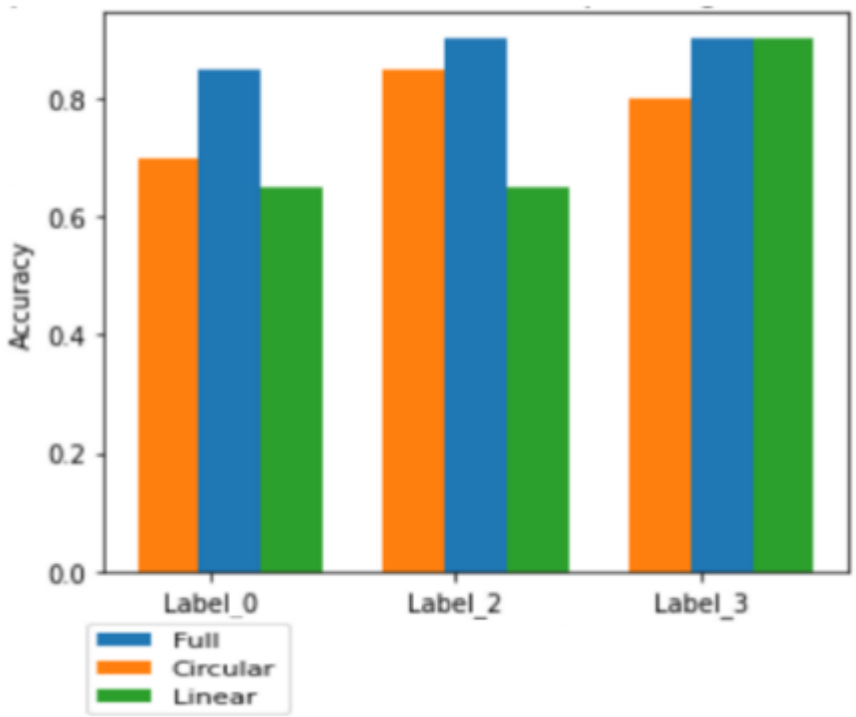


Рис. 1.41. Прогнозирование на основе различных конфигураций запутанности карты ZZFeature

2) Квантовая классификация изображений с использованием обнаружения краев:

Этот метод разделен на два этапа: первый этап включает в себя поиск краев изображения, а на втором этапе края затем используются для классификации изображения. Обнаружение края с помощью классических алгоритмов происходит намного медленнее по сравнению с обнаружением края на основе Адамара. Временная сложность классической операции составляет $O\left(2^n\right)$, однако квантовая система может выполнять расчет градиента между пикселями гораздо быстрее, используя операцию Адамара $O(mn.\log(mn))$. Следовательно, используя более высокую скорость работы, этот метод выполнял обнаружение границ с использованием квантовых схем и выполнял классификацию с использованием логистической регрессии, которая использовалась для получения точности модели.

Для представления изображения используется QPIE (кодирование изображения с квантовой вероятностью), где изображение использует $n = \log_2 N$ кубитов для представления изображения, где N — количество пикселей изображения.

Полученные результаты:

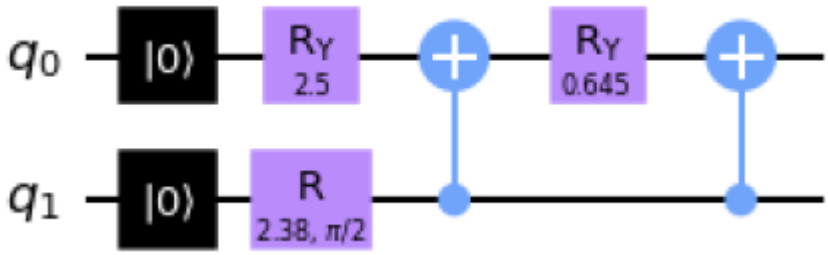


Рис. 1.42. Представление изображения QPIE

Алгоритм выполнил два сканирования для расчета градиента: при первом сканировании были обнаружены только горизонтальные края, а при втором сканировании были обнаружены вертикальные края. Эта операция эквивалентна операциям SobelX и SobelY в классических алгоритмах. Этот метод также обрабатывает большие изображения, разделяя их на более мелкие фрагменты, и отдельно применяет обнаружение краев.

В основном это делается из-за ограничения количества кубитов в реальных квантовых компьютерах, связи между кубитами, а также для уменьшения вероятности влияния шума на работу.

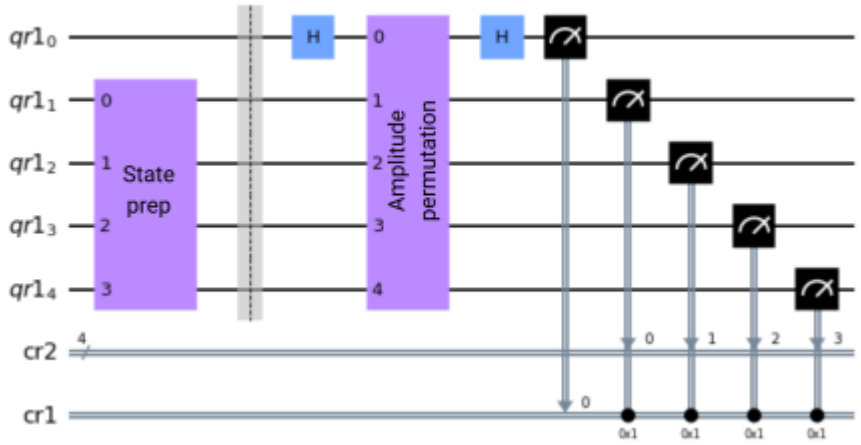


Рис. 1.43. Обнаружение квантового края Адамара с помощью симулятора

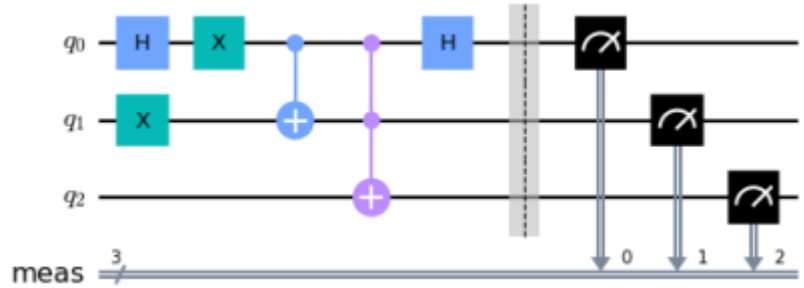


Рис. 1.44. Квантовое обнаружение краев Адамара при горизонтальном сканировании

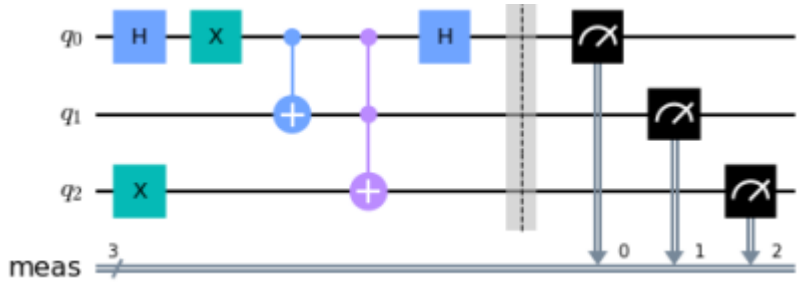


Рис. 1.45. Квантовое обнаружение края Адамара при вертикальном сканировании

Для набора данных MNIST изображение масштабируется до 32x32 пикселей, а затем делится на фрагменты размером 4x4 пикселя. Причина выбора 32 — обеспечить поддержку набора данных CIFAR, а также повысить качество обнаружения границ за счет увеличения масштаба. На этапе 2 метода классификация выполняется с

использованием модели логистической регрессии с моделью нескольких классов, включая член пересечения с регрессией Лассо в качестве штрафа. Набор данных Cifar более сложен по сравнению с набором данных MNIST, поэтому требуется больше времени для обнаружения границ. Для выполнения обучения объем был уменьшен за счет использования 100 выборок для обучения из набора обучающих данных, а тестирование проводилось с использованием 10 выборок тестового набора данных, в результате чего было получено 10 % точность. Снижение точности может быть связано с неправильным соотношением разделения изображения на маленькие выборки, которое использовалось для обнаружения краев, а также с меньшими выборками обучения.

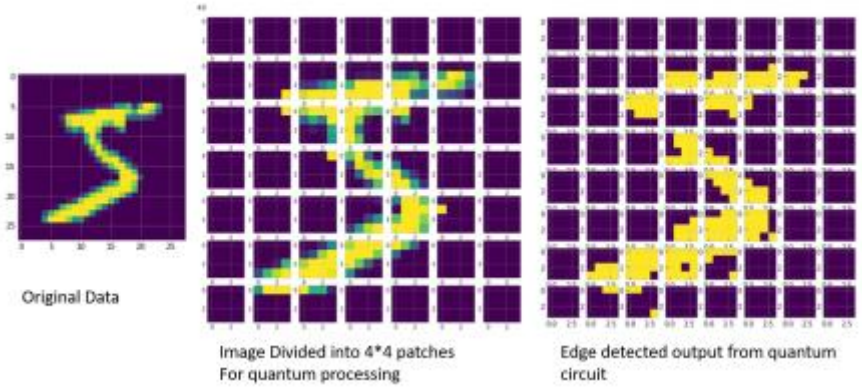


Рис. 1.46. Схема КГЭД с большими изображениями

3) Влияние шума. В эпоху шумных квантов среднего масштаба (NISQ) шум в квантовых компьютерах играет решающую роль. Мы изучаем влияние различных моделей шума для эксперимента Quantum Edge Detection на набор данных MNIST.

• Деполяризующий шум: определяется деполяризующий канал.

$$E(\rho)=(1-\lambda)\rho+\lambda Tr[\rho]\frac{I}{2^n} \tag{1.14}$$

С $0\le\lambda\le 4^n/(4^n-1)$, где λ — параметр деполяризующей ошибки, а n относится к количеству кубитов. Деполяризующий шум создает спонтанные переходы между собственными состояниями, вызывая внезапную смерть максимально запутанных состояний [38]. Мы анализируем влияние деполяризующего шума на однокубитные и двухкубитные вентили, а также комбинированное воздействие на оба типа вентилей.

На рисунках 1.47 и 1.48 показано влияние деполяризующего шума с различной вероятностью шума. Мы заметили, что шум на обоих воротах вместе оказывает большее влияние на производительность модели.

Шум Паули: Амплитуда и фаза переворота кубита на основе параметра шума, предоставленного в модели шума Паули. Мы анализируем влияние шума переворота битов, шума переворота фазы и совокупного эффекта обоих шумов на набор данных Quantum Edge Detection для MNIST. Рисунок 1.49 демонстрирует производительность модели с различными вероятностями шума Паули. Мы замечаем, что вероятность шума 0,001 оказывает большое влияние на производительность модели, снижая точность с 0,74 до 0,68. На рисунке 1.50 сравниваются результаты обнаружения границ для модели без шума, деполяризующего шума и шума Паули.

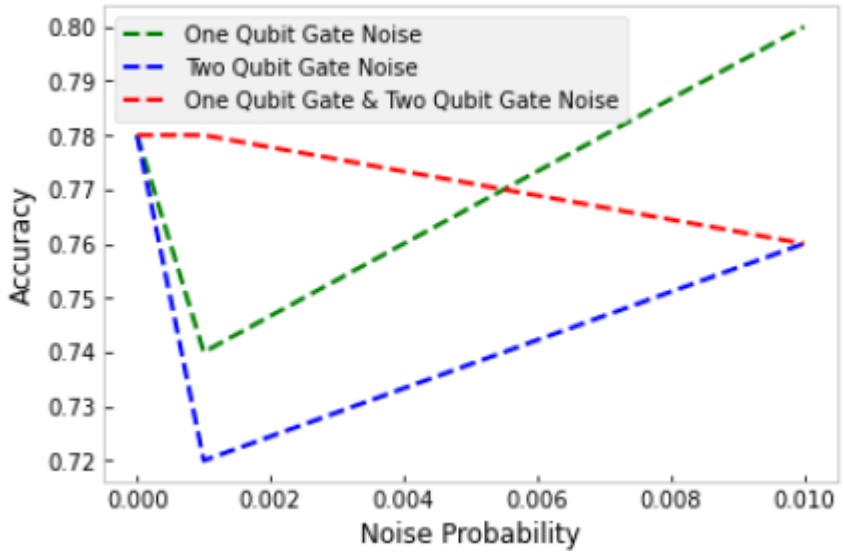


Рис. 1.47. Влияние деполяризующего шума с вероятностью шума 0,001, 0,01

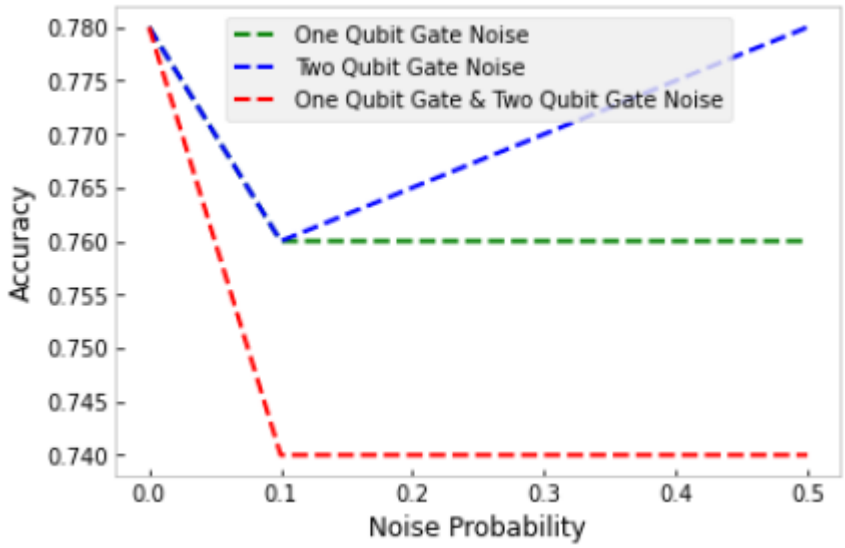


Рис. 1.48. Влияние деполяризующего шума с вероятностью шума 0,1, 0,5

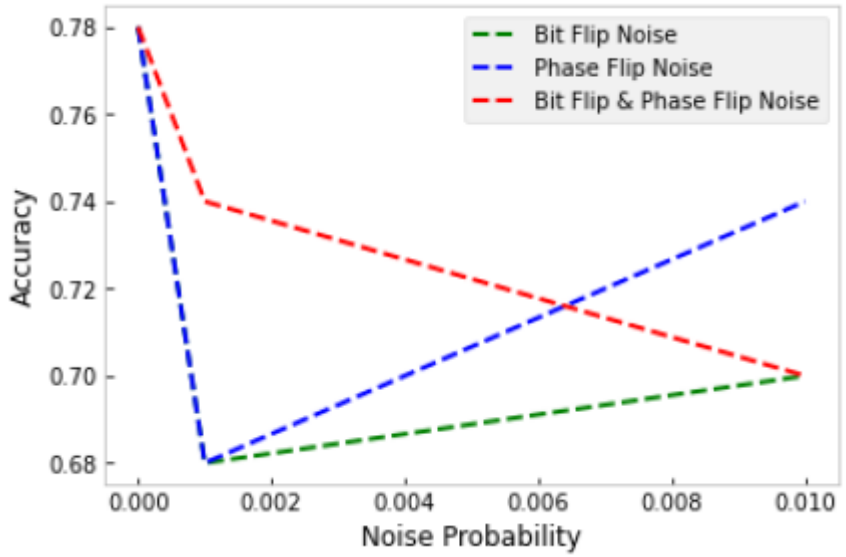


Рис. 1.49. Влияние деполяризующего шума с вероятностью шума 0,75

Рис. 1.50. Сравнение результатов обнаружения границ для бесшумной модели (3 верхних изображения), модели деполяризующего шума с вероятностью шума 0,5 для однокубитных и двухкубитных вентилей (3 средних изображения), модели шума Паули с вероятностью шума 0,01 как для переключения битов, так и для переключения фазы (3 нижних изображения)

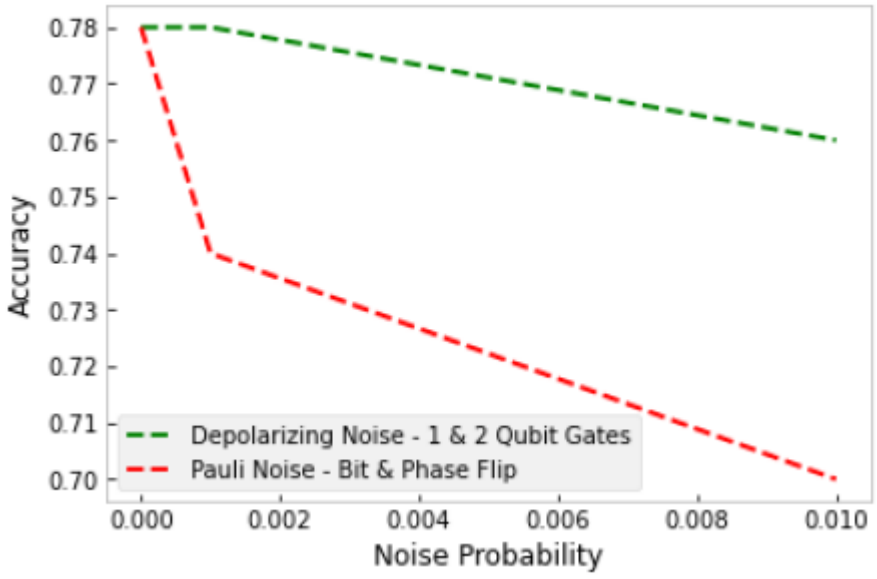


Рис. 1.51. Сравнение производительности модели с деполяризующим шумом и шумом Паули

При квантовой обработке изображений мы использовали различные методы кодирования изображений для обнаружения краев и классификации двоичных изображений. По сравнению с классическим процессом квантовый процесс обеспечивает преимущество во времени выполнения, пространственной сложности, т. е. количестве бит/кубитов, необходимых для кодирования изображения, но недостаточен с точки зрения глубины и ширины изображения. С помощью квантовых методов довольно сложно обрабатывать изображения большего размера, поскольку проектирование схем усложняется с увеличением количества требуемых кубитов, что приводит к добавлению шума с неточными результатами обнаружения границ и классификации. В этом проекте мы использовали квантовый вектор состояния на основе вентиля и симулятор qasm, которые работают медленнее, что, в свою очередь, увеличивает время обучения для более крупных наборов данных и требует разделения данных на более мелкие наборы и повторного объединения для получения более высокой точности для более крупных пикселей изображения и наборов данных. . При работе с реальным квантовым процессором производительность может быть выше, но он более подвержен шуму от устройств. В рамках реализации этих проектов наша команда работает над анализом влияния различных моделей шума на схему кодирования квантовых изображений и ее результатов при обработке изображений как с меньшими, так и с большими пикселями. Наша будущая работа будет включать в себя реализацию результатов квантовой обработки для проверки изображений с помощью исходных изображений в реальном времени для достижения большей точности.

Глава 2. НЕЧЕТКОЕ МНОЖЕСТВО И ОБРАБОТКА ИЗОБРАЖЕНИЯ

2.1. Нечеткая логика

Первые упоминания о нечеткой логике появились в 1965 году в работах Лофте Заде. Первоначально она разрабатывалась как средство моделирования неопределенности естественного языка, однако впоследствии круг задач, в которых нечеткая логика нашла применение, значительно расширился. В настоящее время она используется для управления линейными и нелинейными системами реального времени, при решении задач анализа данных, распознавания, исследования операций.

Нечеткая логика представляет собой надмножество классической булевой логики. Она расширяет возможности классической логики, позволяя оперировать не только значениями "ложь" и "истина", но и значениями в промежутке между ними. Не смотря на то, что нечеткая логика использует нечеткую информацию и основана на теории нечетких множеств, ее аппарат столь же строг и точен, как и классический.

Рассмотрим основные понятия определения из теории нечетких множеств.

Теория нечетких множеств. Одним из базовых понятий нечеткой логики является понятие нечеткого множества.

Пусть X - множество, состоящее из упорядоченных пар вещественных чисел. *Если* $x = (x_1, x_2)$ является элементом X, то этот факт записывается в символическом виде следующим образом:

$$x \in X .$$

В противном случае, т.е. когда x не является элементом X, используется запись

$$x \notin X .$$

Множество, не содержащее ни одного элемента, называется пустым множеством и обозначается символом ∅. Множество задается путем перечисления его содержимого в фигурных скобках: $\{\cdot\}$. Например, записывая выражение в форме $Z = \left\{ z \,\middle|\, z = n^2 + 2,\ n \in N \right\}$, мы имеем в виду, что множество z состоит из элементов z, которые строятся умножением на себя каждого элемента множества N. Один из способов использования множеств в обработке изображений заключается в рассмотрении элементов множества в виде координат

пикселей (упорядоченные пары целых чисел), представляющих объекты или другие интересующие признаки на изображении.

Если все элементы множества A являются также элементами другого множества B, то говорят, что A есть подмножество множества B, что символически обозначается так:

$$A \subseteq B.$$

Объединение двух множеств A и B, которое обозначается через

$$C = A \cup B,$$

есть по определению множество всех элементов, принадлежащих либо множеству A, либо множеству B, либо обоим множествам одновременно. Аналогично пересечение двух множеств A и B, которое обозначается через

$$D = A \cap B,$$

есть по определению множество всех элементов, принадлежащих одновременно обоим множествам A и B. Два множества A и B называются непересекающимися или взаимоисключающими, если у них нет общих элементов. В этом случае

$$A \cap B = \varnothing.$$

Множество всех элементов в данном приложении называют универсальным множеством. Например, при работе с множеством вещественных чисел универсальным множеством является числовая прямая, содержащая все вещественные числа. В цифровой обработке изображений в качестве универсального множества обычно определяется прямоугольник, содержащий все пиксели изображения.

Дополнение множества A есть множество элементов, не содержащихся в A:

$$\tilde{A} = \{z | z \notin A, z \in U\}.$$

Разность двух множеств A и B обозначается через $A \backslash B$ и определяется следующим образом:

$$C = A \setminus B = \{z | z \in A, z \notin B\}.$$

Видно, что это множество состоит из элементов A, которые не входят во множество B. Можно в качестве примера определить дополнение в терминах универсального множества и разности множеств: $\tilde{A} = U \setminus A$.

Следует отметить, что все элементы множества имеют одно и то же значение, поскольку в определениях теоретико-множественных операций не участвуют значения яркости (например, мы не

определили, какую яркость имеют элементы пересечения двух множеств). Однако можно говорить о принадлежности пикселей множеству на основе только координат, предполагая, что все элементы множеств имеют одинаковую яркость.

При работе с полутоновыми изображениями изложенные выше понятия неприменимы, потому что необходимо указать значения всех пикселей для результата операции над множествами. На самом деле, в случае полутонов операции объединения и пересечения обычно определяются как соответственно максимум и минимум для пары соответственных пикселей, а дополнение определяется как попарные разности между константой и яркостью каждого пикселя. Тот факт, что мы имеем дело с парами соответственных пикселей, указывает на поэлементный характер операций над полутоновыми множествами, как это определялось в подразделе 2.1.1. Рассмотрим пример по применению операции над множествами с учетом яркости изображений, который иллюстрирует теоретико-множественные операции над полутоновыми изображениями.

Пусть пиксели полутонового изображения представляются множеством *A*, элементами которого являются тройки вида *(x, y, z)*, где *x* и *y* – пространственные координаты, а *z* обозначает яркость в точке с указанными координатами. Можно определить дополнение *A* как множество $\tilde{A} = \{(x, y, L - z) | (x, y, z) \in A\}$, которое есть просто множество пикселей *A* с яркостями, полученными вычитанием их исходных значений из константы *L* ($L=2^k-1$, где *k* - число бит, используемых для представления яркости *z*). Предположим, что в качестве *A* выступает 8-битовое полутоновое изображение и требуется сформировать негатив *A* с помощью операций над множествами. Для этого построим множество $\tilde{A} = \{(x, y, 255 - z) | (x, y, z) \in A\}$.

Объединение двух полутоновых множеств *A* и *B* определяется как множество

$$A \cup B = \left\{ \max_z (a,b) \middle| a \in A, b \in B \right\},$$

где *a,b* – яркость изображений *A* и *B* соответственно в координатах *(x,y)*.

В результате применения операции объединения двух полутоновых изображений получается массив, сформированный из

максимальных значений яркости каждой пары соответственных пикселей.

2.1.1. Логические операции

При работе с двоичными изображениями можно говорить о множествах пикселей фона (со значениями 0) и переднего плана (со значениями 1). Если определить объекты как области, состоящие из пикселей переднего плана, то операции над множествами превращаются в операции над координатами объектов в бинарном изображении. Имея дело с бинарными изображениями, обычно говорят об объединении, пересечении и дополнении как о логических операциях OR (ИЛИ, логическое сложение), AND (И, логическое умножение) и NOT (НЕ, отрицание). Слово «логические» указывает на происхождение из формальной логики, в которой 1 и 0 означают соответственно истинное и ложное значения.

Рассмотрим две области, т.е. множества A и B, состоящие из пикселей переднего плана. Операция OR над этими двумя множествами дает множество элементов (пар координат), принадлежащих либо A, либо B, либо обоим множествам одновременно. Операция AND дает множество элементов, являющихся общими для A и B. Операция NOT в применении к множеству A дает множество элементов, не принадлежащих A. Поскольку речь идет об изображении, где A есть данное множество пикселей переднего плана, множество NOT(A) состоит из всех пикселей изображения, не входящих в A, т.е. всех пикселей фона и, возможно, других пикселей переднего плана. Эту операцию можно рассматривать как установку значений всех элементов A в 0 (черное), а элементов вне A в 1 (белое).

Известно, что для методов, основанных на логических операциях, достаточно рассмотреть только три логические операции AND, OR и NOT, поскольку они образуют функционально полный класс.

Рассмотренные выше теоретико-множественные и логические действия оперируют четкими множествами, т.е. обрабатываемый элемент либо принадлежит, либо не принадлежит множеству. В некоторых приложениях это сильно ограничивает возможности применения методов обработки, основанных на теоретическо-множественных и логических операциях.

Известно [62-64], что множество рассматривается как некоторая совокупность объектов (элементов), а теория множеств состоит из набора приемов, которые имеют отношение к операциям над и между множествами. Несомненно, что на базе теории множеств и математической логики построены аксиоматические основы классической математики. Одним из основных понятий в теории множеств является понятие принадлежности элемента к множеству.

Обычно при решении ряда задач используются так называемые «четкие» множества. Принадлежность их элементов может быть только истинной или ложной (истина обычно обозначается как 1, а ложь - как 0) (рис. 2.1). Например, пусть X означает множество всех представителей. Требуется определить подмножество A в Z, называемое «множеством молодых преподавателей».

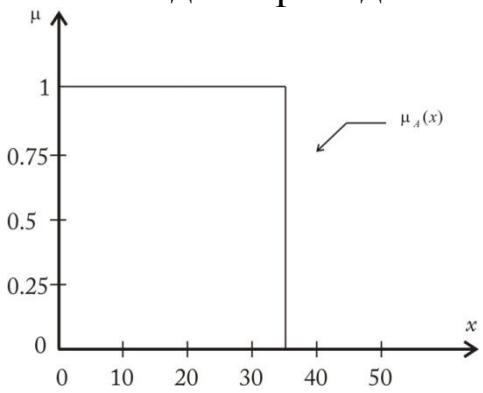


Рис. 2.1. Функция принадлежности, используемая при порождении четкого множества: μ - степень принадлежности; x - возраст; A - множество

Для определения этого подмножества необходимо задать функцию принадлежности элемента A. Заданная функция принадлежности определяет элементы A. При этом каждому элементу x из X присваивается определенное значение (точки 1 или 0). Известно, что в классической математике имеют дело с булевой логикой. Поэтому функция принадлежности задается просто порогом, определяемым в некоторой области; преподаватель, возраст которого ниже или совпадает со значением порога, признается молодым, а выше – не молодым. Этот подход, иллюстрируется рис. 2.2. На этом примере использован возрастной порог в 35 лет. На этом рисунке рассмотренная функция принадлежности обозначена через $\mu_A(z)$.

Необходимо отметить, что функции принадлежности в классической математике называются характеристическими функциями.

Нетрудно заметить некоторые трудности, которые возникают с этим определением: преподаватель возрастом в 35 лет признается молодым, а преподаватель в 30 лет и 1 минута уже не относится к множеству молодых специалистов. Данное обстоятельство является основной проблемой в работе с четкими множествами, которая ограничивает использование классической теории множеств во многих практических приложениях. Для определения функции принадлежности требуется большая гибкость, использование которой поможет описать постепенный переход от молодого к не молодому. Один из вариантов этой функции приведен на рис. 2.2. Ключевым свойством функции принадлежности должна быть его многозначность, которая обеспечивает непрерывный переход между понятиями «молодой специалист» и «не молодой специалист». Это позволяет представить степень «молодости». После этого можно ввести такие понятия как: «молодой специалист» (верхняя плоская часть графика), «сравнительно молодой специалист» (верхняя часть склона), «не слишком молодой специалист» (нижняя часть склона) и так далее.

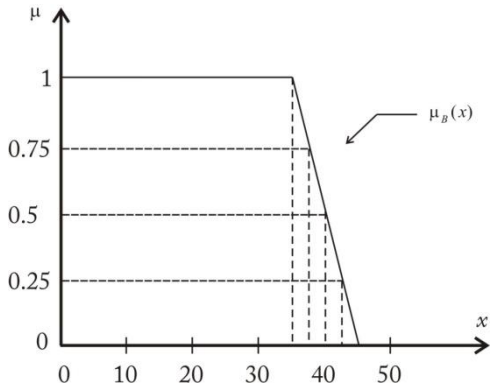


Рис. 2.2. Функция принадлежности, используемая при порождении нечетного множества: μ - степень принадлежности; x - возраст; A - нечеткое множество

Отметим, что уменьшение крутизны спада функции принадлежности на рис. 2.2 дает больше неопределенности в том, что необходимо понимать под «молодым специалистом». Эти типы неопределенности (нечеткости) в определениях лучше согласуются с тем, что имеется в виду, когда люди говорят неточно о возрасте. Таким образом, функцию принадлежности многозначности можно интерпретировать как основу нечеткой логики, а множества, возникающие в результате использования этой логики, могут быть

рассмотрены как нечеткие множества. Эти идеи формализованы и развиты в работах [65-70].

Очень часто при описании какого-либо объекта возникают такие ситуации, когда мы не можем достаточно точно определить численные значения его параметров, а можем лишь задать их приблизительные значения. К примеру, глубина реки «больше 80» метров, Петрову «около 30» лет и т.д. Это все есть примеры так называемых нечетких чисел.

Нечеткое число **A** – это нечеткое множество действительных чисел с нормальной, выпуклой и непрерывной функцией принадлежности ограниченного опорного множества.

В зависимости от того, как мы определяем нечеткое число s $\in$ **S** – «s приблизительно равен a» или «s приблизительно лежит в интервале [a;b]», – оно может быть представлено либо как триангулярное, либо как трапецеидальное.

Нечеткое множество **A** называется триангулярным нечетким числом с вершиной (или центром) a, шириной слева $\alpha>0$ и шириной справа $\beta>0$, если его функция принадлежности имеет вид

$$A(t)=\begin{cases}1-(a-t)/\alpha, a-\alpha\le t\le a\\ 1-(t-a)/\beta, a\le t\le a+\beta\\ 0, t<a-\alpha, t>a+\beta\end{cases}, \qquad (2.1)$$

и мы используем обозначение **A** = (a, α, β). Опорным множеством множества **A** является (a – α, a + β). Данное нечеткое число с центром a можно понимать как нечеткую величину «s приблизительно равно a».

Нечеткое множество **A** называется трапецеидальным нечетким числом с интервалом допуска [a,b], шириной слева $\alpha>0$ и шириной справа $\beta>0$, если его функция принадлежности имеет вид

$$A(t)=\begin{cases}1-(a-t)/\alpha, a-\alpha\le t\le a\\ 1,\ a\le t\le b\\ 1-(t-b)/\beta, a\le t\le b+\beta\\ 0,\ t<a-\alpha, t>b+\beta\end{cases}, \qquad (2.2)$$

и мы используем обозначение **A** = (a, b, α, β). Опорным множеством множества **A** является (a – α, b + β). Данное нечеткое число можно понимать как нечеткую величину «s приблизительно находится в интервале [a,b]».

Нечеткие системы управления. Нечеткие системы управления (нечеткие контроллеры, НСУ) являются наиболее важным

приложением теории нечетких множеств. Функционирование нечетких логических контроллеров отличается от работы обычных контроллеров. Это отличие состоит в том, что обычные контроллеры основаны на аналитическом выражении, описывающем объект управления, в то время как нечеткие контроллеры используют знания экспертов. Эти знания могут быть выражены естественным образом с помощью так называемых лингвистических переменных.

Лингвистические переменные могут рассматриваться либо как переменные, значения которых являются нечеткими числами, либо как переменные, значения которых определяются в лингвистических термах [61].

Лингвистические переменные характеризуются пятеркой (x, T(x), U, G, M), в которой x – имя переменной, T(x) – множество термов x, т.е. множество имен лингвистических величин x, каждое значение которой есть нечеткое число, определенное на U, G – синтаксические правила для выработки имен величин x, M – семантические правила для связывания каждой величины с ее смыслом.

Предназначение НСУ состоит в том, чтобы следить за значениями переменных состояния управляемой системы и получать величины переменных управления путем определенных связей, которые представляют собой БП системы.

Нечеткие логические системы управления обычно состоят из четырех основных частей: Интерфейс фаззификации (введение нечеткости на четких входных данных), База нечетких правил, Машина нечеткого вывода и Интерфейс дефаззификации (исключение нечеткости для обеспечения четкого выхода) [61].

Интерфейс фаззификации – оператор фаззификации переводит четкие данные в нечеткие множества.

База нечетких правил – содержит информацию о связях нечетких входных и выходных значений.

Машина нечеткого вывода – механизм сопоставления по БП нечетких значений входных параметров нечетким значениям выходных.

Интерфейс дефаззификации – оператор дефаззификации переводит нечеткие множества в четкие значения.

В общем виде принцип работу НСУ представлен на рисунке 2.3.

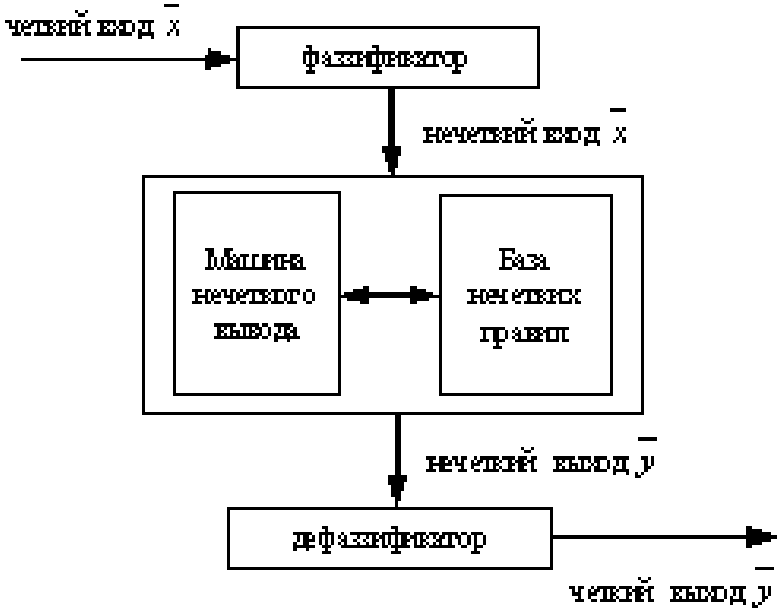


Рис. 2.3. Структурная схема нечеткого логического контроллера

2.1.2. Подавление шума упорядочивающими фильтрами

Упорядочивающие фильтры, основанные на порядковых статистиках, представляют собой пространственные фильтры, вычисление отклика которых требует предварительного ранжирования, т.е. упорядочивания значений пикселей, заключенных внутри обрабатываемой фильтром области S_{xy} изображения. Отклик фильтра определяется по результатам упорядочивания и не может быть представлен в виде свертки.

Наиболее известным из фильтров, основанных на порядковых статистиках, является медианный фильтр. Действие этого фильтра, как следует из его названия, состоит в присвоении пикселю восстановленного изображения с координатами *(x,y)* значения медианы упорядоченного множества значений яркости из окрестности S_{xy} обрабатываемого изображения [64]:

$$z'(x,y) = \underset{(s,t)\in S_{xy}}{med}\{z(s,t)\}. \qquad (2.3)$$

Широкая популярность медианных фильтров обусловлена тем, что они прекрасно приспособлены для подавления некоторых видов

случайных шумов и при этом приводят к меньшему размыванию контуров по сравнению с линейными сглаживающими фильтрами того же размера. Медианные фильтры эффективны при наличии как биполярного, так и униполярного импульсного шума, но мало эффективны при устранении гауссова шума (рис. 2.4). Существенным недостатком медианных фильтров является подавление и искажение мелких объектов, например, регулярных структур, показанных на рис. 2.4.

Для устранения данного вида шума («соль и перец») медианная фильтрация является одним из наиболее простых и эффективных методов. Размер локальной апертуры влияет на результат обработки, но не в такой степени, как в других рассмотренных нами методах. Одно из основных преимуществ медианной фильтрации состоит в том, что она приводит к устранению импульсных выбросов, не размывая границ объектов.

Хотя медианные фильтры принадлежат к числу наиболее часто используемых в обработке изображений фильтров, основанных на порядковых статистиках, это отнюдь не единственный пример таких фильтров. Медиана (2.3) представляет собой середину упорядоченного набора чисел, однако использование иных характеристик этого набора предоставляет много других возможностей. Например, использование крайних значений ранжированного набора чисел приводит к фильтрам максимума и минимума, основанным на выборе максимального или минимального значения яркости из окрестности S_{xy} соответственно [63].

Фильтр максимума полезен при обнаружении наиболее ярких точек на изображении. Кроме того, поскольку униполярный «черный» импульсный шум принимает минимальные значения, применение этого фильтра приводит к уменьшению такого шума, так как в процессе фильтрации из окрестности S_{xy} выбирается максимальное значение:

$$z'(x,y) = \max_{(s,t)\in S_{xy}} \{z(s,t)\}.$$

Рис. 2.4. Медиана при размере локального окна:
а - 3×3; *б* - 9×9

Соответственно, фильтр минимума полезен при обнаружении наиболее темных точек на изображении и его применение приводит к уменьшению униполярного «белого» импульсного шума:

$$z'(x,y) = \min_{(s,t)\in S_{xy}} \{z(s,t)\}.$$

На рис. 2.5, 2.6 приведены результаты обработки изображений с помощью фильтра типа максимум (рис. 2.5) и минимум (рис. 2.6) по окрестности. Конечно, результаты такой обработки существенно зависят от размеров локальной апертуры.

Последовательное применение максимального и минимального фильтров одинаковым размером апертуры позволяет эффективно удалять дефекты (пятна, разрывы линии контура) на бинарных изображениях без изменения формы и размеров фрагментов, несущих полезную информацию, как это показано на рис. 2.5,2.6. Максимальный размер удаляемых дефектов равен размеру окрестности S_{xy}.

а) *б*)

Рис. 2.5. Максимум по окрестности при размере локального окна: *а* - 3×3; *б* - 9×9

Рис. 2.6. Минимум по окрестности при размере локального окна: *а* - 3×3; *б* - 9×9

Фильтр срединной точки, или Чебышевский, заключается в вычислении среднего между максимальным и минимальным значениями в окрестности S_{xy} [63]:

$$z'(x,y)=\frac{1}{2}\left[\max_{(s,t)\in S_{xy}}\{z(s,t)\}+\max_{(s,t)\in S_{xy}}\{z(s,t)\}\right].$$

Обработка изображений данным методом показывает его неэффективность к устранению шума типа «соль и перец». Также из представленных на рис. 2.7 результатов следует вывод, что работа

фильтра типа срединная точка зависит от размеров локальной окрестности.

а) *б*)

Рис. 2.7. Срединная точка при размере локального окна: *а* - 3×3; *б* - 9×9

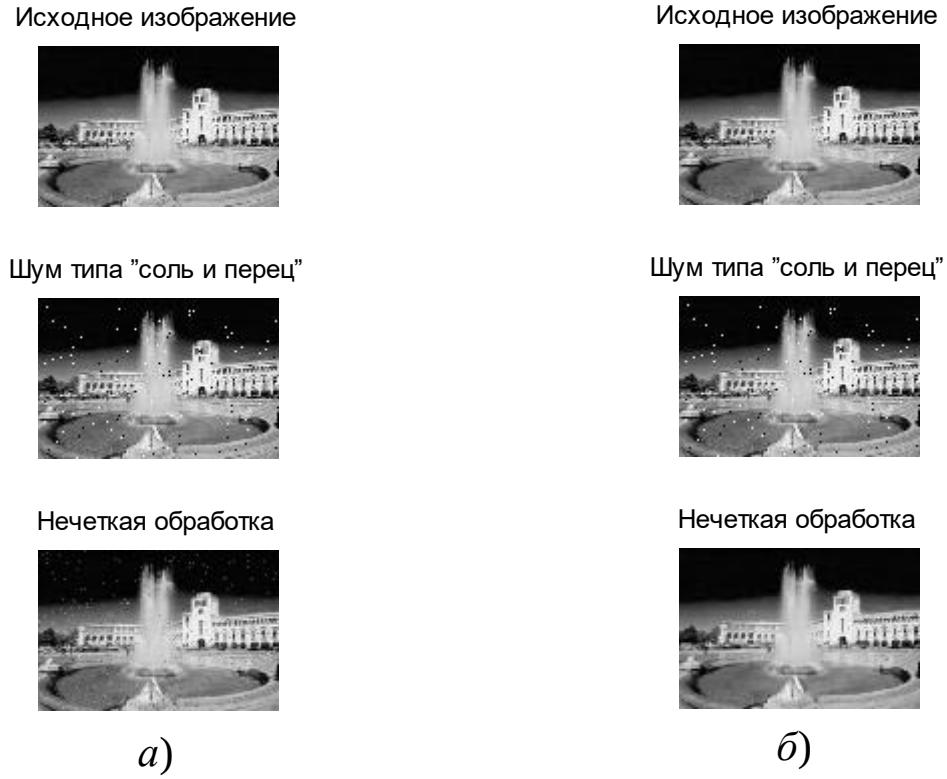
2.1.3. Нечеткая обработка изображения

Рассмотрим нечеткое множество $\grave{A}=\{(z,\mu_A(z))\}$, где Z – дискретное множество элементов $\{z_1,...,z_n\}$.

При рассмотрении дискретных нечетких множеств и случайных событий следует учитывать, что функция принадлежности задается только на дискретном множестве элементов из Z или R^n. Приведем выражения для нечеткой обработки изображения:

$$z'(x,y)=\frac{\sum_{i=1}^{n}\sum_{(s,t)\in S_{xy}}\mu(z_i)z_i(s,t)}{\sum_{i=1}^{n}\mu(z_i)}.$$

На рис. 2.8 показана нечеткая обработка изображения при размере локального окна 3×3 и 9×9.



а) *б*)

Рис. 2.8. Нечеткая обработка изображения при размере локального окна: *а* -3×3; *б* - 9×9

2.2. Предварительная обработка изображений с использованием аппарата нечетких множеств

2.2.1. Архитектура системы обработки изображений, основанной на аппарате нечетких множеств

Известно, что теория нечетких множеств оперирует качественными понятиями. Это характерно для человека, вместе с тем она дает им количественную оценку, что характерно для компьютера. Этим она объединяет достоинства человеческого оперирования знаниями и вычислительной мощностью компьютера. Нечеткая логика, которая служит основой для реализации методов нечеткой обработки изображений, при решении задач анализа и распознавания изображений более естественно описывает характер человеческого мышления и ход его рассуждений, чем традиционные формальные системы. Именно поэтому использование математических средств для представления нечеткой исходной информации позволяет строить модели, которые наиболее адекватно отражают различные аспекты неопределенности, постоянно присутствующей в окружающей нас реальности. Однако при всем своем потенциале теория нечетких множеств не была способна решать практические задачи, а существовала действительно всего лишь как теория. Ситуация изменилась в последние 2-3 десятка лет и это связано с появлением прикладных реализаций для решения интеллектуальных задач с использованием теории нечетких множеств [61-65]. Следует отметить, что при решении задач с использованием аппарата нечеткой логики аксиомы вводятся по ходу решения задачи и формируются, главным образом, из эмпирических знаний человека.

В данном разделе представлены состав и структура системы обработки изображений с использованием концепции нечетких множеств. Сначала, в целях простоты изложения, рассмотрим систему нечеткой обработки изображений как черный ящик, который представляет собой удобную схему - преобразование входного пространства в выходные (рис.2.9). При этом изображение рассматривается как нечеткое множество и обрабатывается методами «мягкого вычисления».

Рис. 2.9. Схема преобразования входных изображений в выходные

На основе цели и задач исследований определена архитектура системы обработки изображений, основанной на концепции нечетких множеств. Функциональная схема этой системы представлена на рис. 2.10.

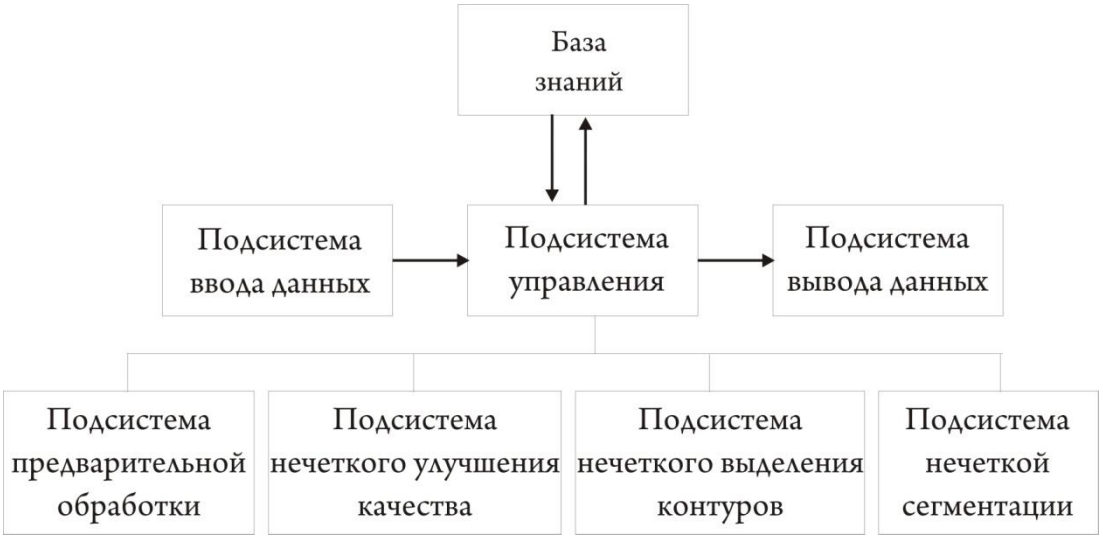


Рис. 2.10. Функциональная схема разрабатываемой системы «FuzzyIP»

В основу разрабатываемой системы обработки изображений заложены основные принципы разработки программной системы промышленного назначения. Эти принципы заключаются в следующем.

1. *Принцип эффективности.* Система должна обеспечить эффективный выбор схемы организации системы и математических методов решения задачи.

2. *Принцип простоты эксплуатации.* Максимальное удобство и простота общения пользователя с системой осуществлются входным языком, позволяющим компактно сформулировать задания.

3. *Принцип типизации и стандартизации.* Система должна учесть внесение корректив и доработок модулей при построении функциональных подсистем из стандартных элементов.

4. *Принцип формализации.* В системе задание на решение любой задачи должно производиться по общему правилу.

5. *Принцип надежности.* Система должна гарантировать достоверность результатов решения задач, что обеспечивается средством контроля вычислительного процесса на всех этапах работы ПК.

Реализация перечисленных принципов обеспечивает:

1) выбор соответствующей формальной модели в зависимости от условий задачи и исходного материала;

2) возможность работы всех функциональных подсистем системы в одном сеансе;

3) автономную работу функциональных подсистем системы;

4) простоту изменений, исправлений и расширений функциональных подсистем системы в рамках неизменности организации программ;

5) управление функционированием ПК с помощью центральной информационной программы управления.

Структура разрабатываемой системы «FuzzyIP» состоит из вспомогательных и функциональных подсистем. В состав системы входят следующие вспомогательные подсистемы:

- управления, осуществляющей организацию информационной связи в программной системе «FuzzyIP» с потребителем, координацию и управление работ составных подсистем в соответствии с заданием;

- ввода обрабатываемых изображений и исходных данных на ЭВМ для решения конкретных практических задач;

- вывода промежуточных и конечных результатов анализа и обработки изображений, а также значений требуемых параметров в удобной для потребителя форме, с соответствующими комментариями.

Под функциональной подсистемой понимается структурный единый комплекс, ориентированный на решение определенного класса задач обработки и анализа изображений. Разрабатываемая система будет состоять из следующих функциональных подсистем:

- предварительная обработка для улучшения результатов выделения контуров и сегментации изображений с использованием концепции нечетких множеств;

- выделение контуров изображений с использованием концепции нечетких множеств;

- сегментация изображений с использованием концепции нечетких множеств.

В функциональной подсистеме предварительной обработки изображений реализованы алгоритмы повышения качества изображений с использованием концепции нечетких множеств. Эти алгоритмы ориентированы на обработку исходного изображения с

целью улучшения результатов применения алгоритмов выделения границ (контуров) и сегментации.

В функциональной подсистеме выделения контуров изображений представлен набор процедур и функций, реализующих алгоритмы нечеткого приращения первого и второго порядков.

В функциональной подсистеме сегментации реализованы алгоритмы сегментации изображений на основе нечеткого критерия однородности области и нечеткой разрывности яркостных свойств изображения при переходе от одной однородной области к другой.

Все подпрограммы, реализованные в рассмотренных подсистемах, обладают следующими свойствами:

- упорядоченность: имеется один вход в подпрограмму и один выход из нее;

- автономность: передача управления осуществляется согласно значениям параметров, выработанным внутри каждой подпрограммы.

Основные подпрограммы предназначены для решения собственно задач анализа и обработки изображений на основе концепции нечетких множеств. При этом основное внимание уделяется задачам, связанным с выделением контуров и сегментацией изображений. Они составляют операционную часть алгоритмов системы «Fuzzy IP».

Вспомогательные подпрограммы служат для приведения анализируемого изображения к некоторой стандартной форме и вычисления значений вспомогательных параметров, входящих в алгоритмы основных задач, с привлечением общеизвестных процедур.

Функциональное назначение разрабатываемой системы «FuzzyIP», включающей в себя подпрограммы (набор алгоритмов) и программу-диспетчер, заключается в автоматизированном выборе оптимального, в некотором смысле, метода решения конкретной задачи с учетом ее специфики, и эта система предназначена для решения следующих задач обработки изображений с использованием концепции нечетких множеств:

- улучшение качества изображений;

- выделение краев области на анализируемом изображении;

- формирование области по свойствам пикселя анализируемого изображения;

- выделение некоторых реальных объектов на цветном изображении.

Все рассмотренные подсистемы, независимо от их функционального назначения, имеют унифицированную структуру. Это позволяет определить общую структуру для всех функциональных подсистем. Поэтому любую подсистему разрабатываемой системы FuzzyIP можно представить как схему преобразования, которая показана на рис. 2.11.

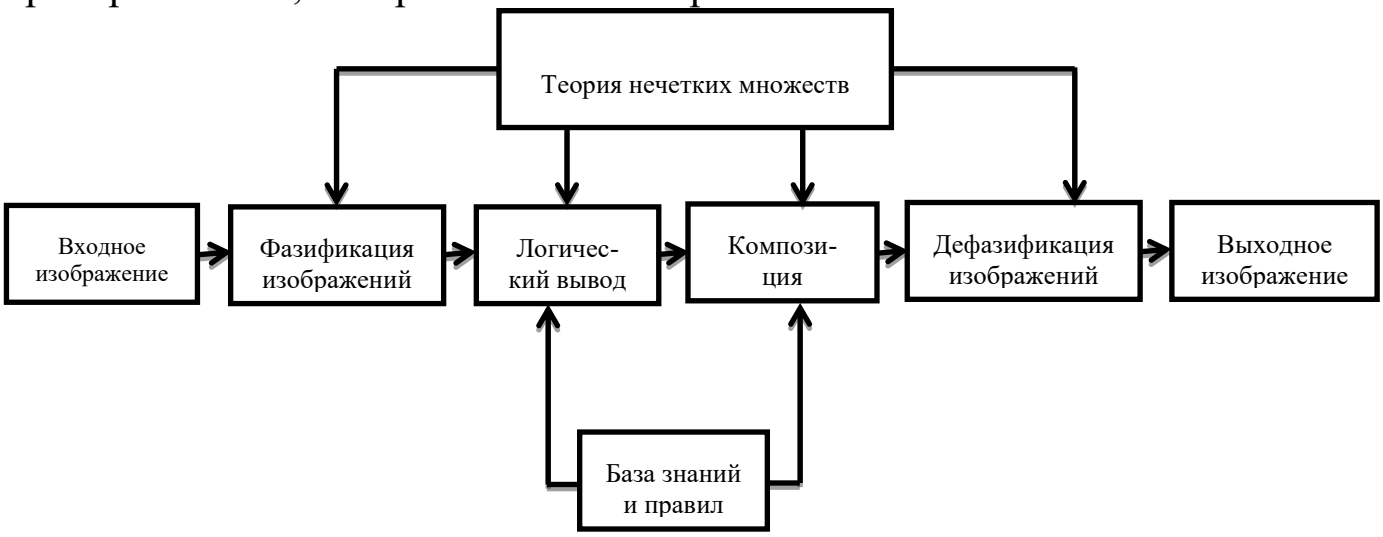


Рис. 2.11. Функциональная схема подсистемы разрабатываемой системы «FuzzyIP»

Вначале в каждой подсистеме происходит фаззификация изображения (т.е. приведение к нечеткости). Далее степень принадлежности элементов нечеткого множества (иными словами, пикселей изображения) изменяется в соответствии с базой нечетких правил. В последующем все переменные вывода объединяются для формирования одного нечеткого подмножества. И, наконец, происходит дефаззификация изображения (т.е. преобразование значений функций принадлежности в четкие значения яркости). Эти этапы обработки более подробно рассмотрены в разделе 2.2.2.

2.2.2. Основные этапы обработки изображений с использованием аппарата нечетких множеств

Предположим, что нас интересует установление правил для распознавания состояния фруктов, например, некоторого сорта абрикоса. Множество рассматриваемых абрикосов может быть разбито на три категории: неспелый, полуспелый и спелый. При этом на основе наблюдения фруктов на разных стадиях спелости можно

сделать вывод, что неспелые абрикосы выглядят зелеными, полуспелые - желтыми, а спелые - красными. Эти данные формализуются с помощью правил типа «ЕСЛИ-ТО». Вопросы формирования таких правил, которые представляют знания экспертов в предметной области, более подробно рассмотрены в разделе 2.12.

Далее рассмотрим определение способа использования входа (цвета) и базы знаний, представленной правилами «ЕСЛИ-ТО». Подобные процессы называются импликацией или логическими выводами. При этом исходные условия по каждому из правил должны быть обработаны и получено единое значение до осуществления импликации. В связи с тем, что мы имеем дело с нечеткими входами, то и выходы тоже являются нечеткими. Следовательно, для выходов должны быть определены функции принадлежности. На рис. 2.12 приведены функции принадлежности нечетких выходов, которые мы планируем использовать в этом примере.

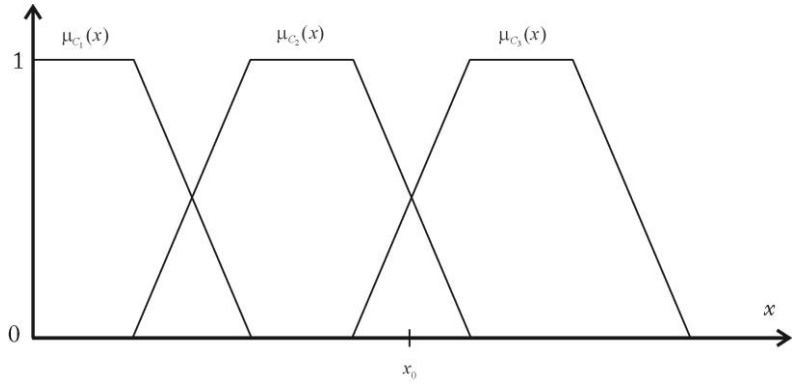


Рис. 2.12. Функции принадлежности, используемые при порождении нечетного множества $C(C=\{C_1,C_2,C_3\})$: C - нечеткие множества цвета; C_1 - нечеткое подмножество «Зелёный»; C_2 - нечеткое подмножество «Желтый»; C_3 - нечеткое подмножество «Красный»

Определим, что независимой переменной на выходе является спелость, которая отличается от независимых переменных на входах.

Вся информация, необходимая для формирования связи между входами и выходами, находится в базе правил. Например, на рис. 2.13, 2.14 приведены функции принадлежности красного и спелого абрикоса. Для того чтобы найти результат операции «И», между этими двумя функциями определяется минимум этих функций принадлежности [65,70]:

$$\mu_1(x,y) = \min\{\mu_{\text{зеленый}}(x), \mu_{\text{неспелый}}(y)\}, \qquad (2.4)$$

где цифра 1 в индексе означает, что это результат применения правила D1 из базы знаний. Все правила, использованные в данном разделе, приведены в разделе 2.2.3.

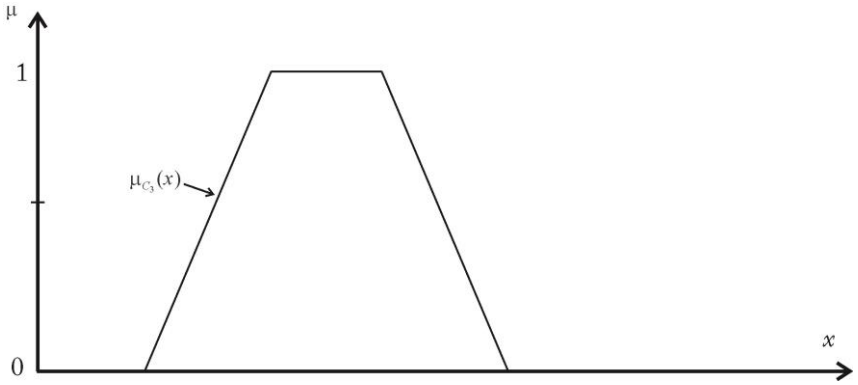


Рис. 2.13. Входная функция принадлежности, характеризующая красный цвет

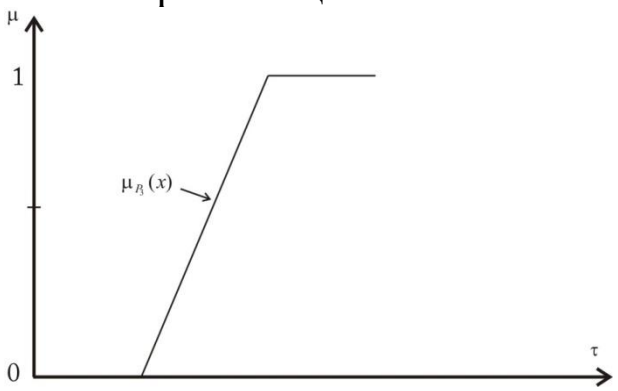


Рис. 2.14. Выходная функция принадлежности, соответствующая степени спелости абрикоса

Уравнение (2.4) представляет общую зависимость, включающую две функции принадлежности. При решении задачи распознавания состояния абрикосов нас интересует выход, получающийся в конкретном входе (например, зеленый).

Пусть x_0 – некоторое конкретное значение зеленого. Степень принадлежности зеленой цветовой компоненты в ответ на этот вход есть просто скалярная величина $\mu_{зеленый}(x_0)$. Требуется определить выход, соответствующий правилу D3 и конкретному значению x_0. Искомое значение выхода получается после выполнения операции «И» между $\mu_{зеленый}(x_0)$ и результатом общего вида $\mu_3(z, v)$, вычисленным при x_0:

$$R_2(\tau) = min\{\mu_{зеленый}(x_0), \mu_2(x_0, \tau)\}, \quad (2.5)$$

где $R_2(\tau)$ означает нечеткий выход в результате правила D1 и конкретного входа.

Единственной переменной в R_2, как и ожидалось, является выходная переменная τ. При выполнении операции минимума данной функции с положительной константой ξ все значения $\mu_2(x, \tau)$, большие этой константы, будут урезаны. Однако нас интересует только значение x_0 на цветовой оси. Поэтому рассмотрим только профиль функции, урезанной плоскостью $x = x_0$ вдоль по оси спелости. Он соответствует правилу D2, следовательно, $r = \mu_{зеленый}(x_0)$. Уравнение (2.5) есть выражение для этого профиля.

Нетрудно определить нечеткие отклики по остальным двум правилам и выбранному входу x_0, следуя аналогичному рассуждению:

$$R_2(\tau) = min\{\mu_{желтый}(x_0), \mu_2(x_0, \tau)\}, \quad (2.6)$$

$$R_1(\tau) = min\{\mu_{зеленый}(x_0), \mu_1(x_0, \tau)\}. \quad (2.7)$$

Отметим, что эти уравнения формируют выход, ассоциированный с конкретным правилом и заданным входом. Несмотря на то что значение на входе этих уравнений является скалярным, каждый из этих трех откликов образует нечеткое множество.

В целях формирования общего отклика решается вопрос группировки отдельных откликов. Например, три правила, сформулированные в начале данного раздела, присоединены операцией «ИЛИ». Поэтому конечный (сгруппированный) нечеткий выход задается в виде общего отклика, объединяющего три отдельных нечетких множества:

$$R = R_1 \; ИЛИ \; R_2 \; ИЛИ \; R_3. \quad (2.8)$$

Известно [65, 90, 91], что операция «ИЛИ» определяется как операция максимума. Исходя из этого конечного результата при $d = \{1, 2, 3\}$ и $r =$ {зеленый, желтый, красный}, запишем общий отклик в виде

$$R(\tau) = \max_d \left\{ \min_r \{\mu_r(x_0), \mu_d(r_0, \tau)\} \right\}. \quad (2.9)$$

Несмотря на то что данный результат получен в контексте некоторого примера, выражение (2.9) является совершенно общим. Достаточно принять $d = \{1, 2, ..., n\}$ для расширения его на n правил. Точно также расширяется функция принадлежности до любого конечного объема. Отметим, что уравнения (2.8) и (2.9) утверждают одно и то же: «Откликом нечеткой системы (R) является

объединение отдельных нечетких множеств, которые возникают в процессе импликации».

На рис. 2.15 приведены три входные функции принадлежности, оцененные в точке x_0. Выходы в ответ на значение входа x_0 приведены на рис. 2.16. Эти нечеткие множества являются урезанными профилями. Необходимо отметить, что численно R_3 состоит только из нулей, поскольку $\mu_{красный}\ (x_0) = 0$. На рис. 2.17 представлен итоговый результат R, сформированный из объединения R_1, R_2 и R_3. Несмотря на то, что полный выход, соответствующий конкретному входу, получен, он является нечетким множеством.

Завершающим этапом является формирование четкого выхода τ_0 на основе нечеткого множества R путем использования процесса, который называется приведением к четкому (дефаззификация). Существует много способов дефаззификации R для получения четкого выхода. Одним из наиболее распространенных способов является вычисление центра тяжести этого множества [91,94]:

$$y_0 = \frac{\sum_{\tau=1}^{k} \tau R(\tau)}{\sum_{\tau=1}^{k} R(\tau)}. \qquad (2.10)$$

Подставляя в это уравнение дискретные значения R, получаем $\tau_0 = 74{,}8$ (рис. 2.17). Этот результат свидетельствует о том, что заданный цвет x_0 означает приблизительно 75 % неспелости абрикосов.

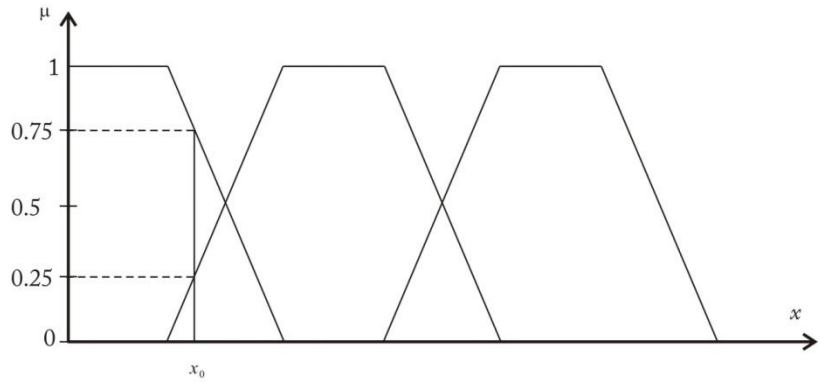


Рис. 2.15. Значения функции принадлежности, соответствующие заданному цвету x_0

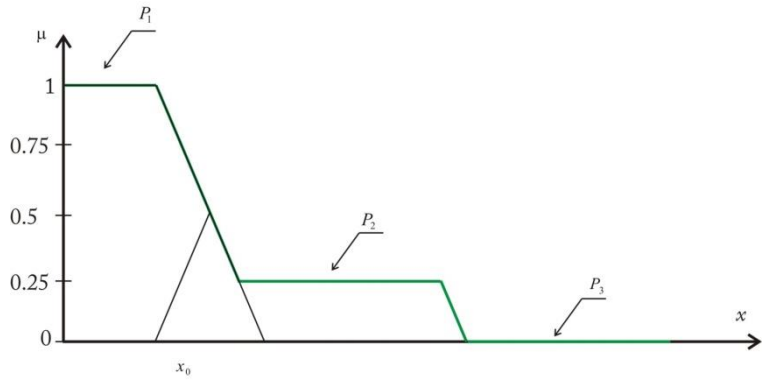


Рис. 2.16. Выделение нечеткого подмножества, сформированного в соответствии с уравнениями (2.6) и (2.7)

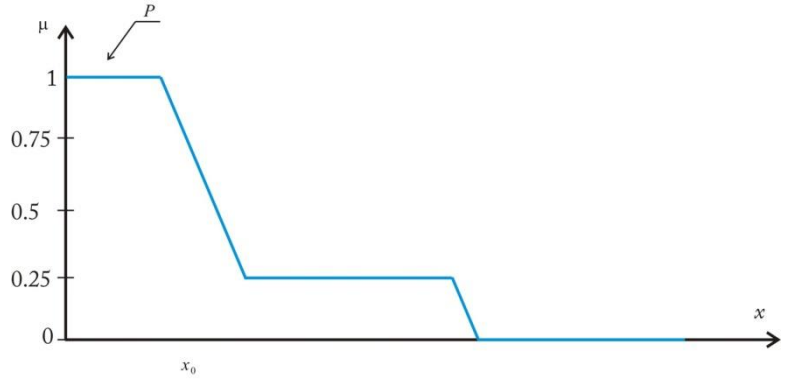


Рис. 2.17. Итоговое нечеткое подмножество, построенное с использованием уравнения (2.9)

До сих пор мы рассматривали правила «ЕСЛИ-ТО» (т.е. исходные условия), которые имеют только один параметр. Далее рассмотрим правила, содержащие несколько посылок. Например, вместо правила D1 даны его некоторые уточнения: «ЕСЛИ цвет рассматриваемого абрикоса зеленый ИЛИ его состояние твердое, ТО он неспелый». Для лингвистической переменной «твердый» должна быть определена функция принадлежности.

Для получения единственного числового результата по этому правилу требуется учитывать обе части исходных условий. Поэтому сначала оценивается заданное значение входного цвета «зеленый» с применением функции его принадлежности. Потом с применением функции принадлежности «твердый» определяется заданное значение состояния. Поскольку обе части этого условия связаны оператором «ИЛИ», выбирается максимум из двух результирующих значений. Затем полученное значение используется в процессе импликации для «урезания» выходной функции принадлежности.

Необходимо подчеркнуть, в случае использования m правил «ЕСЛИ–ТО», n входов (например, V_1, V_2, ..., V_N) и одного выхода (τ)

для формулирования правил нечеткой логики они могут иметь следующий вид:

$$\begin{gathered}\text{ЕСЛИ } (V_1, \Re_{11}) \text{ И } (V_2, \Re_{12}) \text{ И ... И } (V_n, \Re_{1n}) \text{ ТО } (\tau, B_1),\\ \text{ЕСЛИ } (V_1, \Re_{21}) \text{ И } (V_2, \Re_{22}) \text{ И ... И } (V_n, \Re_{2n}) \text{ ТО } (\tau, B_2),\\ \text{ЕСЛИ } (V_1, \Re_{m1}) \text{ И } (V_2, \Re_{m2}) \text{ И ... И } (V_n, \Re_{mn}) \text{ ТО } (\tau, B_m),\\ \text{ИНАЧЕ } (\tau, B_E),\end{gathered} \qquad (2.11)$$

где $\Re_{ij}$ – нечеткое множество, сформулированное с помощью *i*-го правила и *j*-й входной переменной; B_i – нечеткое множество, сформулированное с помощью выхода *i*-того правила. При этом предполагается, что составные части исходных условий правила связаны операторами «И»; B_e – нечеткое множество, сформулированное с помощью правила «ИНАЧЕ». Заметим, что правило «ИНАЧЕ» выполняется в случае, когда ни одно из предшествующих правил не удовлетворяется полностью.

При этом значения всех пар элементов исходных условий в каждом правиле должны быть определены, что позволяет определить единственное скалярное значение. В результате определения исходных условий *i*-го правила в (2.11) получаем скалярное значение χ_i, $(i = \overline{i,m})$:

$$\chi_i = \min_{1 \le j \le m} \left\{ \mu_{A_{ij}}(V_j) \right\}, \qquad (2.12)$$

где $\mu_{A_{ij}}(V_j)$ является функцией принадлежности нечеткого множества $\Re_{ij}$, полученного по значению *j*-го входа с χ_i уровнем мощности (или уровнем включения) *i*-го правила.

Правило «ИНАЧЕ» выполняется в случае, когда условия всех правил «ТО» удовлетворяются слабо. Данное правило дает сильный отклик в случае слабых откликов всех остальных. Поэтому правило «ИНАЧЕ» можно рассматривать как выполнение операции «НЕ» по условиям всех остальных правил. Используя этот способ комбинирования (операций «И») всех уровней условий правил «ТО», можно получить следующий уровень включения для правила «ИНАЧЕ»:

$$\chi_E = \min_{1 \le i \le m} \{1 - \chi_i\}. \qquad (2.13)$$

Если хотя бы одно из правил «ТО» включается на «полный уровень» (его отклик равен 1), то отклик правила «ИНАЧЕ» равен 0. При понижении уровней откликов условий правил «ТО» мощность правила «ИНАЧЕ» увеличивается и является нечетким эквивалентом

известных правил «ЕСЛИ-ТО-ИНАЧЕ», используемых в языках программирования [66,67].

При рассмотрении вопросов обработки цифровых изображений с помощью концепции нечетких множеств были использованы базы правил. Процесс формирования этих правил детально рассмотрен в следующем разделе.

2.2.3. Формирование базы правил

Нечеткие правила конструируются из функции принадлежности входов и выходов. Такие правила обеспечивают связь между предпосылкой и заключением, выраженными в форме «ЕСЛИ–ТО». Входы и выходы, приведенные к нечеткому виду, облегчают выделение правил и их обобщение. В общем случае правила могут быть получены из опыта одного или нескольких экспертов в рассматриваемой предметной области.

Несомненно, что определение хороших лингвистических правил зависит от объема и качества знаний эксперта. Однако в настоящее время не существует формализованного подхода к задаче точного и однозначного представления знаний в рамках теории нечетких множеств. В связи с этим представление знаний в фаззифицированном виде, по существу, произвольно.

Предположим, что нас интересует установление правил для распознавания состояния некоторого фрукта, например, абрикоса. Несомненно, что в приведенных выводах цвета абрикоса, наблюдаемые на различных стадиях спелости, обозначены не слишком четкими описаниями цветовых ощущений. Поэтому в качестве отправной точки при этих обозначениях должны быть использованы описания в нечеткой форме. Это достигается введением функции принадлежности абрикосов к классу неспелых, полуспелых или спелых на основе функции цвета (рис. 2.18, 2.19). В данном контексте цвет является лингвистической переменной, а конкретный цвет (т.е. красный фиксированный) – лингвистической величиной. Лингвистическая величина x определяется с использованием функций принадлежности (рис. 2.18). Приведенное знание может быть формализовано в форме следующих нечетких правил «ЕСЛИ–ТО»:

D1: ЕСЛИ цвет рассматриваемого абрикоса зеленый, ТО он неспелый
ИЛИ

D2: ЕСЛИ цвет рассматриваемого абрикоса желтый, ТО он полуспелый,
ИЛИ
D3: ЕСЛИ цвет рассматриваемого абрикоса красный, ТО он спелый.

При определении этого значения использована функция принадлежности, приведенная на рис. 2.18.

Приведенные три правила представляют собой результаты нашего знания о рассматриваемой задаче распознавания. Эти правила не являются ничем, кроме формализма для процесса мышления об этой задаче. Информация о цвете является той частью правила, которая находится слева от «ТО», и называется посылкой (или исходным условием). Правая часть называется следствием (или выводом).

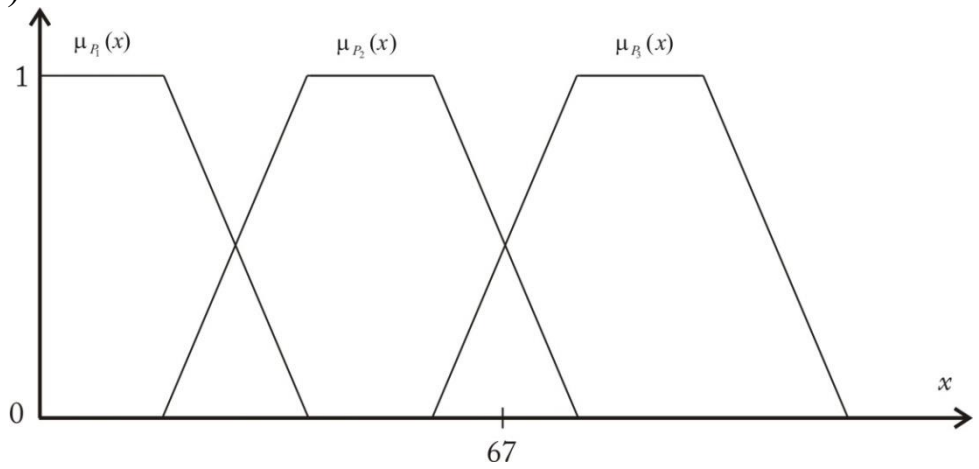


Рис. 2.18. Функции принадлежности, характеризующие спелость абрикоса: x - степень спелости абрикоса; P_1 - нечетное подмножество неспелого абрикоса; P_2 - нечеткое подмножество полуспелого абрикоса; P_3 - нечетное подмножество спелого абрикоса

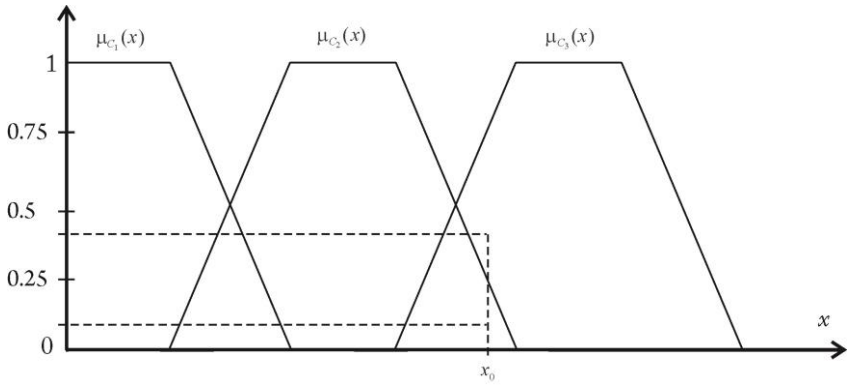


Рис. 2.19. Определение значения нечеткости для конкретного цвета x_0

Специфика четких правил диктует частые изменения, что вызывает значительные проблемы в оценке их поведения. Такие системы с часто изменяющимися параметрами не имеют практического значения. В то же время нечеткая реализация приводит к тому, что активизация правил будет зависеть от нечетких множеств и от правил, связывающих свойства входа и выхода.

База знаний состоит из определенного набора нечетких правил. Обычно число правил, требуемых в системе, связано с числом управляемых параметров. Таким образом, в рассматриваемом примере оценка состояния некоторого абрикоса является единственной переменной выхода, зависящей от двух входных переменных: цвета и мягкости абрикоса. Если принять, что каждый вход имеет функцию принадлежности, состоящую из трех элементов, тогда возможно девять входных комбинаций цвета и мягкости. В большинстве случаев можно использовать меньшее количество правил. Например, в работах [55, 68, 69] приведены нечеткие правила для сглаживания резкости и выделения контуров на изображении в градациях серого, которые состоят из трех входов и стольких же выходов:

ЕСЛИ пиксель темнее соседних пикселей,
ТО сделать его светлее,
ИНАЧЕ
ЕСЛИ пиксель светлее соседних пикселей,
ТО сделать его темнее,
ИНАЧЕ оставить без изменения.

В этих правилах различия значений яркостей между рассматриваемым пикселем и его соседями являются входными данными, а увеличение/уменьшение значений яркостей этого пикселя - выходным данным. Если предположить, что значения яркости изображения находятся в интервале [0, L-1] (где L - число градаций яркости), то простые треугольные нечеткие множества – среднее положительное и среднее отрицательное – определены на интервале [-L+1, L-1] и представляют яркие и темные значения яркостей входных переменных, а малый положительный, нулевой и небольшой отрицательный треугольные нечеткие множества определены как приращение указанного значения по следствиям составленных правил. Выходное значение добавляется к исходному значению яркости пикселя.

Правила или нечеткие ассоциации представляют собой знания, которые важны для системы с точки зрения отклика на всевозможные комбинации входов. Нечеткие системы запоминают наборы (банки) нечетких ассоциаций или правил.

2.3. Вопросы улучшения контраста изображений с использованием аппарата нечетких множеств

2.3.1. Локально-адаптивное улучшение качества изображений

Качество изображения на локальных участках можно улучшать, используя такие параметры интенсивностей пикселей, как среднее значение интенсивности и изменение интенсивности (или стандартное среднеквадратичное отклонение интенсивностей элементов локальной окрестности изображения). Среднее значение - это мера средней яркости. При вычислении и анализе средней яркости элементов изображения существует возможность ее коррекции, т.е. затемненные участки изображения делать более светлыми, а слишком светлые участки изображения затемнять. Однако в случае, если на изображении присутствуют темные и светлые области, то такой подход только ухудшит его визуальное восприятие. Поэтому целесообразно использовать еще один параметр, который характеризовал бы распределение яркостей элементов изображения в некоторой локальной окрестности. Другими словами, этот параметр характеризовал бы изменения интенсивностей или меру контрастности изображения.

Типичное локальное преобразование, основанное на этих параметрах, переводит интенсивность исходного изображения L_{in} в интенсивность нового изображения L_{out} путем осуществления следующей операции над расположением (i,j) каждого пикселя:

$$L_{out}(i,j) = k\frac{\bar{L}}{\sigma(i,j)}[L_{in}(i,j) - \bar{L}_{lokal}(i,j)] + \bar{L}_{lokal},$$

где

$\bar{L}$ - среднее значение интенсивностей элементов всего изображения L_{in};

$\sigma(i,j)$ - среднеквадратичное отклонение интенсивностей элементов локальной окрестности изображения в точке с координатами (i,j);

$L_{in}(i,j)$ - среднее значение интенсивности для окрестности с центром в точке (i,j);

k - некоторая константа, $0 < k < 1$.

Локальные изменения увеличиваются за счет умножения разности на

$$k\frac{\bar{L}}{\sigma(i,j)}.$$

Среднеквадратичное отклонение будет принимать меньшие значения в малоконтрастных окрестностях и более высокие значения в окрестностях с более высокой контрастностью. С учетом этого, а также того, что $\sigma(i,j)$ находится в знаменателе, участки с низкой контрастностью будут иметь большее усиление, чем участки с большей контрастностью. Среднее значение подставляется для восстановления среднего уровня интенсивности изображения на локальном участке. На практике целесообразно ограничивать диапазон значений множителя $k\frac{\bar{L}}{\sigma(i,j)}$ во избежание больших отклонений интенсивностей на отдельных участках.

Результаты компьютерного моделирования рассмотренного метода представлены на рис. 2.20 - 2.23.

Рис. 2.20. Исходное изображение

Рис. 2.21. Обработанное изображение при $m=35$ и $k=0.7$

Проанализируем результаты моделирования. В методе есть два основных параметра, которые существенно влияют на результат обработки - размер локальной окрестности m и коэффициент k. Рассмотрим два изображения, которые представлены соответственно на рис. 2.21 и 2.23, которые представляют результат обработки при одинаковом коэффициенте k, но разных размерах локальных окон m. Из этих изображений видно, что уменьшение размера локального

окна приводит к увеличению детальности обработки. Для анализа влияния коэффициента k при одинаковых размерах локальной окрестности m рассмотрим два других изображения, которые представлены на рис. 2.21 и 2.22. Уменьшение коэффициента k приводит к устранению резких перепадов на изображении и понижению его контрастности. Таким образом, используя различные значения параметров m и k, можно управлять уровнем контрастности и детальности обработки изображений.

Рис. 2.22. Обработанное изображение при $m = 35$ и $k = 0.3$

Рис. 2.23. Обработанное изображение при $m = 15$ и $k = 0.7$

2.3.2. Методы улучшения контраста изображений при нечеткой исходной информации

Концепция нечеткой обработки и идентификации изображений предполагает использование подходов проблемно-ориентированной предварительной обработки, сохраняющей информационные признаки объектов. Она позволяет упростить и ускорить процесс обучения и выделения (идентификации) объекта на изображении для нечеткой обработки за счет использования информации об иерархии признаков, что сокращает затраты времени на обработку. В этом разделе рассматривается нечеткое описание повышения контраста при нечеткой исходной информации.

Известно, что искаженные элементы изображения часто весьма заметно отличаются от соседних элементов. Это наблюдение послужило основой для многих алгоритмов, обеспечивающих подавление шума [67, 70, 71]. Если яркость данного элемента превышает среднюю яркость группы ближайших элементов на некоторую пороговую величину, яркость элемента заменяется на нечеткую среднюю яркость.

При обработке изображений для визуализации получили распространение методы, в которых часто отсутствуют строгие математические критерии оптимальности, их заменяют качественные представления о целесообразности той или иной обработки, опирающиеся на субъективные оценки результатов.

Подавляющее большинство процедур обработки для получения результата в каждой точке изображения привлекает входные данные из некоторого множества точек исходного изображения, окружающих обрабатываемую точку. Однако имеется группа процедур, где осуществляется поэлементная обработка. При размытом изображении каждый элемент можно рассматривать как нечеткое множество.

Сущность поэлементной обработки изображений сводится к следующему. Пусть $f(x,y)$ и $g(x,y)$ - значения яркости исходного и получаемого после обработки изображений соответственно в точке кадра, имеющей декартовы координаты: x – номер строки и y – номер столбца.

Поэлементная обработка означает, что существует функциональная зависимость между этими яркостями:

$$g(x,y) = F(f(x,y)),$$

позволяющая по значению исходного сигнала определить значение выходного сигнала.

Задача констрастирования связана с улучшением согласования динамического диапазона изображения и экрана, на котором выполняется визуализация. Если для цифрового представления каждого отсчета изображения отводится 1 байт (8 бит) запоминающего устройства, то входной или выходной сигналы могут принимать одно из 256 значений. Обычно в качестве рабочего используется диапазон 0...255, при этом 0 соответствует при визуализации уровню черного, а значение 255 – уровню белого. Предположим, что минимальная и максимальная яркости исходного изображения равны $f_{\min}$ и $f_{\max}$ соответственно.

Если эти параметры или один из них существенно отличается от граничных значений яркостного диапазона, то визуализированное изображение выглядит как неудобное, утомляющее при наблюдении.

Часто бывает удобно рассматривать изображение как реализацию нечеткого случайного процесса. Введем порождающую изображения непрерывную случайную функцию $f(x,y)$ двух переменных пространственных координат x, y. Случайный процесс $f(x,y)$ полностью описывается совместной плотностью вероятности $P[A]$.

Пусть $\{R^n,\sigma,P\}$ - вероятностное пространство, в котором R – пространство n-мерных вещественных векторов; σ – поле бореловских множеств в R^n; P – вероятностная мера на R^n.

Нечеткое случайное событие A в R^n есть нечеткое множество, функция принадлежности которого $\mu_A(x)\in\{R^n\to[0,1]\}$ измерима по Борелю при $x\in X$. Вероятность нечеткого случайного события A равна математическому ожиданию функции принадлежности μ_A и определяется при помощи интеграла Лебега-Стильтьеса в виде

$$P[A]=\int_{R^n}\mu_A(x)dP(x)=M[\mu_A].$$

Другой способ описания случайного процесса состоит в вычислении средних по ансамблю.

Так как операции дополнения, объединения, пересечения, суммы и произведения нечетких событий A и B используют $1-\mu_A$, $\max\{\mu_A,\mu_B\}$, $\min\{\mu_A,\mu_B\}$, $\mu_A+\mu_B$, $\mu_A\cdot\mu_B$, которые измеримы по Борелю, и

поскольку измеримы $\mu_A(x)$ и $\mu_B(x)$, то можно сказать, что нечеткие события в отношении операций дополнения, объединения и пересечения образуют борелевскую σ-алгебру. Также можно определить нечеткое вероятностное пространство, индуцированное вероятностным пространством $\{R^n, \sigma, P\}$.

Это позволяет определить основные характеристики случайных нечетких событий, такие, как математическое ожидание, дисперсия, начальные и центральные моменты и т.д.:

$$M[A] = \frac{1}{P(A)} \int_{R^n} x\mu_A(x) dP(x),$$

$$\sigma^2[A] = \frac{1}{P(A)} \int_{R^n} [x - M[A]]^2 \mu_A(x) dP(x),$$

$$m_v[A] = \frac{1}{P(A)} \int_{R^n} x^v \mu_A(x) dP(x),$$

$$M_v[A] = \frac{1}{P(A)} \int_{R^n} [x - M[A]]^v \mu_A(x) dP(x).$$

При рассмотрении дискретных нечетких множеств и случайных событий следует учитывать, что функция принадлежности задается только на дискретном множестве элементов из X или R^n, а интеграл следует заменить соответствующей суммой. Приводим выражения для основных вероятностных характеристик нечеткого полного случайного события:

$$P[A_\Theta] = \sum_{j=1}^{n} \mu_j p_j,$$

$$M[A_\Theta] = \frac{\sum_{j=1}^{n} \theta_j \mu_j p_j}{\sum_{j=1}^{n} \mu_j p_j},$$

На рис. 2.24, 2.25 показано обработанное изображение с использованием локально-адаптивного метода и при нечеткой исходной информации.


Рис. 2.24. Обработанное изображение с использованием локально-адаптивного метода


а)


б)

Рис. 2.25. Обработанное изображение при нечеткой исходной информации

2.3.2.1. Функции неопределенности нечетких множеств

Рассмотрим нечеткое множество $A=\{(x,\mu_A(x))\}$, где X – дискретное множество элементов $\{x_1,...,x_n\}$.

Мера нечеткости A может быть определена по аналогии с энтропией Шеннона в виде [41]:

$$H(\mu_A) = -a\sum_{i=1}^{n}\{\mu_A(x_i)\ln\mu_A(x_i) + [1-\mu_A(x_i)]\ln[1-\mu_A(x_i)]\},$$

причем $H(\mu_A)$ удовлетворяет следующим свойствам.

1. $H(\mu_A) \geq 0, H(\mu_A) = 0 \Leftrightarrow \mu_A(x)$ вырождена (т.е. принимает значение 0 или 1).

2. $H(\mu_A)$ достигает максимума по $\mu_A \Leftrightarrow \mu_A(x) \equiv \frac{1}{2} \forall x \in X.$

3. $H(\mu_A)$ вогнута по $\mu_A \in [0,1]$.

4. $H(\mu_A) = H(\mu_{\bar{A}}).$

5. $H(\mu_A) \geq H(\mu_{A^*}) \Leftrightarrow \mu_{A^*}(x) \geq \mu_A(x)$ при $\mu_A(x) \geq 1/2$, $\mu_{A^*}(x) \leq \mu_A(x)$ при $\mu_A(x) \leq \frac{1}{2}.$

6. $H(\max[\mu_A, \mu_B]) + H(\min[\mu_A, \mu_B]) = H(\mu_A) + H(\mu_B), \quad \forall \mu_A, \mu_B \in \{X \to [0,1]\}.$

7. $H(\mu_{A\cdot B}) = |A|H(\mu_A) + |B|H(\mu_B),$ где $B = \{(x, \mu_B(x))\}_{x\in X}$, $|A|$ и $|B|$ - мощности нечетких множеств A и B, соответственно

$$|A| = \sum_{i=1}^{n}\mu_A(x_i), \qquad |B| = \sum_{i=1}^{n}\mu_B(x_i),$$

$A \cdot B -$ произведение нечетких множеств A и B.

Энтропию нечеткого множества $H(\mu_A)$ будем называть прямой функцией неопределенности четвертого рода, порожденной функцией неопределенности четвертого рода, порожденной функцией принадлежности нечеткого множества A.

Если коэффициент $a = 1/n$, то $H(\mu_A)$ может иметь интерпретацию, несколько отличную от общепринятого толкования энтропии случайного события как недостоверности информационного предсказания результатов наблюдений. Например, $H(\mu_A)$ может рассматриваться как структурная мера неопределенности по составу структуры некоторой системы. И, кроме того, энтропия Шеннона и другие функции неопределенности первого рода определены на плоском множестве Δ_n n-мерного пространства, а $H(\mu_A)$ определяются на прямоугольном параллелепипеде $[0, 1]^n$.

Заметим, что возможно определение других прямых функций неопределенности четвертого рода, если при их конструировании исходить не из обобщения энтропии Шеннона, а из обобщения класса функций неопределенности первого рода.

2.3.2.2. Связанная и полная функции неопределенности четвертого рода

Рассмотрим нечеткое случайное событие $A = \{x, \mu_A(x) : P\{x = x_i\} = p_i \ (i = 1,...,n)\}$. Мера нечеткости события A может быть определена [70] как энтропия нечеткого подмножества A четкого множества X с распределением $p = (p_1, \ ..., \ p_n)$: $H(\mu_A, p) = -\sum_{i=1}^{n} \mu_A(x_i) p_i \ln p_i$.

Будем называть $H(\mu_A, p)$ связанной функцией неопределенности четвертого рода, соответствующей функции принадлежности $\mu_A(x)$ и распределению p на X.

В качестве другой связанной функции неопределенности четвертого рода, соответствующей $\mu_A(x)$ и p, является статически нормированная средняя мера нечеткости события A вида

$$H(p, \mu_A) = -\frac{1}{n} \sum_{i=1}^{n} p_i [\mu_A(x_i) \ln \mu_A(x_i) + (1 - \mu_A(x_i)) \ln(1 - \mu_A(x_i))].$$

Функции неопределенности четвертого рода $H(\mu_A, p)$ и $H(p, \mu_A)$ имеют концептуальное различие ввиду того, что $H(\mu_A, p)$ представляет собой по существу энтропию нечеткого подмножества состояний случайного вектора x с распределением p на X, а $H(p, \mu_A)$ – математическое ожидание прямой функции неопределенности – энтропии нечеткого множества, определенного на состояниях случайного вектора x с распределением p на X. При этом, если распределение p^0 вырождено, т.е. $p^0_{i_0} = 1, \ p^0_i = 0$ при $i \neq i_0$, то $H(p^0, \mu_A) \neq 0,$ хотя $H(\mu_A, p^0) = 0$.

Полная функция неопределенности четвертого рода определяется как полная энтропия [4, 41] нечеткого случайного события A в виде $H(p \lozenge \mu_A) = H(p) + H(p, \mu_A),$ где $H(p) -$ функция неопределенности первого рода для распределения p на X. Другие полные функции неопределенности четвертого рода определяются следующим образом:

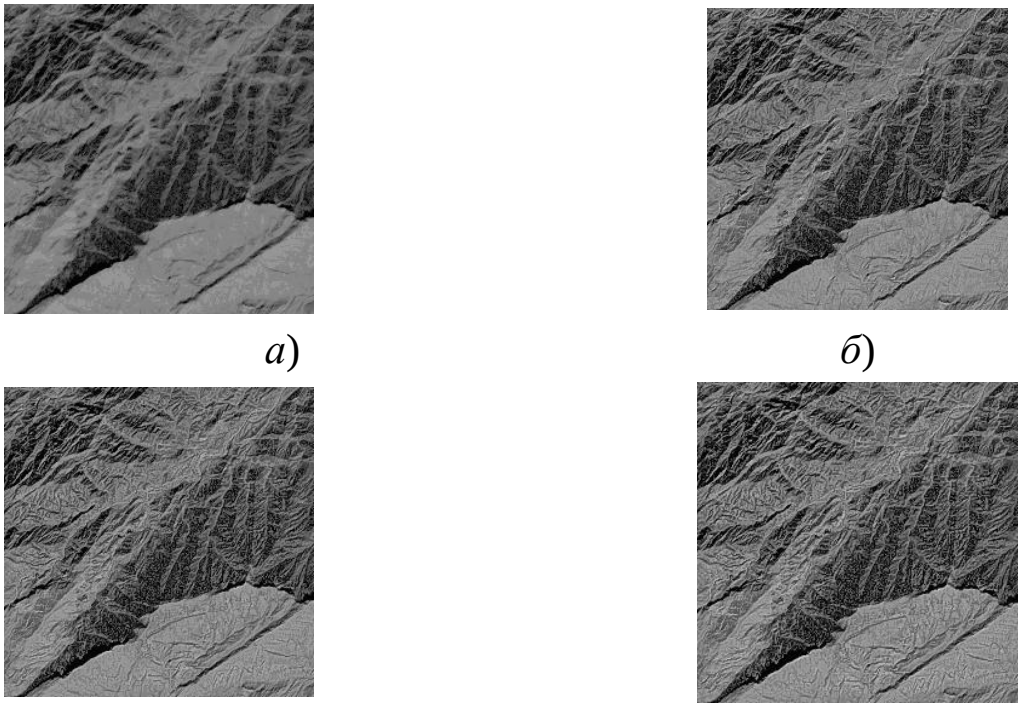
$$H(p \mu_A) = H(p) + H(\mu_A, p), \qquad H(p * \mu_A) = H(\mu_A) + H(p, \mu_A),$$
$$H(\mu_A * p) = H(\mu_A) + H(\mu_A, p), \qquad H(p, \mu_A) = H(p) + H(p, \mu_A) + H(\mu_A),$$
$$H(\mu_A, p) = H(\mu_A) + H(\mu_A, p) + H(p).$$

При этом можно определить, что приведенный класс функций неопределенности четвертого рода порождается рассмотрением бинарных отношений понятий случайности и нечеткости;

естественно, что могут быть определены и более сложные классы функций неопределенности четвертого рода, основанные на рассмотрении других, более тонких отношений между понятиями случайности и нечеткости с привлечением классов функций неопределенности первого и второго рода.

Отметим, что энтропия локальной окрестности изображения определяется как сумма произведений вероятностей элементов окрестности с различными значениями яркостей на логарифм этих вероятностей, взятая с противоположным знаком (рис. 2.26).

Локальную окрестность следует рассматривать как некоторую сложную систему, состоящую из простых подсистем - элементов окрестности, и уже с этих позиций искать энтропию окрестности. Кроме того, такой подход для определения энтропии локальной окрестности требует значительных вычислительных затрат.



а) *б*)

в) *г*)

Рис. 2.26. Локально-адаптивный метод повышения визуального качества изображений с использованием энтропии: *а* - исходное изображение; *б* - изображение *а*, обработанное известным методом с использованием энтропии; *в* - изображение *а*, обработанное известным методом с использованием классического подхода к определению энтропии; *г* - изображение *а*, обработанное предложенным методом с использованием нечеткого подхода к определению энтропии

Далее приводим алгоритм линейного повышения контраста при нечеткой исходной информации.

Функции принадлежности $\mu^f(x,y)$ и $\mu^g(x,y)$ определяются следующим образом:

1. Нормализация:

$$u(x,y) = l\frac{f(x,y) - f_{\min}}{f_{\max} - f_{\min}}.$$

2. Фаззификация:

$$\mu_i^f(x,y) = \frac{1}{1 + \dfrac{u(x,y) - c_i}{\sigma_f}},\ i = \overline{1,k}.$$

3. Уточнение фаззификации:

$$\mu_i^f(x,y) = \begin{cases} 2(\mu_i^f(\mathrm{x},\mathrm{y}))^2, & 0 \le \mu_i^f(\mathrm{x},\mathrm{y}) \le \dfrac{1}{2}, \\ 1 - 2(1 - \mu_i^f(\mathrm{x},\mathrm{y}))^2, & \dfrac{1}{2} < \mu_i^f(\mathrm{x},\mathrm{y}) \le 1. \end{cases}$$

4. Нормализация:

$$v(x,y) = l\frac{g(x,y) - g_{\min}}{g_{\max} - g_{\min}}.$$

5. Фаззификация:

$$\mu_i^g(x,y) = \frac{1}{1 + \dfrac{v(x,y) - c_i}{\sigma_g}},\ i = \overline{1,k}.$$

6. Уточнение фаззификации:

$$\mu_i^g(x,y) = \begin{cases} 2(\mu_i^g(\mathrm{x},\mathrm{y}))^2, & 0 \le \mu_i^g(\mathrm{x},\mathrm{y}) \le \dfrac{1}{2}, \\ 1 - 2(1 - \mu_i^g(\mathrm{x},\mathrm{y}))^2, & \dfrac{1}{2} < \mu_i^g(\mathrm{x},\mathrm{y}) \le 1. \end{cases}$$

Здесь c_i, σ_f и σ_g - параметры функции принадлежности.

Изображения, вводимые в компьютер, часто оказываются малоконтрастными, т.е. у них изменения яркости малы по сравнению с ее средним значением. При этом яркость меняется не от черного к белому, а от серого к чуть более светлее серого. То есть реальный диапазон яркости оказывается намного меньше допустимого (шкалы яркости). Задача повышения контраста заключается в «растягивании» диапазона яркости изображения на всю шкалу.

Эту задачу можно решать при помощи поэлементного преобразования линейного контрастирования:

$$g(x,y) = af(x,y) + b,$$

т.е. берутся такие a и b, которые приводят нечеткие значения поля яркости к некоторым стандартным величинам. Здесь предварительно оцениваются $M[f(x,y)]$, $\sigma[f(x,y)]$, а коэффициенты a, b выбираются так, чтобы для выходного поля получить $M[g(x,y)]$, $\sigma[g(x,y)]$:

$$\bar{g}(x,y) = \frac{f(x,y) - M[f(x,y)]}{\sigma[f(x,y)]} \cdot \sigma[g(x,y)] + M[g(x,y)] =$$

$$= \frac{\sigma[g(x,y)]}{\sigma[f(x,y)]} f(x,y) + M[g(x,y)] - M[f(x,y)] \frac{\sigma[g(x,y)]}{\sigma[f(x,y)]},$$

т.е.

$$a = \frac{\sigma[g(x,y)]}{\sigma[f(x,y)]};\ b = M[g(x,y)] - M[f(x,y)] \frac{\sigma[g(x,y)]}{\sigma[f(x,y)]}.$$

Здесь

$$M[f(x,y)] = \frac{\sum_{i=1}^{k} f_i(x,y) \cdot \mu_i^f(x,y)}{\sum_{i=1}^{k} \mu_i^f(x,y)},$$

$$M[g(x,y)] = \frac{\sum_{i=1}^{k} g_i(x,y) \cdot \mu_i^g(x,y)}{\sum_{i=1}^{k} \mu_i^g(x,y)};$$

$$g(x,y) = F(f(x,y)) = \begin{cases} 0, & \bar{g}(x,y) < 0, \\ \bar{g}(x,y), & 0 \le \bar{g}(x,y) \le 255, \\ 255, & \bar{g}(x,y) > 255. \end{cases}$$

Таким образом, при обработке изображений требуется по некоторым признакам выделять некоторые однородные области изображения. Этапы предварительной обработки изображения позволяют уменьшить влияние искажений на процесс распознавания. Тем не менее, имеет место распознавание в условиях неполной и нечеткой информации. Для ее решения наиболее подходят технологии нечеткой логики, нечеткая логика при этом выступает в роли классификатора. Применение нечеткой логики в задачах обработки визуальной информации обосновывается также свойством обучаемости или адаптивности нечеткой логики к новым задачам, при этом сохраняются архитектура сети и алгоритм ее

функционирования. Программная реализация и применение рассмотренного метода в задачах обработки изображений будут рассмотрены ниже, где предусмотрены вопросы разработки алгоритмов с использованием аппарата нечетких множеств.

На рис. 2.27 показан результат линейного повышения контраста при нечеткой исходной информации.

Рис. 2.27. Результат линейного повышения контраста при нечеткой исходной информации

2.4. Сегментация цветных изображений на основе кластеризации

2.4.1. Сегментация цветных изображений на основе кластеризации по методу *k*-средних

Одно из главных достоинств кластерного анализа в том, что его можно использовать циклически до тех пор, пока не будут достигнуты необходимые результаты. При этом каждый цикл дает информацию, которая влияет на применяемые в дальнейшем подход и направление. Этот процесс можно представить как систему с обратной связью. На этом основывается еще одно применение кластерного анализа - распознавание с самообучением.

Каждая единица совокупности в кластерном анализе считается «точкой в признаковом пространстве». Значение каждого из признаков у данной единицы совокупности служит ее координатой в этом «пространстве». Таким образом, «признаковое пространство» – это область варьирования всех признаков совокупности изучаемых явлений, а «вектор признаков» - объект (точка) в данном «пространстве».

Для решения задачи классификации массива большого объема многомерных наблюдений, таких как данные грозопеленгации, воспользуемся хорошо зарекомендовавшим себя методом кластерного анализа - *k*-средних (*k*-means). Суть данного метода

кластеризации заключается в разбиении массива данных на кластеры таким способом, чтобы минимизировать сумму расстояний от объектов до соответствующих им центров кластеров. За меру близости примем евклидово расстояние, так как рассматриваемое явление, а именно грозовой очаг, является локализуемым в Евклидовой метрике [69, 70]. Тогда геометрическое расстояние в n - мерном признаковом пространстве между точками p и q с n - координатами (признаками) определяется так:

$$r_{p,q} = \sqrt{\sum_{i=1}^{n} (X_{ip} - X_{iq})^2},$$

где X – вектор признаков: $X = (X_1, X_2, \ldots, X_n)$.

Квадрат евклидова расстояния между X_p и X_q представим в виде

$$r_{pq}^2 = \left(X_p - X_q\right)^T \left(X_p - X_q\right).$$

В таком случае алгоритм стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров.

Алгоритм работы k-средних имеет следующий вид:

1. Принимаем входные данные и предполагаемое число кластеров k.

2. Выбираем центры кластеров.

3. Рассчитываем расстояния от центров кластеров до всех объектов и ассоциируем каждый объект с ближайшим центром кластера.

4. Перемещаем центр кластера в центроид его точек данных, т.е. вычисленные средние становятся координатами нового центра каждого кластера.

5. Возвращаемся на шаг 3 пока не достигнута сходимость, например, центр кластера остается неподвижен, заняв устойчивое положение, т.е. пока состав кластеров не перестанет меняться.

Таким образом, алгоритм k-средних принимает в качестве входных данных вектор признаков X, содержащий n точек, а также параметр k, задающий требуемое количество кластеров. На выходе - набор из k центроидов кластеров, кроме того, всем точкам множества X присваиваются метки, относящие их к определенному кластеру, причем все точки в пределах данного кластера расположены ближе к своему центроиду, чем к любому другому центроиду.

На рис. 2.28 приведены совмещенные данные радиолокатора МРЛ-5 и грозопеленгатора LS8000.

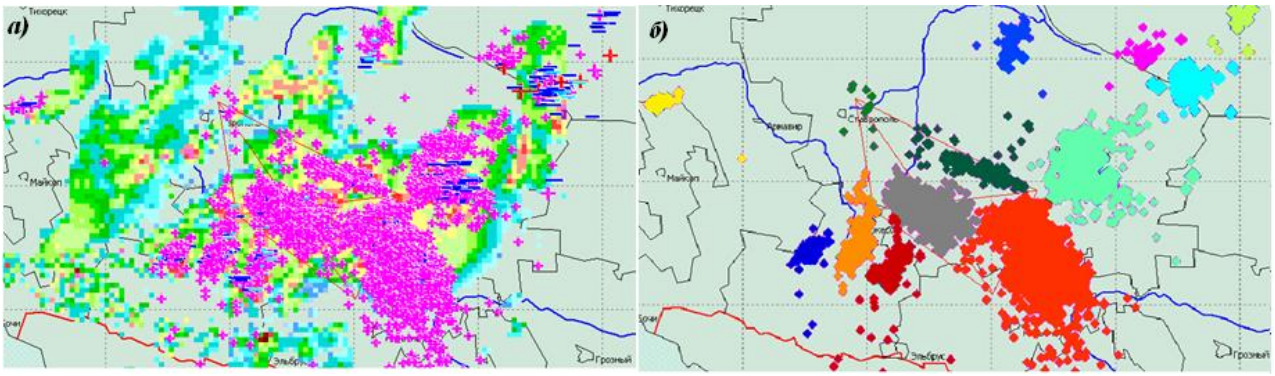


Рис. 2.28. Совмещенные данные радиолокатора МРЛ-5 и грозопеленгатора LS8000 (*а*). Изображены отражаемость облака и разряды внутриоблачные и между облаком и землей. Плюс обозначает положительную полярность молний, минус – отрицательную. Полученные в результате обработки кластеры (*б*)

Для определения числа кластеров используется «разведочный» алгоритм: сначала совокупность делится на два кластера, затем на три и так до тех пор, пока не будет найдено оптимальное число кластеров.

После получения результатов следует проверить правильность кластеризации. Для этого рассчитываются средние значения для каждого кластера. При хорошей кластеризации должны быть получены сильно отличающиеся средние для всех измерений или хотя бы большей их части.

Далее рассмотрим задачу, основной целью которой является автоматическая сегментация на основе кластеризации (метод *k*-средних) цветных изображений, представленных в цветовом пространстве *L*a*b**.

2.4.2. Сегментация цветных изображений на основе кластеризации по методу c-средних

На основе нечеткого c-means алгоритм выполняет кластеризацию данных. Этот алгоритм кластеризации предложил Джеймс Бэздэк (James Bezdek) в 1981 г.

Задача нечеткой кластеризации ставится следующим образом.

Дано:

$X = (X_1, X_2, ..., X_n)^T$ – объекты, подлежащие кластеризации (n – количество объектов).

Каждый объект $X_k = \left(x_{k_1}, x_{k_2}, ..., x_{k_p}\right)$ представляет собой точку в p-мерном пространстве признаков $\left(k = \overline{1,n}\right)$; c – количество кластеров $\left(2 \le c < n\right)$.

Необходимо каждому элементу множества X поставить в соответствие степени принадлежности к классам.

Элементы одного кластера должны быть так близки каждый к каждому, как это только возможно, и, одновременно, кластеры должны быть на наибольшем удалении друг от друга. Для обеспечения управляемости процесса кластеризации необходимо использовать меру близости, в качестве которой обычно определяют расстояние между двумя объектами (точками в p-мерном пространстве) X_k и X_i в виде вещественной функции $d : X \times X \to R^+$ такой, что

$$d(X_k, X_{i1}) = d_{ki} \ge 0,$$
$$d_{ki} = 0 \Leftrightarrow X_k = X_i,$$
$$d_{ki} = d_{ik}.$$

Дополнительно, если функция d удовлетворяет правилу треугольника, т.е. $d_{ki} \le d_{kj} + d_{ji}$, тогда эта функция является метрикой, хотя выполнения этого свойства не всегда необходимо для задач кластеризации.

Любое разбиение множества $X = \left(X_1, X_2, ..., X_n\right)^T$ на нечеткие подмножества S_i $(i = \overline{1,c})$ может быть полностью описано функцией принадлежности $\mu_{S_i} : X \to [0,1]$.

Обозначим через μ_{ik} степень принадлежности объекта $X_k = \left(x_{k_1}, x_{k_2}, ..., x_{k_p}\right)$ к подмножеству S_i, т.е. $\mu_{ik} \equiv \mu_{S_i}(X_k)$, и через V_{cn} - множество всех действительных матриц размером $c \times n$. Тогда нечетким c-разбиением (или матрицей степеней принадлежности) называется матрица $M = [\mu_{ik}] \in V_{cn}$ при выполнении следующих условий:

1. $\mu_{ik} \in [0,1],\ i = \overline{1,c},\ k = \overline{1,n}.$
2. $\sum_{i=1}^{c} \mu_{ik} = 1,\ k = \overline{1,n}$.
3. $\sum_{k=1}^{n} \mu_{ik} \in (0,n),\ i = \overline{1,c}$.

В отличие от четкого, при нечетком c-разбиении любой объект одновременно принадлежит к различным кластерам, но с разной

степенью. Условия 2) и 3) требуют только, чтобы сумма степеней принадлежности объекта ко всем кластерам была нормализована к 1, а также, чтобы количество кластеров, к которым принадлежит объект, не превышало c.

Обозначим центры кластеров, т.е. точки в p-мерном пространстве, вокруг которых сконцентрированы соответствующие объекты, через $V_i = (v_{i1}, v_{i2}, ..., v_{ip}),\ i = \overline{1,c}$.

При использовании эвклидового расстояния задача нечеткой кластеризации состоит в нахождении такой матрицы степеней принадлежности M и таких координат центров кластеров $V = (V_1, V_2, .., .V_c)$, которые обеспечивают минимум следующего критерия:

$$\sum_{i=1}^{c} \sum_{k=1}^{n} (\mu_{ik})^m \cdot \left\| X_k - V_i \right\|^2 \to \min ,$$

где $V_i = \dfrac{1}{\sum\limits_{k=1}^{n} \mu_{ik}} \sum\limits_{k=1}^{n} (\mu_{ik})^m \cdot X_k$ - центр i-го кластера, $i = \overline{1,c}$; m - так называемый экспоненциальный вес ($m \geq 1$).

Значение экспоненциального веса устанавливается до начала кластеризации. Экспоненциальный вес m влияет на матрицу степеней принадлежности. Чем больше m, тем конечная матрица c-разбиения становится более «размазанной», и при $m \to \infty$ она примет вид $M = [1/c]$, что является очень плохим решением, так как все объекты принадлежат ко всем кластерам с одной и той же степенью. Также экспоненциальный вес позволяет при формировании координат центров кластеров усилить влияние объектов с большими значениями степеней принадлежности и уменьшить влияние объектов с малыми значениями степеней принадлежности. На сегодня не существует теоретически обоснованного правила выбора значения m. Обычно устанавливают $m = 2$.

Приводим алгоритм.

Шаг 1. Считывание изображения.

Шаг 2. Преобразование изображения из цветовой системы RGB в цветовую систему *L*a*b**.

Шаг 3. Классификация цветов в пространстве '*a*b**' с использованием кластеризации (метод *k*-средних или метод *c*-средних).

Шаг 4. Присвоение меток каждому пикселю изображения на основе метода *k*-средних или метода *c*-средних.

Шаг 5. Создание сегментированного изображения на основе цветного.

Шаг 6. Сегментация ядер на основании отдельного изображения.

Шаг 1. Считывание изображения. Считаем файл, который содержит изображение. Здесь применен метод окрашивания для детального анализа патологий (рис. 2.29).

Рис. 2.29. Исходное изображение

*Шаг 2. Преобразование изображения из цветовой системы RGB в цветовую систему L*a*b*.* Какое количество цветов видно на изображении, когда не принимать во внимание возможность комбинации яркостей? На самом деле их пять. Следует отметить различия этих цветов между собой. Цветовое пространство *L*a*b** (оно еще известно как CIELAB или CIE *L*a*b**) позволяет различать эти визуальные различия.

Цветовое пространство *L*a*b** получено на основе трехцветных значений CIE XYZ. Пространство *L*a*b** включает информацию о значении интенсивности '*L**', значении цветности '*a**', которое показывает, какой цвет выбран на красно-зеленой оси, а значение цветности '*b**' показывает, какой цвет выбран на голубо-желтой оси. Вся информация о цветах содержится в значениях '*a**' и '*b**'. Оценить разницу между двумя цветами можно с использованием евклидового расстояния.

Преобразуем изображение в цветовое пространство *L*a*b**.

*Шаг 3. Классификация цветов в пространстве 'a*b*' с использованием кластеризации (метод k-средних или метод c-средних).*

Кластеризация приводит до разделения объектов на группы. Кластеризация методом *k*-средних или методом *c*-средних приводит также к локализации объектов в пространстве. Поиск разделения, т.е.

какой объект к какому классу принадлежит, происходит на основе анализа метрического расстояния между объектами.

Далее на основании информации о цветах в пространстве *'a*b*'* каждому пикселю объекта присваивается значение *'a*'* и *'b*'*. Используем кластеризацию методом *k*-средних для разделения объектов на пять кластеров и кластеризацию методом *c*-средних для разделения объектов на три кластера. Для этого используем евклидовую метрику.

Шаг 4. Присвоение меток каждому пикселю изображения на основе метода k-средних или метода c-средних. Для каждого объекта на исходном изображении метод *k*-средних или *c*-средних возвращает индекс соответствующего кластера. Значение параметра центра кластера, которое получено в результате применения метода *k*-средних или *c*-средних, будет использовано при дальнейшей демонстрации метода. Отметим пиксели, которые содержатся в индексе кластера (рис. 2.30, 2.31).

Рис. 2.30. Изображение, отмеченное кластерными индексами по методу *k*-средних

Рис. 2.31. Изображение, отмеченное кластерными индексами по методу *c*-средних

*Шаг 5. Создание сегментированного изображения на основе цветного. О*бъекты можно разделить на изображения по цветам (рис. 2.32 – 2.39).

Рис. 2.32. Объекты в кластере 1 по методу *k*-средних

Рис. 2.33. Объекты в кластере 1 по методу *c*-средних

Рис. 2.34. Объекты в кластере 2 по методу *k*-средних

Рис. 2.35. Объекты в кластере 2 по методу *c*-средних

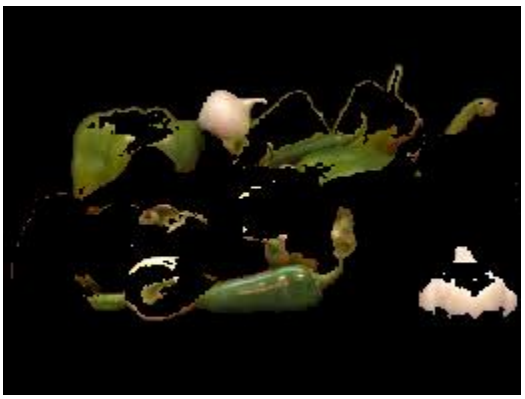


Рис. 2.36. Объекты в кластере 3 по методу *k*-средних

Рис. 2.37. Объекты в кластере 3 по методу *c*-средних

Рис. 2.38. Объекты в кластере 4 по методу *k*-средних

Рис. 2.39. Объекты в кластере 5 по методу *k*-средних

Шаг 6. Сегментация ядер на основании отдельного изображения. Рассмотрим изображение (рис. 5.35), которое содержит синие объекты. Отметим, что они являются темно-синими и светло-синими. Используя значение '*L**' в цветовом пространстве *L*a*b**, можно отделить темно-синие объекты от светло-синих.

2.5. Интуитивная нечеткая обработка изображений

Рассмотрим изображение A размера $M \times N$ пиксель, наличие серого уровня r в диапазоне между 0 и L-1. При применении интуитивного нечеткого множества (ИНМ) для обработки изображений они рассматриваются как массив нечетких синглтонов. Каждый элемент массива указывает на значение принадлежности $\mu_{\tilde{A}}(g_{ij})$ серого уровня g_{ij}, соответствующее (i,j)-му пикселю, в соответствии с предопределенными свойствами изображения, такими как яркость, резкость, однородность [74].

В виде обобщения этого подхода введем следующее представление изображения в интуитивной нечеткой среде.

Изображение A, описанное ИНМ, имеет вид

$$A = \left\{ \left\langle g_{ij},\ \mu_A(g_{ij}),\ \nu_A(g_{ij}) \right\rangle \middle| g_{ij} \in \{0,\ldots,L-1\} \right\}, \qquad (2.14)$$

где $i \in \{1,\ldots,M\}$, $j \in \{1,\ldots,N\}$, $\mu_A(g_{ij})$ и $\nu_A(g_{ij})$ обозначают соответственно степень принадлежности и непринадлежности (i,j)-го пикселя к множеству в соответствии со свойствами изображения.

Функции μ_A и ν_A соответствуют принадлежности и непринадлежности множествам компонентов изображения. Если

вместо ИНМ мы рассмотрим нечеткое множество, то определение (2.14) приведет к одному из приведенных определений в [72-74].

Методы обработки изображений на основе теории ИНМ предоставляют гибкую математическую базу, для того чтобы справиться с «качественными» свойствами, такими, как контрастность изображения в условиях неоднозначности и расплывчатости, часто присутствующих в цифровых изображениях.

В терминах интуитивной нечеткой обработки изображений (ИНОИ) вопрос, который, естественно, возникает при попытке определить ИНМ яркости пикселей, можно сформулировать так: «как мы можем определить принадлежность и непринадлежность функции серых уровней для описания изображения в ИНМ или более понятным с человеческим восприятием: «как ярок серый уровень и как мы можем быть уверены, что он такой яркий?».

Неопределенность в изображениях исходит из различных факторов. Они влияют на нашу уверенность в принятии решения, является ли пиксель «серым» или «резким», и поэтому вводят определенные сомнения, связанные с соответствующей точкой. Определение принадлежности компонента А-ИНМ, описывающего яркость пикселей изображения, является более простой задачей, которая может быть проведена аналогичным образом, как и в традиционных нечетких системах обработки изображений. В представленной эвристической системе мы рассматриваем принадлежность значения уровня серого цвета g его нормализованному уровню интенсивности:

$$\mu_A(g) = \frac{g}{L-1}, \tag{2.15}$$

где $g \in \{0,\ldots,L-1\}$.

Следует отметить, что любой другой метод расчета μ_A также может быть применен.

Шум квантования является неотъемлемой частью любой физической системы, которая включает в себя аналого-цифровое преобразование. Для того чтобы смоделировать неточность этого типа серых уровней, в [75] было предложено понятие нечеткой гистограммы на основе нечетких чисел.

Нечеткое число $\tilde{g}: R \to [0,1]$ является нормальным и выпуклым. Мы ограничиваем наш выбор для симметричных нечетких чисел, которые концептуально подходят для представления понятия «серый

уровень примерно *g*». Симметричное треугольное нечеткое число определяется как

$$\mu_{\tilde{g}}(x) = \max\left\{0{,}1 - \frac{|x-g|}{p}\right\},$$

где положительный параметр *p* контролирует форму числа.

Рассматривая уровни серого в виде нечетких чисел, в [73] понятие гистограммы определено в нечеткой обстановке. Нечеткая гистограмма цифрового изображения является последовательностью $h_A^f(g)$ с $g \in \{0,...,L-1\}$ и определяется как

$$h_A^f(g) \stackrel{\Delta}{=} \left\| \left\{ \left\langle (i,j),\ \mu_{\tilde{g}_{ij}}(g) \right\rangle | i \in \{1,...,M\},\ j \in \{1,...,N\} \right\} \right\|,$$

где $\|\cdot\|$ обозначает число элементов в нечетком множестве.

Более того, $h_A^f(g)$ представляет собой частоту появления уровня яркости «примерно *g*». Однако, из-за его определения, нечеткой гистограмме не удается быть функцией плотности вероятности.

Нормированная гистограмма имеет вид

$$\tilde{h}_A^f(g) = \frac{h_A^f(g)}{\sum_{g=0}^{L-1} h_A^f(g)},$$

где $g \in \{0,...,L-1\}$.

Влияние ошибок квантования можно увидеть путем сравнения четких и нечетких гистограмм. В случае «жесткого» первого порядка статистики существует ряд уровней серого с нулевой или почти нулевой частотой появления из-за шумов квантования, в то время как уровни серого в их окрестности обладают высокой частотой [75]. Это не тот случай, когда рассматривается нечеткая гистограмма. Поэтому для того чтобы моделировать неопределенности родом из шума квантования, колебания, соответствующие уровню серого изображения, должны быть пропорционалены нормированной абсолютной разнице между нормализованной четкой и нечеткой гистограммой:

$$\pi_A(g) \propto \frac{\left|\tilde{h}_A^c(g) - \tilde{h}_A^f(g)\right|}{\max_g \left\{\left|\tilde{h}_A^c(g) - \tilde{h}_A^f(g)\right|\right\}},$$

где $\tilde{h}_A^c$ является нормированной гистограммой четких изображений.

Комбинируя все приведенные выше утверждения, смоделируем гистограмму *πA(g)*, соответствующую серому уровню *g* изображения *A*, определенного в (2.14):

$$\pi_A(g) = (1-\mu_A(g))\frac{\left|\tilde{h}_A^c(g) - \tilde{h}_A^f(g)\right|}{\max\left\{\left|\tilde{h}_A^c(g) - \tilde{h}_A^f(g)\right|\right\}}. \qquad (2.16)$$

Легко увидеть, что степень четкости, вычисленная по (2.16), удовлетворяет условиям, описанным в (2.14). ИНМ в союзе с изображением A определяют компоненты принадлежности и четкости (2.15) и (2.16) соответственно.

Размытость изображения - это мера серости двусмысленности, связанная с пикселями изображения. Иногда требуется в несколько раз уменьшить размытость, присутствующую в изображении, в целях повышения контрастности между яркой и темной областями. Алгоритм контрастного усиления, основанный на минимизации нечеткости, предлагается в [74]:

$$\mu_{\tilde{A}}(g) = \left(1 + \frac{g_{\max} - g}{F_d}\right)^{-F_e},$$

где g_{max} обозначает максимально серый уровень желаемого, а F_e, F_d являются экспоненциальным и деноминационным фаззификаторами соответственно, которые контролируют неопределенность в нечетких плоскостях.

Фаззификатор F_d определяется как

$$F_d = \frac{g_{\max} - g}{\left(\frac{1}{2}\right)^{\frac{-1}{F_e}} - 1}.$$

Модификация принадлежности значений осуществляется с помощью оператора интенсификации [75] в следующем виде:

$$T_1(\mu_{\tilde{A}}(g)) = \begin{cases} 2(\mu_{\tilde{A}}(g))^2 & \text{åñëè} \quad 0 \le \mu_{\tilde{A}}(g) \le \frac{1}{2}, \\ 1 - 2(1-\mu_{\tilde{A}}(g))^2 & \text{åñëè} \quad \frac{1}{2} \le \mu_{\tilde{A}}(g) \le 1. \end{cases}$$

Оператор интенсификации T последовательно применяется по следующей схеме:

$$T_r(\mu_{\tilde{A}}(g)) = T_1\{T_{r-1}(\mu_{\tilde{A}}(g))\},$$

где r= 1, 2,..., - индекс результата дальнейшего уменьшения размытости изображения. В предельном случае $r \to \infty$ T_r производит двухуровневые (бинарные) изображения.

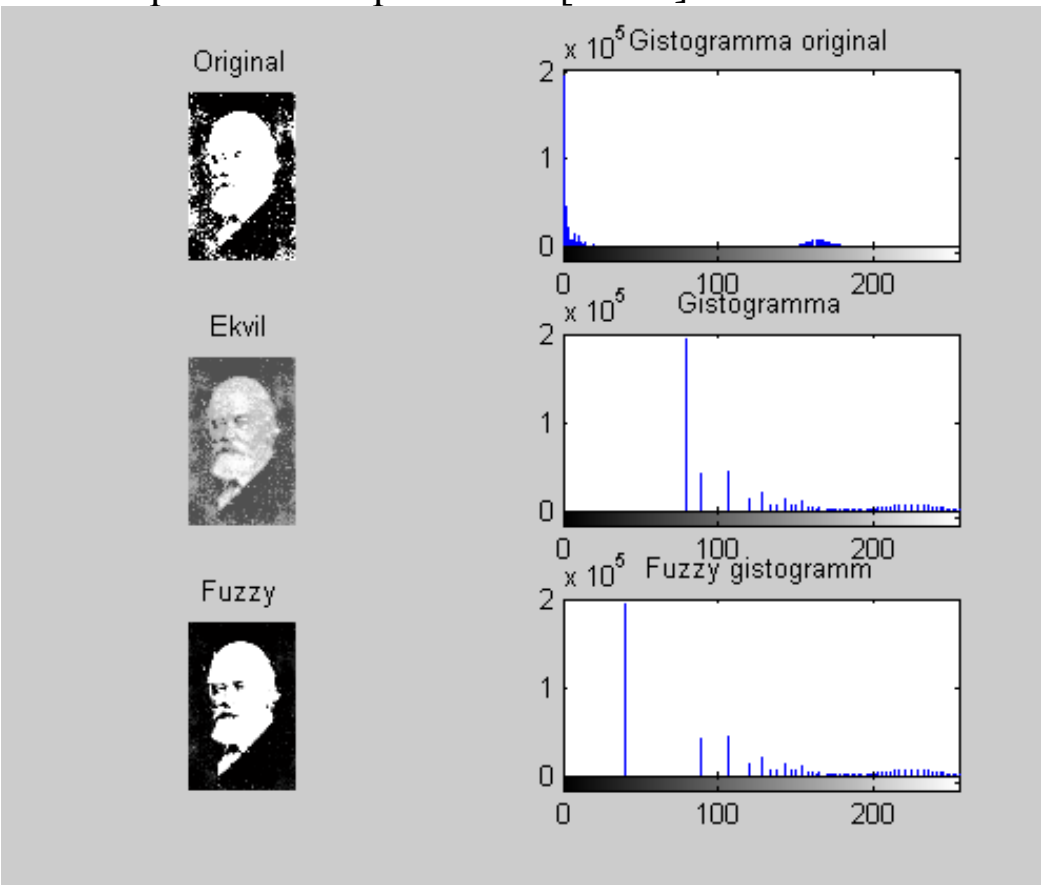
После модификации значений принадлежности дефаззификация выполняется в следующем виде:

$$g' = \begin{cases} 0 & если\ \bar{g}' < 0, \\ \bar{g}' & если\ 0 \le \bar{g}' \le 255, \\ 255 & если\ \bar{g}' > 255, \end{cases}$$

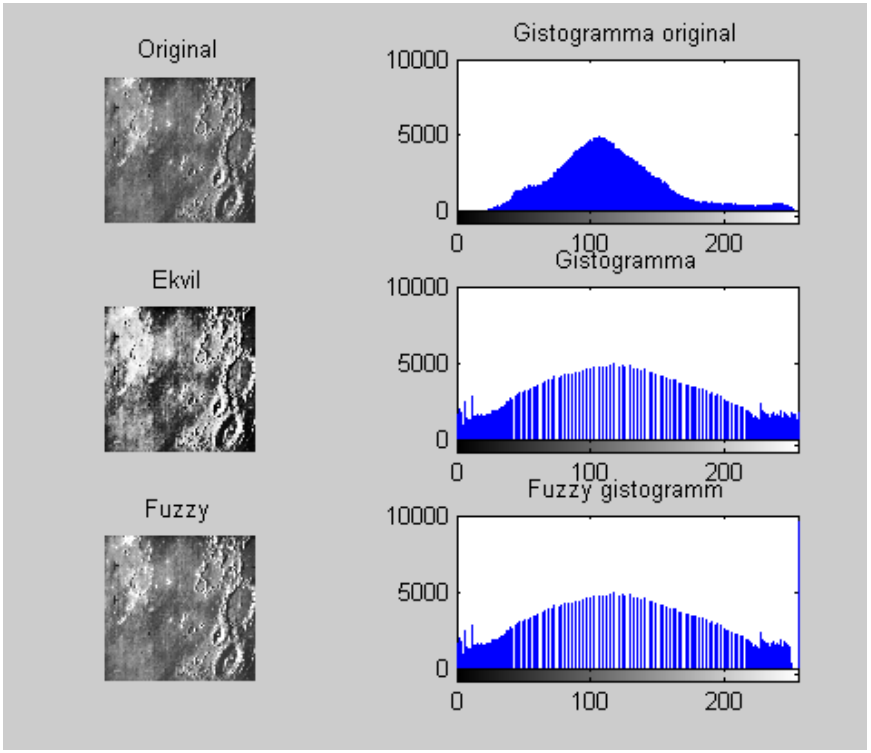
где $g`$ новый серый уровень и $\bar{g}'$ получается из обратной функции принадлежности как

$$\bar{g}' = g_{\max} - F_d\left((\mu'_A(g))^{\frac{-1}{F_e}} - 1 \right).$$

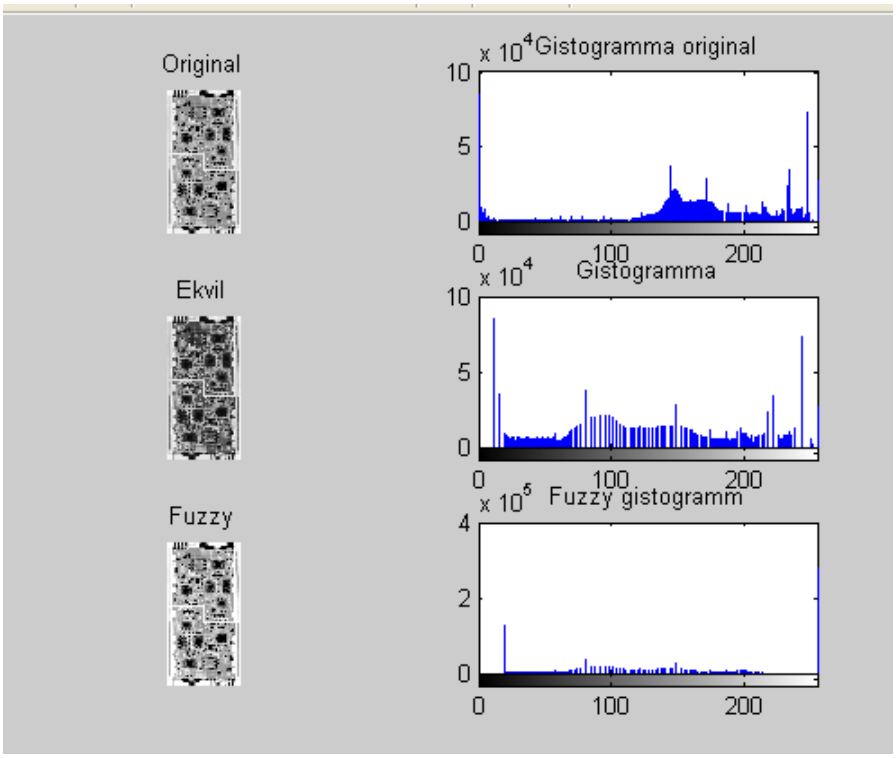
На рис. 2.40 приведены результаты работы рассмотренных алгоритмов на разных изображениях [77-79].



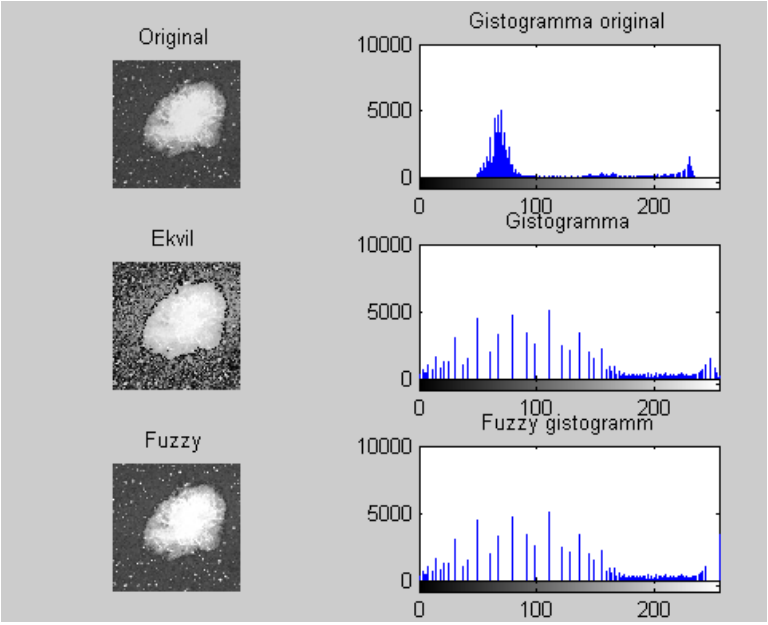
а)


Original
Ekvil
Fuzzy
Gistogramma original
10000
5000
0
0
100
200
Gistogramma
10000
5000
0
0
100
200
Fuzzy gistogramm
10000
5000
0
0
100
200

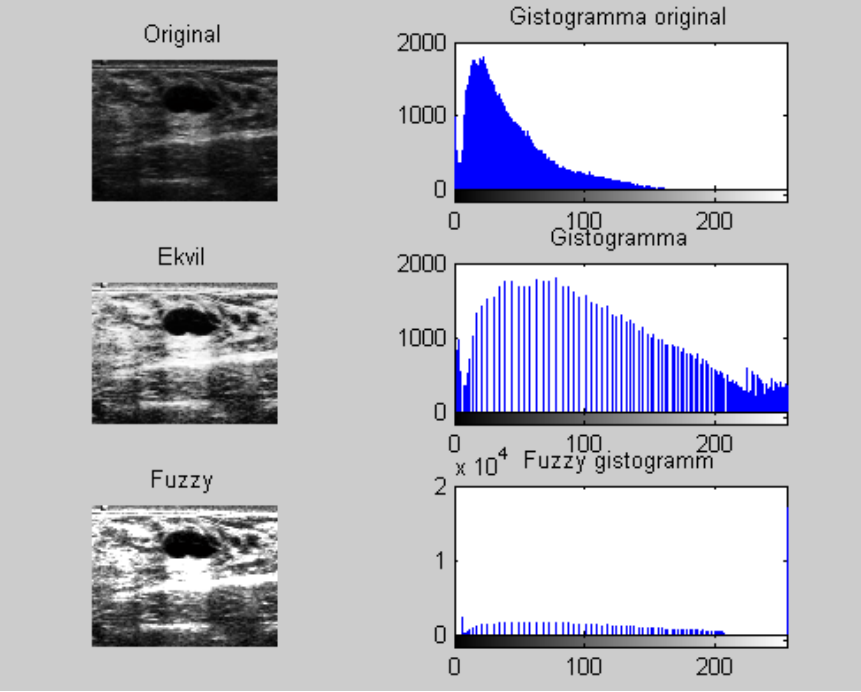
б)


Original
Ekvil
Fuzzy
x 10^4 Gistogramma original
10
5
0
0
100
200
x 10^4
Gistogramma
10
5
0
0
100
200
x 10^5 Fuzzy gistogramm
4
2
0
0
100
200

в)



г)



д)

Рис. 2.40. Результаты работы рассмотренных алгоритмов:
а - на первом изображении; *б* - на втором изображении;
в - на третьем изображении; *г* - на четвертом изображении;
д - на пятом изображении

Таким образом, при обработке изображений требуется по некоторым признакам выделять некоторые однородные области изображения. Этапы предварительной обработки изображения позволяют уменьшить влияние искажений на процесс распознавания. На основе полученных результатов можно заключить, что

предложенный интуитивный нечеткий подход вводит новый потенциал в решение различных задач обработки изображений, так как он предлагает гибкий и адаптируемый способ обработки неопределенности, присутствующей в цифровых изображениях.

2.5.1. Исследование алгоритма нечеткой кластеризации для сегментации изображения

В задачах разработки алгоритмов сегментации изображений возникают некоторые сложности, связанные с определением понятия однородности области. Кроме того, структуризация сложных образов требует учитывать тот факт, что существует множество реальных объектов, не имеющих четких границ по своей природе. В таком случае вопрос о необходимости обеспечения однозначности при сегментации является неадекватным, особенно при необходимости учета незначительных различий или для сегментов сложной формы, перекрывающихся между собой. Поэтому применение существующих методов и алгоритмов не решает задачу сегментации изображений в реальных условиях.

Методы сегментации можно разделить на два класса: автоматические, не требующие взаимодействия с пользователем, и интерактивные, использующие пользовательский ввод непосредственно в процессе работы. Здесь рассматриваем только автоматические методы.

Задачи автоматической сегментации делятся на два класса:

- выделение областей изображения с известными свойствами;
- разбиение изображения на однородные области.

Между этими двумя постановками задачи есть принципиальная разница. В первом случае задача сегментации состоит в поиске определенных областей, о которых имеется априорная информация (например, мы знаем цвет, форму областей, или интересующие нас области представляют собой изображения известного объекта). Методы этой группы узкоспециализированы для каждой конкретной задачи. Сегментация в такой постановке используется в основном в задачах машинного зрения (анализ сцен, поиск объектов на изображении).

Во втором случае никакая априорная информация о свойствах областей не используется, зато на само разбиение изображения накладываются некоторые условия (например, все области должны

быть однородны по цвету и текстуре). Так как при такой постановке задачи сегментации не используется априорная информация об изображенных объектах, то методы этой группы универсальны и применимы к любым изображениям. В основном сегментация в этой постановке применяется на начальном этапе решения задачи, для того чтобы получить представление изображения в более удобном виде для дальнейшей работы.

Задача разбиения изображения на однородные области поставлена некорректно. На рис. 2.41 приведены три варианта сегментации одного и того же изображения [64]. Далеко не всегда для изображения есть единственно «правильная» сегментация, и далеко не всегда задача сегментации имеет единственное решение. По той же причине нет и объективного критерия оценки качества разбиения изображения.

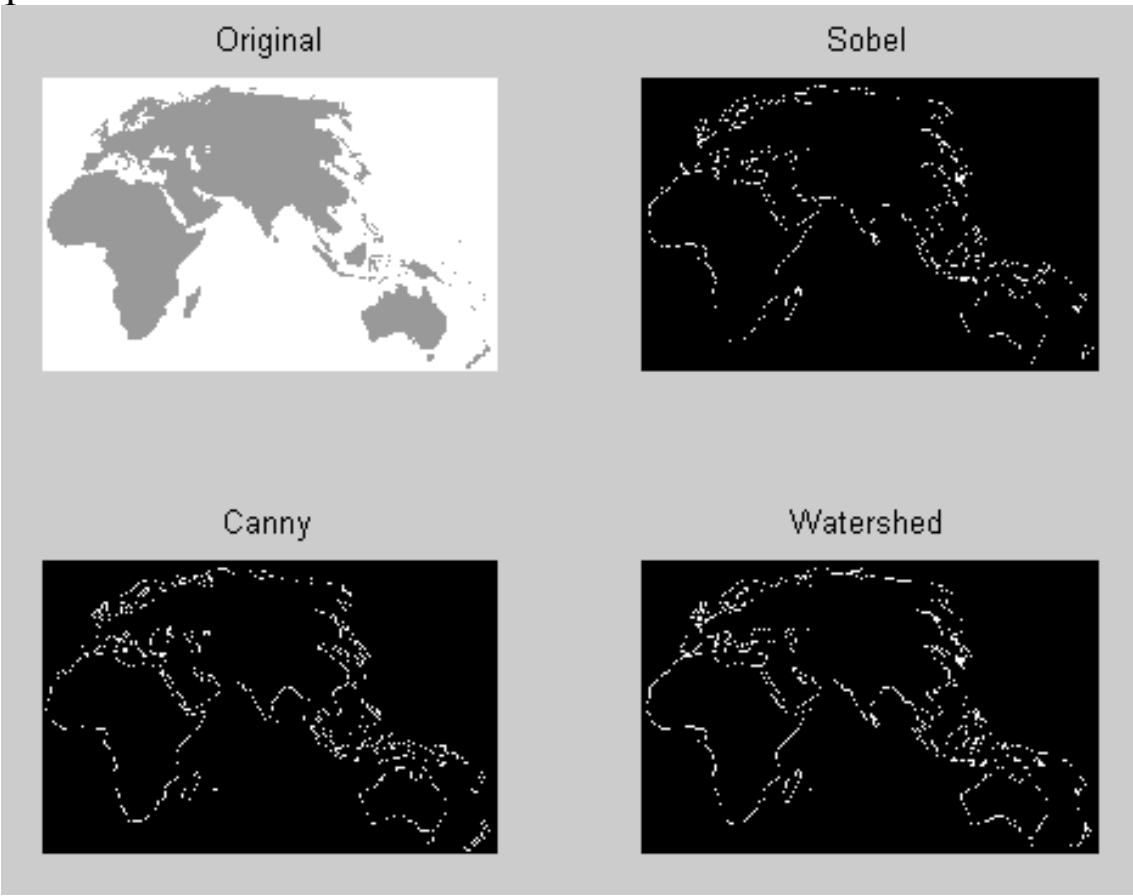


Рис. 2.41. Варианты сегментации изображения

Поскольку сегментация обычно используется не самостоятельно, а как часть некоторой системы (например, системы машинного зрения), то с практической точки зрения качество работы метода оценивается исходя из работы системы в целом. Поэтому

один и тот же метод сегментации может оказаться хорошим для одной задачи и плохим для другой.

Для оценки качества метода в конкретной задаче обычно фиксируют несколько свойств, которыми должна обладать хорошая сегментация.

Качество работы метода оценивается в зависимости от того, насколько полученная сегментация обладает этими свойствами. Наиболее часто используются следующие свойства [61]:

- однородность регионов (однородность цвета или текстуры);
- непохожесть соседних регионов;
- гладкость границы региона;
- маленькое количество мелких «дырок» внутри региона.

Разные методы сегментации ориентированы на разные свойства разбиения. Поэтому при выборе метода сегментации для решения конкретной задачи следует определиться, какие свойства разбиения действительно важны. В некоторых прикладных задачах достаточно того, чтобы разбиение обладало лишь первыми двумя из перечисленных свойств.

Методы выращивания регионов и дробления-слияния учитывают пространственное расположение точек напрямую.

Методы выращивания регионов основаны на следующей идее. Сначала по некоторому правилу выбираются центры регионов (seeds), к которым поэтапно присоединяются соседние точки, удовлетворяющие некоторому критерию. Процесс выращивания регионов (regiongrowing) останавливается, когда ни одна точка изображения не может быть присоединена ни к одному из регионов.

Применяются разные критерии, на основании которых точка присоединяется или не присоединяется к региону: близость (в некотором смысле) точки к центру региона; близость к соседней точке, присоединенной к региону на предыдущем шаге; близость по некоторой статистике региона; стоимость кратчайшего пути от точки до центра региона и т.п.

В основном процедура выращивания региона используется для получения отдельных регионов, однако, применяя эту процедуру последовательно или одновременно для нескольких регионов, можно получить разбиение всего изображения. Существуют различные стратегии выращивания регионов [61].

Методы дробления-слияния состоят из двух основных этапов: дробления и слияния 6[2]. Дробление начинается с некоторого разбиения изображения, не обязательно на однородные области. Процесс дробления областей происходит до тех пор, пока не будет получено разбиение изображения (пересегментация), удовлетворяющее свойству однородности сегментов. Затем происходит объединение схожих соседних сегментов до тех пор, пока не будет получено разбиение изображения на однородные области максимального размера.

Конкретные методы различаются алгоритмами, используемыми на этапах дробления и слияния. Для получения пересегментации изображения используются алгоритмы *k*-средних [1], watershed [62], на втором этапе также используются алгоритмы *k*-средних [61], самоорганизующиеся карты Кохонена [2]. На этапе слияния регионов используются relaxationprocess [63], *k*-средних [61], SIDE-уравнения [62], самоорганизующиеся карты Кохонена [62] и т.д.

При использовании методов, основанных на операторах выделения краев задача сегментации формулируется как задача поиска границ регионов. Методы поиска границ хорошо разработаны для полутоновых изображений. Полутоновое изображение рассматривается как функция двух переменных (*x* и *y*), и предполагается, что границы регионов соответствуют максимумам градиента этой функции. Для их поиска применяется аппарат дифференциальной геометрии (в простейшем случае это фильтры Roberts, Kirsch, Prewitt, Sobel).

Для повышения устойчивости к шуму перед применением фильтрации изображение обычно размывают. Благодаря коммутативности оператора Лапласа и Гауссова фильтра, можно одновременно осуществлять размытие и поиск границ. В методе Canny комбинируются результаты поиска границ при разной степени размытия.

Основной проблемой методов поиска границ является неустойчивость к шуму. Кроме того, поскольку понятие границы свое для каждой задачи, каждый раз при применении методов поиска границ требуется дополнительно выбирать метод доработки результатов фильтрации (edgelinking, edgerelaxation).

2.5.2. Нечеткий алгоритм кластеризации для множества вещественных чисел

Пусть даны вещественное множество $X = \{x_1, x_2, .., x_m\} \subset R$ и k вещественные числа $s_1, s_2, .., s_k$ в возрастающем порядке, а именно:

$$s_1 < s_2, < .. < s_k.$$

Нечеткое разбиение $A_1, A_2, .., A_k$ для множества X может быть организовано с помощью треугольной функции.

Пусть $a \neq b$ - два действительных числа. Открытая треугольная функция имеет вид

$$\mu(x, b; a) = \max\left[\min\left(1, \frac{x-a}{b-a}\right), 0\right]. \qquad (2.17)$$

Если $a<b$, то формула (2.17) определяет правую открытую треугольную функцию, и если $a>b$, то формула (2.17) определяет левую открытую треугольную функцию.

Отметим, что функция μ проверяет следующие четыре свойства:

$$\mu(a, b; a) = 0,$$
$$\mu(b, b; a) = 1,$$
$$\mu\left(\frac{a+b}{2}, b; a\right) = \frac{1}{2},$$
$$\mu(x, b; a) + \mu(x, a; b) = 1. \qquad (2.18)$$

Выражение (2.18) показывает, что функции $\mu(x, b; a)$ и $\mu(x, a; b)$ определяют нечеткую область пространства R.

Пусть $a<b<c$ - три действительных числа. Треугольная функция определяется как

$$t(x, b; a, c) = \mu(x, b; a) \wedge \mu(x, b; c), \qquad (2.19)$$

где $\wedge$ относится к функции "min" или алгебраическому произведению "$\cdot$".

Другими словами, нечеткое множество определяется треугольной функцией принадлежности и представляет пересечение между двумя открытыми нечеткими множествами.

Можно найти и другие формулы вычисления для треугольной функции. Эта функция может иметь следующий вид:

$$\mu(x, b; a) = \alpha \cdot |x - a| + \beta |x - b| + \gamma |x - c|.$$

Параметры функции определяются с помощью значений функций $\mu(a), \mu(b), \mu(c)$ в точках a, b, c. Для этого надо решить систему

$$\begin{cases} \beta|a-b|+\gamma|a-c|=\mu(a,b;a) \\ \alpha|b-a|+\gamma|b-c|=\mu(b,b;a) \\ \alpha|c-a|+\beta|c-b|=\mu(c,b;a) \end{cases}. \quad (2.20)$$

Система (2.20) имеет следующее решение:

$$\begin{cases} \alpha=\frac{1}{2}\cdot\left(\frac{\mu(c,b;a)+\mu(a,b;a)}{c-a}+\frac{\mu(b,b;a)-\mu(a,b;a)}{b-a}\right) \\ \beta=\frac{1}{2}\cdot\left(\frac{\mu(c,b;a)-\mu(b,b;a)}{c-b}+\frac{\mu(b,b;a)-\mu(a,b;a)}{b-a}\right). \\ \gamma=\frac{1}{2}\cdot\left(\frac{\mu(c,b;a)+\mu(a,b;a)}{c-a}-\frac{\mu(c,b;a)-\mu(b,b;a)}{c-b}\right) \end{cases}$$

Для конкретных значений функции $\mu(a)=0,\ \mu(b)=1,\ \mu(c)=0$ решение имеет вид

$$\begin{cases} \alpha=\frac{1}{2}\cdot\frac{1}{b-a} \\ \beta=\frac{1}{2}\cdot\left(\frac{1}{b-a}-\frac{1}{c-b}\right) \\ \gamma=\frac{1}{2}\cdot\frac{1}{c-b} \end{cases}$$

и

$$\mu(x,b;a)=\frac{1}{2}\cdot\frac{|x-a|}{b-a}+\frac{1}{2}\cdot\left(\frac{1}{b-a}-\frac{1}{c-b}\right)\cdot|x-b|+\frac{1}{2}\cdot\frac{|x-c|}{c-b},$$

или

$$\mu(x,b;a)=\frac{1}{2}\cdot\frac{|x-a|-|x-b|}{b-a}+\frac{1}{2}\cdot\frac{|x-c|-|x-b|}{c-b}. \quad (2.21)$$

Точно так же мы можем получить эквивалентную формулу для открытой треугольной функции. Для двух действительных чисел $a\neq b$ функция

$$\mu(x,b;a)=\frac{1}{2}\cdot\frac{|x-a|-|x-b|}{b-a}+\frac{1}{2} \quad (2.22)$$

определяет открытую треугольную функцию. С практической точки зрения мы можем вместо формулы (2.22) использовать следующую:

$$\mu(x,b;a)=\frac{1}{2}\cdot\frac{|x-a|-|x-b|+\varepsilon}{|b-a|+\varepsilon}+\frac{1}{2}, \quad (2.23)$$

где

$$1>>\varepsilon>0. \quad (2.24)$$

Более обобщенно нечеткое разбиение $A_1,A_2,..,A_k$ для множества X может быть организовано с помощью L-R функции.

Нечеткое число $\tilde{x}$ называется нечетким числом $L-R$ типа, если

$$\mu(\mathrm{x}, \mathrm{x}_R(\alpha), \mathrm{x}_L(\alpha)) = \begin{cases} \mu_L(x) = 1 - \dfrac{x - x_L(\alpha)}{u_L}, \\ \mu_R(x) = 1 - \dfrac{x_R(\alpha) - x}{x_R}, \end{cases}$$

где x - четкое значение числа $\tilde{x}$, т.е. $x = x_L(1) = x_R(1)$; x_L и x_R - соответственно левое и правое растяжения нечеткого числа $\tilde{x}$; $x_L(\alpha)$ и $x_R(\alpha)$ - соответственно левое и правое значения нечеткого числа $\tilde{u}$ четкости α.

Из определения следует, что если

$$\tilde{x}(\alpha) = \{x, x_L(\alpha), x_R(\alpha)\},$$

то

$$x_L(\alpha) = x - (1-\alpha)x_L;\ x_R(\alpha) = x + (1-\alpha)x_R.$$

Рассмотрим алгебраическое действие над нечеткими числами $L-R$ типа.

Сложение:

$$\tilde{u} + \tilde{v} = \{u + v - (1-\alpha)(u_L + v_L); u + v + (1-\alpha)(u_R + v_R)\}.$$

Вычитание:

$$\tilde{u} - \tilde{v} = \{u - v - (1-\alpha)(u_L + v_L); u - v + (1-\alpha)(u_R + v_R)\}.$$

Умножение:

1) для $u > 0; v > 0$

$$\tilde{u} \cdot \tilde{v} = \{u \cdot v; (1-\alpha)(uv_L + vu_L) - (1-\alpha)u_L v_L;\ (1-\alpha)(uv_R + vu_R) + (1-\alpha)u_R v_L\};$$

2) для $u > 0; v < 0$

$$\tilde{u} \cdot \tilde{v} = \{u \cdot v; (1-\alpha)(uv_R + vu_L) - (1-\alpha)u_L v_L;\ (1-\alpha)(uv_L + vu_R) + (1-\alpha)u_R v_L\};$$

3) для $u < 0; v < 0$

$$\tilde{u} \cdot \tilde{v} = \{u \cdot v; (1-\alpha)(uv_R + vu_L) - (1-\alpha)u_R v_R;\ (1-\alpha)(uv_L + vu_L) - (1-\alpha)u_L v_L\}.$$

Деление:

$$\frac{\tilde{u}}{\tilde{v}} = \tilde{u} \cdot \frac{1}{\tilde{v}}.$$

При разработке алгоритма нечеткой кластеризации для сегментации изображения, во-первых, мы рассматриваем функции принадлежности:

$$\mu_1(x) = \mu(x, s_1; s_2),$$

для $i = 2,3, ..., k-1$

$$\mu_i(x) = \mu(x, s_i; s_{i-1}) \wedge \mu(x, s_i; s_{i+1}),$$

$$\mu_k(x) = \mu(x, s_k; s_{k-1}).$$

Функции $\mu_1, \mu_2, .., \mu_k$ определяют раздел разбиения по следующему равенству:

$$\mu_1 + \mu_2 + ... + \mu_k = 1.$$

Во-вторых, определяется оператор дефаззификации $\tau(\mu, \gamma)$, который применяется к нечеткому разбиению:

$$\tau(\mu, \gamma) = (\tau_1, \tau_2, .., \tau_k),$$

где

$$\tau(x, \mu, \gamma) = \frac{\mu_i^{\gamma}(x)}{\sum_{j=1}^{k} \mu_i^{\gamma}(x)}. \qquad (2.25)$$

Пусть v_i и F_i определяются следующим образом:

$$v_i(x) = \mu_i(x) \wedge \tau_i(x, \mu; \gamma), \qquad (2.26)$$

$$F_i(s_i) = \frac{\sum_{j=1}^{m} v_i(x_j) \cdot x_j}{\sum_{j=1}^{m} v_i(x_j)}. \qquad (2.27)$$

Рассмотрим теперь следующее ограничение для параметров $s_1, s_2, .., s_k$:

$$s_i = F_i(s_i). \qquad (2.28)$$

Нечеткое множество A_i определяется уравнением (2.28). Центр s_i принадлежит выпуклой оболочке множества X, и это есть фиксированная точка для функции F_i.

Результаты получаются с применением следующего нечеткого алгоритма кластеризации.

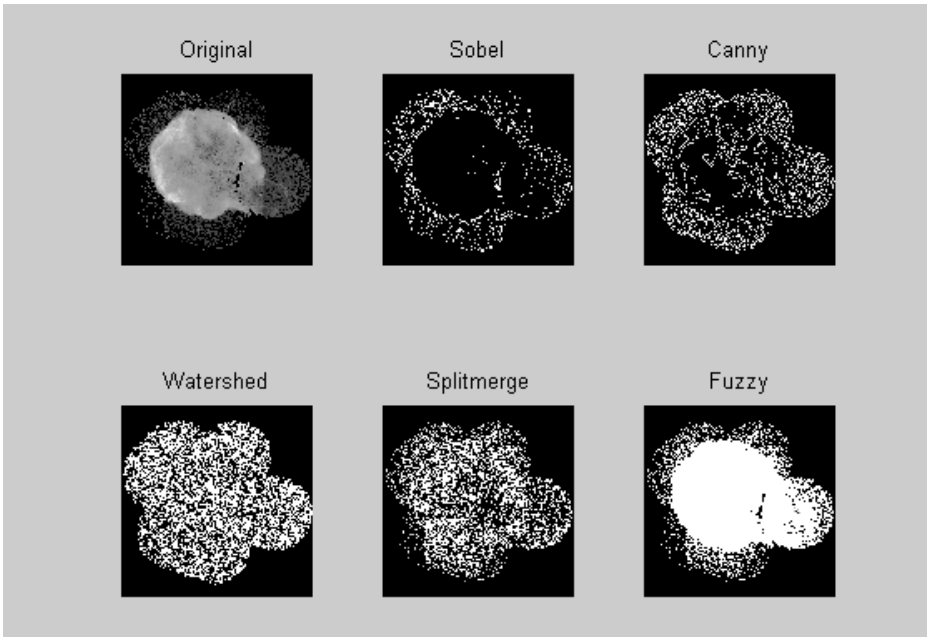
Шаг 1. Инициализируются количество кластеров k, параметр дефаззификации γ, процедура остановки параметра δ, индекс итерации $l = 0$ и центры кластеров. Далее, вычисляются нечеткие функции принадлежности $\mu_1^{(0)}, \mu_2^{(0)}, .., \mu_k^{(0)}$ и компоненты дефаззификации функции $\tau_1^{(0)}, \tau_2^{(0)}, .., \tau_k^{(0)}$.

Шаг 2. Индекс итерации увеличивается, т.е. $l \to l+1$. Мы рассчитываем центры кластера $s_1^{(l)} = F_1(s_1^{(l-1)}), s_2^{(l)} = F_2(s_2^{(l-1)}), ..., s_k^{(l)} = F_k(s_k^{(l-1)})$, нечеткие функции принадлежности $\mu_1^{(l)}, \mu_2^{(l)}, .., \mu_k^{(l)}$, компоненты дефаззификации $\tau_1^{(l)}, \tau_2^{(l)}, .., \tau_k^{(l)}$ и функции $v_1^{(l)}, v_2^{(l)}, .., v_k^{(l)}$.

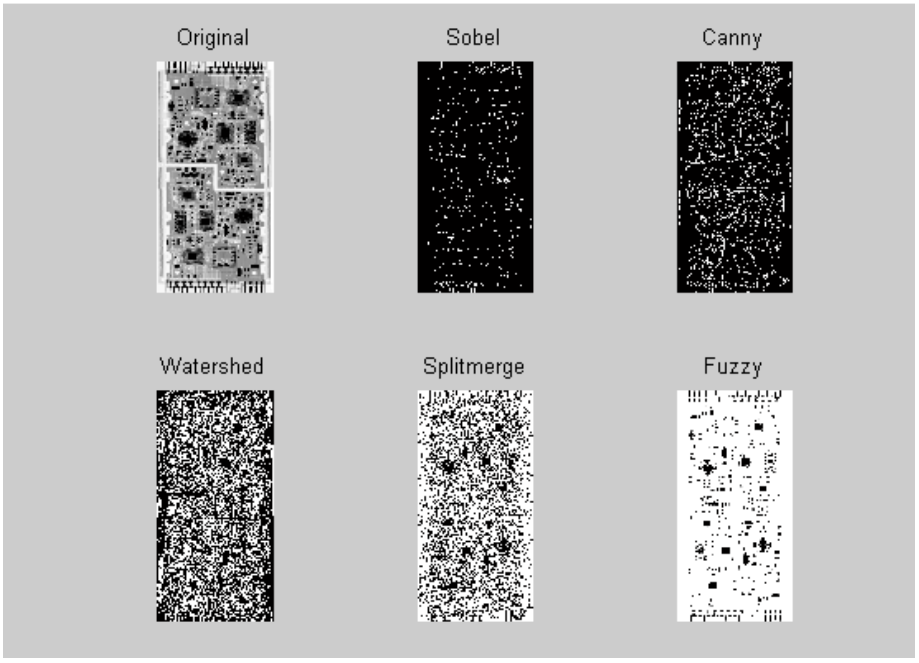
Шаг 3. Вычисляем $d = \sum_{i=1}^{k} \left| \mu_i^{(l)} - \mu_i^{(l-1)} \right|$. Если $d > \delta$, то вернемся к шагу 2, в противном случае - перейти к шагу 4.

Шаг 4. Сохранение данных и конец.

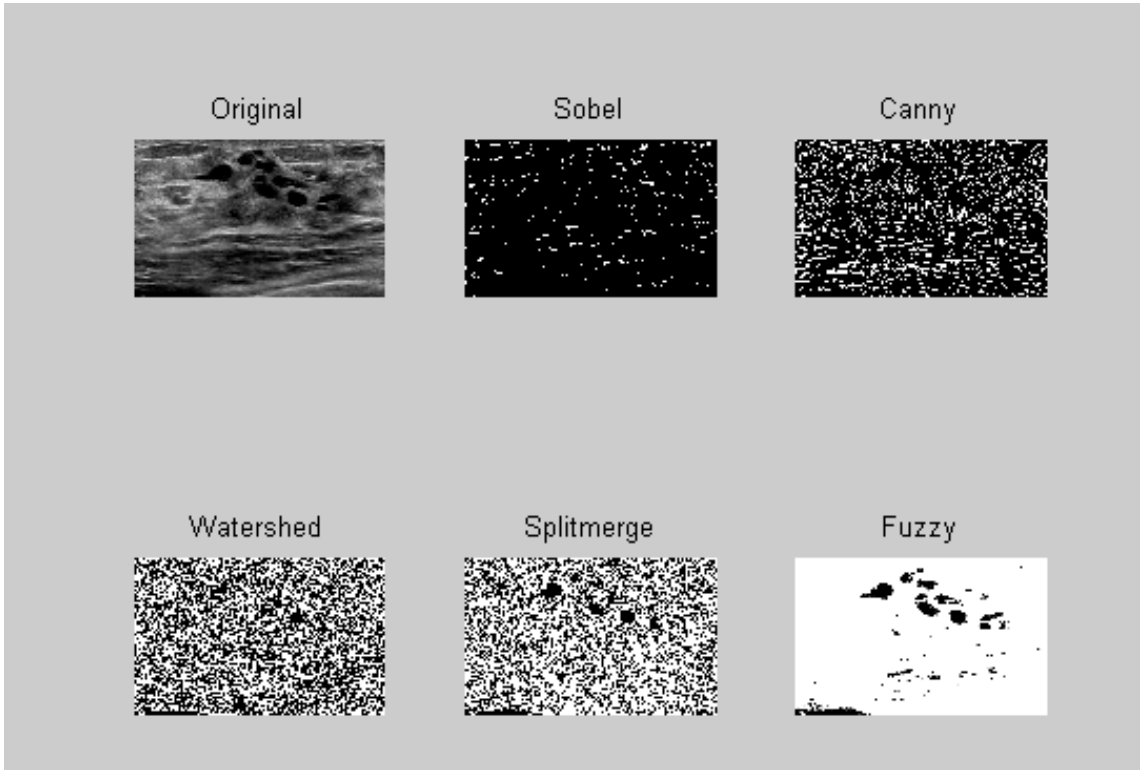
На рис. 2.42 приведены результаты работы рассмотренных алгоритмов на разных изображениях [64].



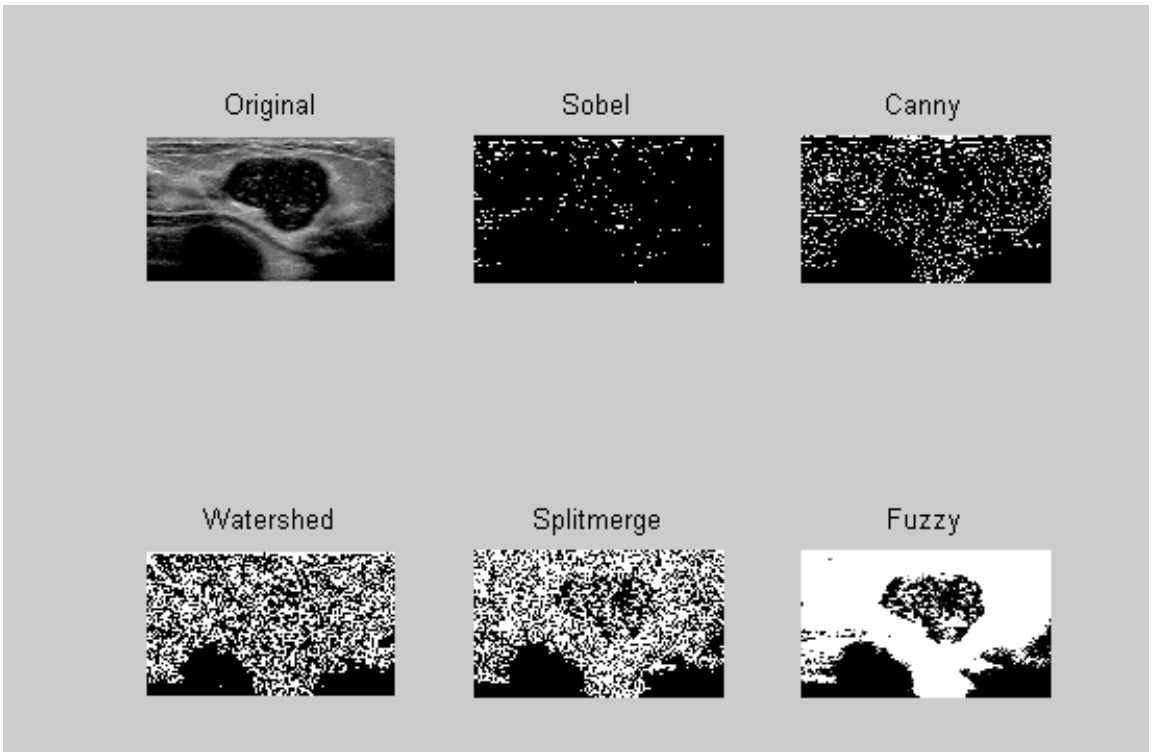
а)



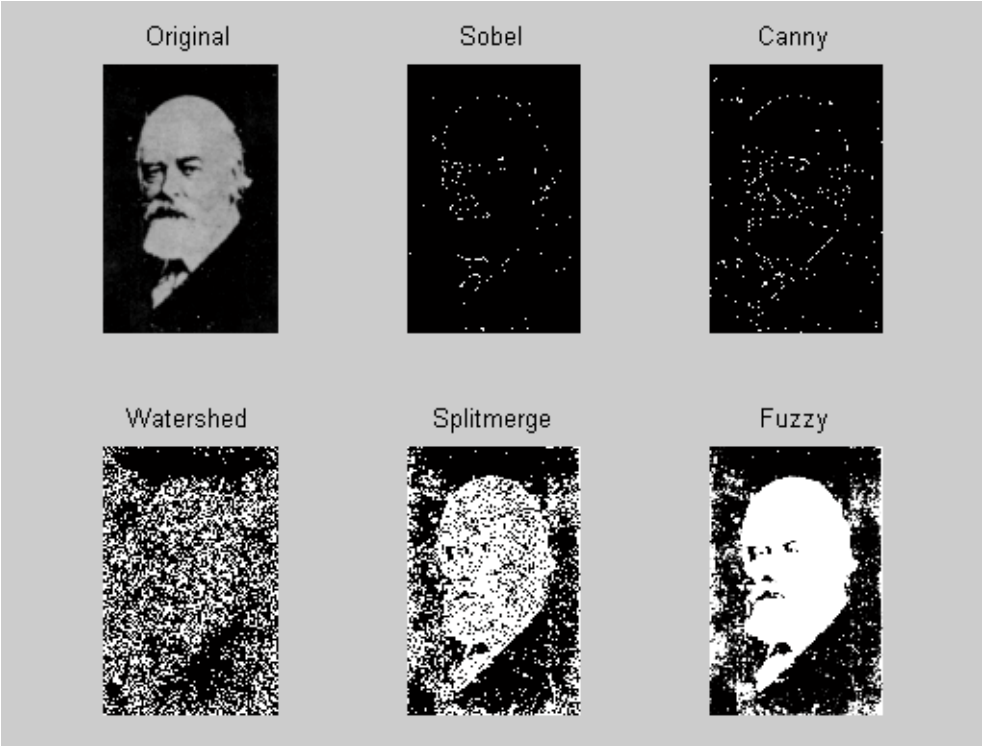
б)


Original
Sobel
Canny
Watershed
Splitmerge
Fuzzy

в)


Original
Sobel
Canny
Watershed
Splitmerge
Fuzzy

г)



д)

Рис. 2.42. Результаты работы рассмотренных алгоритмов на:
а - первом; *б* - втором; *в* - третьем; *г* – четвертом;
д - пятом изображениях

В работе представлены нечеткие алгоритмы кластеризации для пространства *R*, исследуется метод сегментации изображений, которые используют алгоритмы кластеризации. Достоинствами такого подхода являются:

- широкий спектр применения. За счёт выбора отображений и меры можно приспособить алгоритм под различные задачи;
- гибкость. Изменяя порог и меру, можно эффективно менять чувствительность алгоритма;
- скорость. Алгоритм работает существенно быстрей алгоритма с поиском границы метода;
- устойчивость. Алгоритм более устойчив к ошибкам, чем методы, основанные на нахождении границ, так как при ошибке мы теряем не весь регион, а лишь его небольшую часть.

2.5.3. Обнаружение объекта на основе цвета

Существует большое количество задач, одной из составляющих которых является вопрос автоматического распознавания лиц на изображениях. Подходов к решению этого вопроса существует много, однако в рамках данного материала рассмотрим метод автоматического обнаружения лиц на изображениях на основе анализа цвета.

Считаем некоторое исходное изображение (рис. 2.43 - 2.46).

Рис.2.43. 1-е исходное изображение

Рис.2.44. 2-е исходное изображение

Рис.2.45. 3-е исходное изображение

Рис.2.46. 4-е исходное изображение

Далее необходимо на изображении лица выбрать пиксели с характерным цветом, определить среднее значение их интенсивности и среднеквадратичное отклонение.

На основе знаний о значениях интенсивностей пикселей лица и их возможных вариациях проводится сегментация изображения (рис. 2.47 – 2.50).

Рис. 2.47. 1-е исходное изображение после сегментации

Рис. 2.48. 2-е исходное изображение после сегментации

Рис. 2.49. 3-е исходное изображение после сегментации

Рис. 2.50. 4-е исходное изображение после сегментации

Целью сегментации является выделение лица на изображении. Однако, вполне естественно, что на изображении присутствовали и другие объекты, значения интенсивностей пикселей которых совпали

с интенсивностью пикселей лица. В результате на сегментированном изображении кроме лица выделились и другие объекты. Теперь на сегментированном изображении предстоит найти изображение нужного объекта, т.е. лица. Критерии поиска могут быть разными. Это может быть площадь, форма и др. В данном примере в качестве критерия для поиска лица выберем площадь. Исходя из выбранного ранее способа сегментации, можно предположить, что лицо занимает наибольшую площадь. Поэтому выбор критерия для поиска лица на основе площади позволит удалить другие объекты, которые меньше за площадью.

Рис.2.51- Исходное изображение после удаления шума

Рис.2.52- Исходное изображение после удаления шума

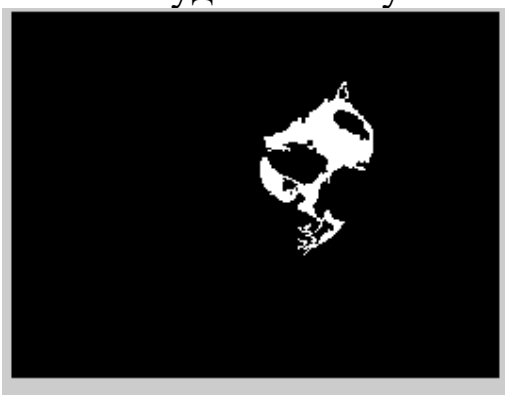


Рис.2.53- Исходное изображение после удаления шума

Рис.2.54- Исходное изображение после удаления шума

Удалим также и другие объекты на сегментированном изображении лица.

Рис.2.55- Исходное изображение после удаления шума небольших объектов

Рис.2.56- Исходное изображение после удаления шума небольших объектов

Рис.2.57- Исходное изображение после удаления шума небольших объектов

Рис.2.58- Исходное изображение после удаления шума небольших объектов

Здесь следует отметить, что выбранного нами критерия может быть недостаточно для достоверного определения объекта поиска, т.е. изображения лица. Такое может произойти в том случае, если на изображении присутствуют другие объекты с похожим цветом, лицо может быть в тени, быть под наклоном и т.п. Поэтому необходимо использовать также другие критерии для поиска изображения лица. Такие критерии могут базироваться на априорной информации о форме лица.

Таким образом, для повышения достоверности распознавания критерий поиска лица на изображении должен быть комплексным.

Далее проводим выделение выбранного лица, например, прямоугольником.

Рис.2.59. Выделение выбранного 1- лица, прямоугольником

Рис.2.60. Выделение выбранного 2- лица, прямоугольником

Рис.2.61. Выделение выбранного 3- лица, прямоугольником

Рис.2.62. Выделение выбранного 4 -лица, прямоугольником

Если же на изображении присутствует несколько лиц одновременно, то рассмотренный выше метод необходимо модифицировать. Рассмотрим это более детально. Первые несколько шагов (считывание исходного изображения, выбор характерных пикселей, сегментация и фильтрация) аналогичны, как и в предыдущем методе, поэтому мы приведем только результаты выполнения этих операций.

Глава 3. ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

3.1. Общие понятия искусственных нейронных сетей

Искусственная нейронная сеть является некоторой моделью естественной нейронной сети. Каждый элемент искусственной сети (нейрон) является прототипом, имитирующим свойства и работу биологического нейрона [61].

Рассмотрим работу искусственной однослойной нейронной сети (рис. 3.1). На **синапсы** (однонаправленные входы) искусственного нейрона поступает некоторое множество сигналов. Каждый сигнал умножается на соответствующий вес, характеризующий синаптическую силу. Входные сигналы после синапсов поступают в ячейки нейронов (блоки суммирования), выходом которых являются **аксоны**, с которых сигналы (возбуждения или торможения), определяющие ***уровень активации нейрона***, поступают на синапсы следующих нейронов (если сеть многослойная) или на основе которых принимается решение (например, пороговым мажоритарным блоком). Хотя сетевые парадигмы весьма разнообразны, в их основе почти всегда лежит именно эта конфигурация.

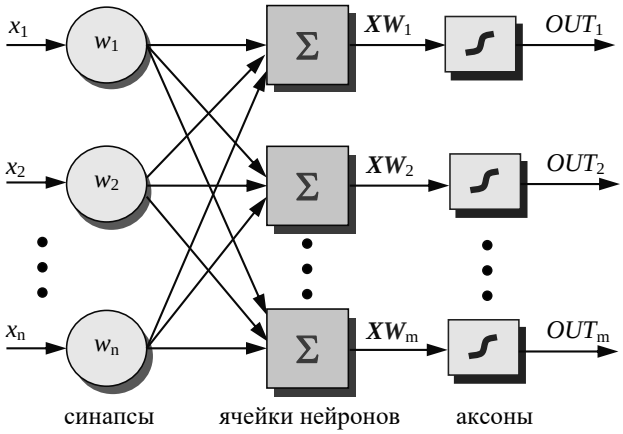


Рис. 3.1. Искусственная однослойная нейронная сеть

Отметим, что синапсы служат лишь для распределения входных сигналов, а аксоны лишь для выдачи сигналов, поэтому они не считаются слоем (слоем являются ячейки нейронов, выполняющие вычисления с комбинациями входных сигналов). В искусственных и биологических сетях многие соединения могут отсутствовать, все

соединения показаны в целях общности. Могут иметь место также соединения между выходами и входами элементов в одном слое.

Условно эту часть нейронной сети может представить искусственным нейроном (рис. 3.2). Эта простая модель искусственного нейрона игнорирует многие свойства своего биологического двойника. Например, она не принимает во внимание задержки во времени, которые воздействуют на динамику системы. Входные сигналы сразу же порождают выходной сигнал. И, что более важно, она не учитывает воздействий функции частотной модуляции или синхронизирующей функции биологического нейрона, которые ряд исследователей считают решающими.

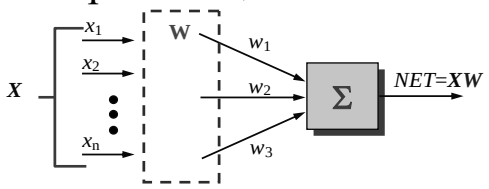


Рис. 3.2. Входная часть искусственного нейрона

Сигнал *NET* далее, как правило, преобразуется некоторой активационной функцией *F* и дает выходной нейронный сигнал *OUT*.

Активационная функция – *это нелинейная усилительная характеристика искусственного нейрона, определяющая его уровень активации.*

Коэффициент усиления активационной функции вычисляется как отношение приращения величины *OUT* к вызвавшему его небольшому приращению величины *NET* – $\frac{\Delta OUT}{\Delta NET}$. Он выражается наклоном кривой при определенном уровне возбуждения и изменяется от малых значений при больших отрицательных возбуждениях (кривая почти горизонтальна) до максимального значения при нулевом возбуждении и снова уменьшается, когда возбуждение становится большим положительным.

Если активационная функция *F* сужает диапазон изменения величины *NET* так, что при любых значениях *NET* значения *OUT* принадлежат некоторому конечному интервалу, то *F* называется *«сжимающей» функцией*.

Качество работы искусственной нейронной сети во многом зависит от весов синаптических связей и от типа применяемых активационных функций. Рассмотрим типы активационных функций.

3.2. Типы активационных функций

Различают следующие виды активационных функций:

- единичная функция;
- линейный порог (функция гистерезиса);
- сигмоидальная функция (гиперболический тангенс).

Единичная функция. Данная функция является самой простой, ее график представлен на рис. 3.3.

Для этой функции $OUT = 1$, если $NET = \boldsymbol{XW}$ больше некоторой величины T и $OUT = 0$ в остальных случаях, где T – некоторая постоянная пороговая величина.

Если в однослойной нейронной сети применяется единичная активационная функция, то такая сеть называется персептроном [62, 63, 64]. Первое систематическое изучение персептронов было предпринято Маккалокком и Питтсом в 1943 г. [62]. В 60-е годы персептроны вызвали большой интерес и оптимизм. Розенблатт доказал замечательную теорему об обучении персептронов [64]. Далее первоначальная эйфория сменилась разочарованием, когда оказалось, что персептроны не способны обучиться решению ряда простых задач. Минский [63] строго проанализировал эту проблему и показал, что имеются жесткие ограничения на то, что могут выполнять однослойные персептроны, и, следовательно, на то, чему они могут обучаться.

Функция линейного порога. Отличием от предыдущей функции является не скачкообразное изменение выхода *OUT*, а в виде линейного тренда. График функции представлен на рис. 3.4.

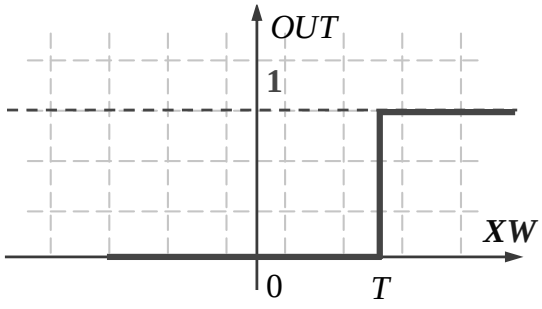


Рис. 3.3. График единичной активационной функции

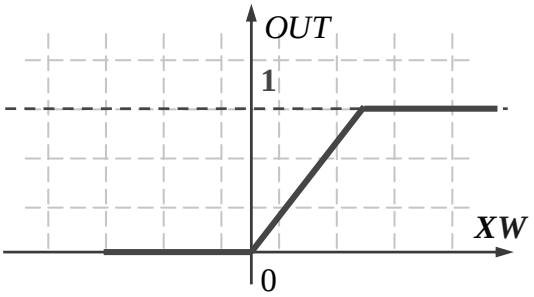


Рис. 3.4. График линейной активационной функции

Сигмоидальная функция. В качестве «сжимающей» функции часто используется логистическая или «сигмоидальная» (S-образная) функция. Эта функция математически выражается как $F(x) = 1/(1+e^{-x})$. Таким образом,

$$OUT = 1/1 + e^{-NET}. \quad (3.1)$$

Данная функция позволяет обрабатывать нейронной сети как слабые, так и сильные сигналы. Слабые сигналы нуждаются в большом сетевом усилении, чтобы дать пригодный к использованию выходной сигнал. Однако усилительные каскады с большими коэффициентами усиления могут привести к насыщению выхода шумами усилителей (случайными флуктуациями), которые присутствуют в любой физически реализованной сети. Сильные входные сигналы в свою очередь также будут приводить к насыщению усилительных каскадов, исключая возможность полезного использования выхода. Сигмоидальная функция решает эту проблему. Впервые это обнаружил Гроссберг в 1973 году.

В качестве сигмоидальных функций может выступать как логистическая функция (3.1), так и функция гиперболического тангенса *OUT* = th(*NET*) (рис. 3.5). Отличием логистической функции от функции гиперболического тангенса является диапазон выходных значений. У первой он находится в диапазоне (0; 1), а у второй в диапазоне (–1; 1).

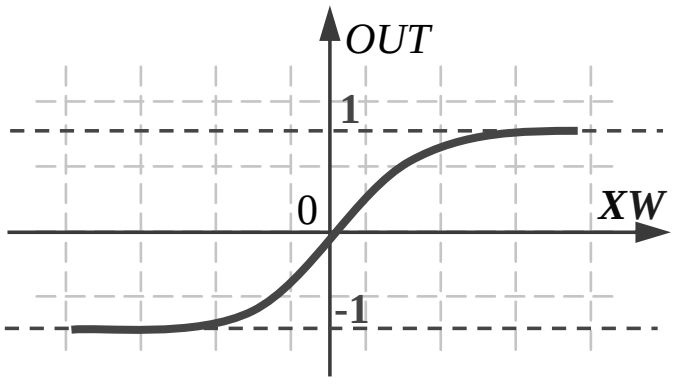


Рис. 3.5. График сигмоидальной активационной функции

3.3. Проблема функции «исключающее или»

Один из самых пессимистических результатов Минского [63] показывает, что однослойный персептрон не может воспроизвести такую простую булевскую функцию, как «*исключающее или*». Это функция от двух аргументов, каждый из которых может быть нулем или единицей. Она принимает значение единицы, когда один из аргументов равен единице (но не оба). Проблему можно проиллюстрировать с помощью однослойной однонейронной системы с двумя входами, показанной на рис. 3.6. Обозначим один вход через *x*, а другой через *y*, тогда все их возможные комбинации будут состоять из четырех точек на плоскости XOY, как показано на рис. 3.7. Например, точка $x = 0$ и $y = 0$ обозначена на рисунке как точка A_0. Таблица 3.1 показывает требуемую связь между входами и выходом, где входные комбинации, которые должны давать нулевой выход, помечены A_0 и A_1, а единичный выход – B_0 и B_1.

В сети на рис. 3.6 активационная функция *F* является обычным порогом, так что *OUT* принимает значение ноль, когда *NET* меньше 0.5, и единица в случае, когда *NET* больше или равно 0.5. Персептрон выполняет следующее вычисление:

$$NET = xw_1 + yw_3. \qquad (3.2)$$

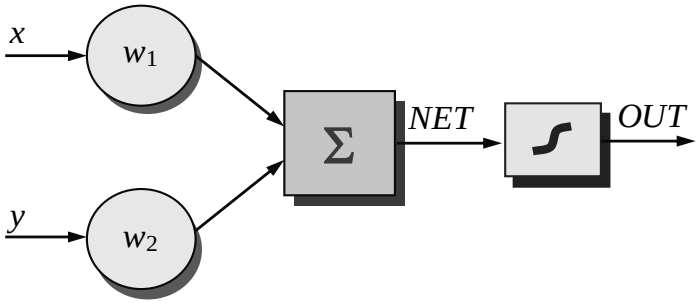


Рис. 3.6. Однонейронная система

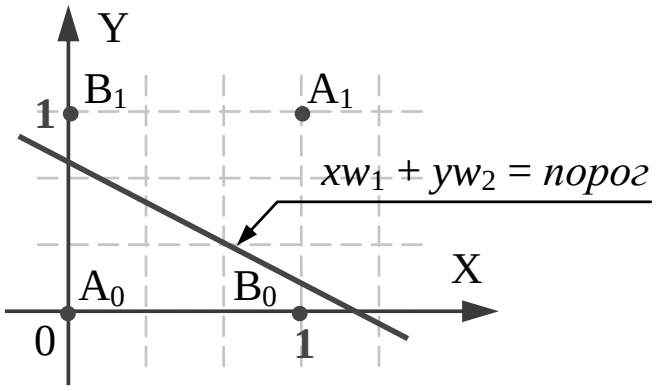


Рис. 3.7. Проблема «исключающее или»

Никакая комбинация значений двух весов не может дать соотношения между входом и выходом, задаваемого табл. 3.1 (таблица истинности для булевой функции). Чтобы понять это ограничение, зафиксируем *NET* для (3.2) на величине порога 0.5. Персептрон в этом случае описывается уравнением (3.3). Это уравнение линейно по *x* и *y,* т.е. все значения по *x* и *y,* удовлетворяющие этому уравнению, будут лежать на некоторой прямой в плоскости XOY

$$xw_1 + yw_2 = 0.5. \qquad (3.3)$$

Любые входные значения для *x* и *y* на этой линии будут давать пороговое значение 0.5 для *NET*. Входные значения с одной стороны прямой обеспечат значения *NET* больше порога, следовательно, *OUT* = 1. Входные значения по другую сторону прямой обеспечат значения *NET* меньше порогового значения, делая *OUT* равным 0. Изменения значений w_1, w_2 и порога будут менять наклон и положение прямой. Для того чтобы сеть реализовала функцию «*исключающее или*», заданную табл. 3.1, нужно расположить прямую так, чтобы точки A были с одной стороны прямой, а точки B – с другой. Попытавшись нарисовать такую прямую на рис. 3.7, убеждаемся, что это невозможно. Это означает, что какие бы значения ни приписывались весам и порогу, сеть неспособна воспроизвести соотношение между входом и выходом, требуемое для представления функции «*исключающее или*».

Таблица 3.1

Таблица истинности для функции «*исключающее или*»

Точки	**Значения *x***	**Значения *y***	**Требуемый выход**
A_0	0	0	0
B_0	1	0	1
B_1	0	1	1
A_1	1	1	0

Невозможность нарисовать прямую линию, разделяющую плоскость XOY так, чтобы реализовывалась функция «*исключающее или*», является не единственным случаем. Имеется обширный класс функций, не реализуемых однослойной нейронной сетью. Об этих функциях говорят, что они являются линейно неразделимыми, и они

накладывают определенные ограничения на возможности однослойных сетей.

Линейная разделимость ограничивает однослойные нейронные сети задачами классификации, в которых множества точек (соответствующих входным значениям) могут быть разделены геометрически. И поскольку линейная разделимость ограничивает возможности персептронного представления, то важно знать, является ли данная функция разделимой. К сожалению, не существует простого способа определить это, если число переменных велико.

Нейрон с n двоичными входами может иметь 2^n различных входных образов, состоящих из нулей и единиц. Так как каждый входной образ может соответствовать двум различным бинарным выходам (единица и ноль), то всего имеется 2^{2n} функций от n переменных (см. зависимость в таблице 3.2) [65].

Как видно из табл. 3.2, вероятность того, что случайно выбранная функция окажется линейно разделимой, весьма мала даже для умеренного числа переменных. По этой причине однослойные персептроны на практике ограничены простыми задачами.

Таблица 3.2

Линейно разделимые функции

n	**2^{2n}**	**Число линейно разделимых функций**
1	4	4
2	16	14
3	256	104
4	65536	1882
5	4294967296	94572

3.4. Многослойные нейронные сети

К концу 60-х годов проблема линейной разделимости была хорошо понята. К тому же было известно, что это серьезное ограничение представляемости однослойными сетями можно преодолеть, добавив дополнительные слои.

Рассмотрим простую двухслойную сеть с двумя входами, подведенными к двум нейронам первого слоя, соединенными с единственным нейроном в слое 2 (рис. 3.8). Пусть порог выходного нейрона равен 0,75, а оба его веса равны 0,5. В этом случае для того,

чтобы порог был превышен и на выходе появилась единица, требуется, чтобы оба нейрона первого уровня на выходе имели единицу. Таким образом, выходной нейрон реализует логическую функцию «*и*».

Каждый нейрон слоя 1 разбивает плоскость XOY на две полуплоскости, один обеспечивает единичный выход для входов ниже верхней линии, другой – для входов выше нижней линии. На рис. 3.9 показан результат такого двойного разбиения, где выходной сигнал нейрона второго слоя равен единице только внутри V-образной области. Аналогично во втором слое может быть использовано три нейрона с дальнейшим разбиением плоскости и созданием области треугольной формы. Включением достаточного числа нейронов во входной слой может быть образован выпуклый многоугольник любой желаемой формы. Так как они образованы с помощью операции «*и*» над областями, задаваемыми линиями, то все такие многогранники выпуклы, следовательно, только выпуклые области и возникают. Точки, не составляющие выпуклой области, не могут быть отделены от других точек плоскости двухслойной сетью.

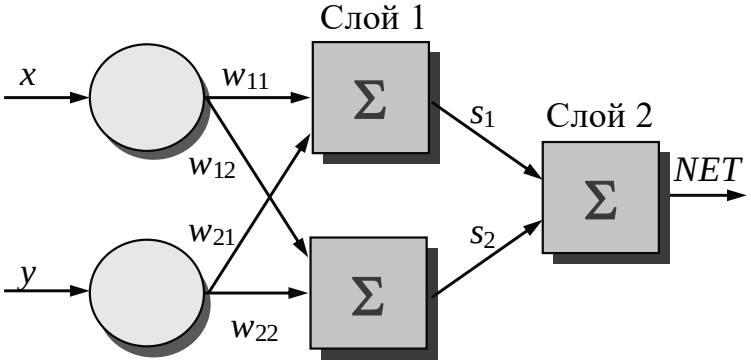


Рис. 3.8. Простейшая двухслойная

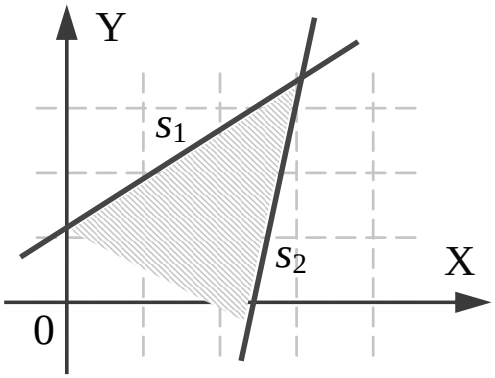


Рис. 3.9. Выпуклая область решений,
нейронная сеть, задаваемая двухслойной сетью

Нейрон второго слоя не обязательно ограничен функцией «*и*», он может реализовывать многие другие функции при подходящем выборе весов и порога. Например, можно сделать так, чтобы единичный выход любого из нейронов первого слоя приводил к появлению единицы на выходе нейрона второго слоя, реализовав тем самым логическое «*или*». Имеется 16 двоичных функций от двух переменных. Если выбирать подходящим образом веса и порог, то можно воспроизвести 14 из них (все, кроме «*исключающее или*» и «*исключающее нет*»).

Входы не обязательно должны быть двоичными. Вектор непрерывных входов может представлять собой произвольную точку на плоскости XOY. В этом случае мы имеем дело со способностью сети разбивать плоскость на непрерывные области, а не с разделением дискретных множеств точек. Для всех этих функций, однако, линейная разделимость показывает, что выход нейрона второго слоя равен единице только в части плоскости XOY, ограниченной многоугольной областью. Поэтому для разделения плоскостей P и Q необходимо, чтобы все P лежали внутри выпуклой многоугольной области, не содержащей точек Q (или наоборот).

Трехслойная сеть, однако, является более общей (рис. 3.10). Ее классифицирующие возможности ограничены лишь числом искусственных нейронов и весов. Ограничения на выпуклость отсутствуют. Теперь нейрон третьего слоя принимает в качестве входа набор выпуклых многоугольников, и их логическая комбинация может быть невыпуклой.

На рис. 3.11 иллюстрируется случай, когда два треугольника A и B, скомбинированные с помощью функций «A и не B», задают невыпуклую область. При добавлении нейронов и весов число сторон многоугольников может неограниченно возрастать. Это позволяет аппроксимировать область любой формы с любой точностью. Вдобавок не все выходные области второго слоя должны пересекаться. Возможно, следовательно, объединять различные области, выпуклые и невыпуклые, выдавая на выходе единицу всякий раз, когда входной вектор принадлежит одной из них.

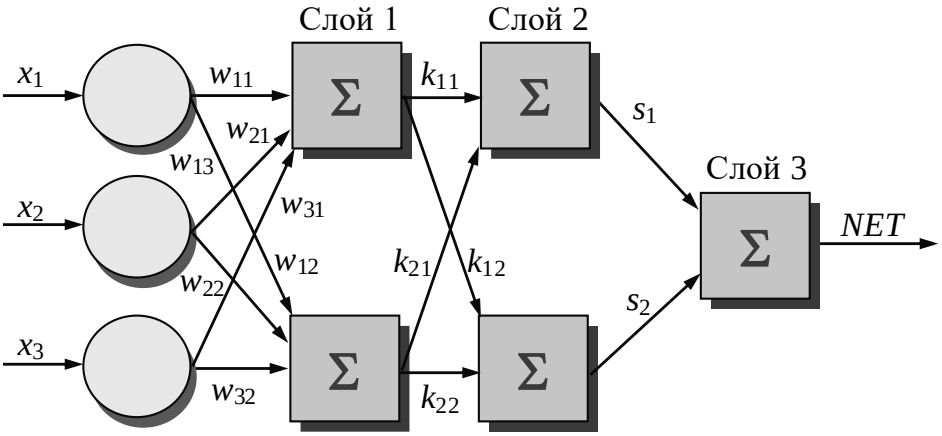


Рис. 3.10. Простейшая двухслойная нейронная сеть

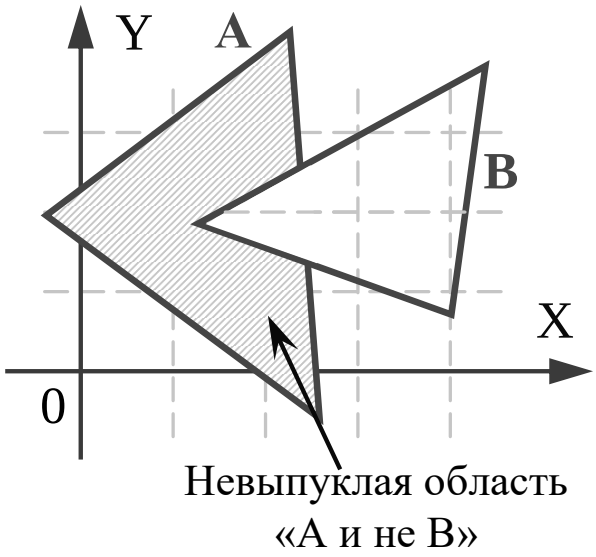


Рис. 3.11. Вогнутая область решений, задаваемая трехслойной сетью

Более крупные и сложные нейронные сети обладают, как правило, и большими вычислительными возможностями. Хотя созданы сети всех конфигураций, какие только можно себе представить, послойная организация нейронов копирует слоистые структуры определенных отделов мозга.

Существуют даже трехмерные и многомерные слоистые нейронные структуры. Однако, многослойные сети могут привести к увеличению вычислительной мощности по сравнению с однослойной сетью лишь в том случае, если активационная функция между слоями будет нелинейной. Вычисление выхода слоя заключается в умножении входного вектора на первую весовую матрицу с последующим умножением (если отсутствует нелинейная

активационная функция) результирующего вектора на вторую весовую матрицу

$$NET = (\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{W}_2.$$

Так как умножение матриц ассоциативно, то

$$NET = \boldsymbol{X}(\boldsymbol{W}_1\boldsymbol{W}_2). \qquad (3.4)$$

Это показывает, что двухслойная линейная сеть эквивалентна одному слою с весовой матрицей, равной произведению двух весовых матриц. Следовательно, любая многослойная линейная сеть может быть заменена эквивалентной однослойной сетью.

3.5. Обучение нейронных сетей

Способность искусственных нейронных сетей обучаться является их наиболее интригующим свойством. Подобно биологическим системам, которые они моделируют, эти нейронные сети сами моделируют себя в результате попыток достичь лучшей модели поведения.

Используя критерий линейной разделимости, можно решить, способна ли однослойная нейронная сеть реализовывать требуемую функцию. Даже в том случае, когда ответ положительный, это принесет мало пользы, если у нас нет способа найти нужные значения весовых коэффициентов. Чтобы сеть представляла практическую ценность, нужен систематический метод (алгоритм) для вычисления этих значений. Розенблатт [64] сделал это в своем алгоритме обучения персептрона вместе с доказательством того, что персептрон может быть обучен всему, что он может реализовывать.

Целью обучения нейронной сети *является такая подстройка ее весов, чтобы приложение некоторого множества входов приводило к требуемому множеству выходов*.

Для краткости эти множества входов и выходов будут называться *векторами.* При обучении предполагается, что для каждого входного вектора существует парный ему целевой вектор, задающий требуемый выход. Вместе они называются ***обучающей парой***. Как правило, сеть обучается на многих парах. Например, входная часть обучающей пары может состоять из набора нулей и единиц, представляющего двоичный образ некоторой буквы алфавита. Выход может быть числом, представляющим букву «А», или другим набором из нулей и единиц, который может быть использован для получения выходного образа. При необходимости

распознавать с помощью сети все буквы алфавита, потребовалось бы 26 обучающих пар. Такая группа обучающих пар называется ***обучающим множеством***.

Обучение может быть с учителем или без него. Для обучения с учителем нужен «внешний» учитель, который оценивал бы поведение системы и управлял ее последующими модификациями. При обучении без учителя сеть путем самоорганизации делает требуемые изменения.

Рассмотрим обучение персептрона на задаче распознавания образов (рис. 3.12). Обучение персептрона является обучением с учителем.

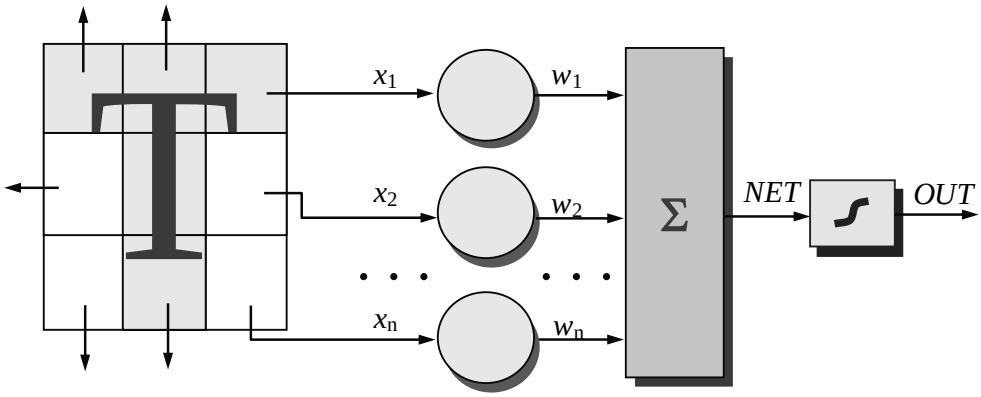


Рис. 3.12. Персептронная система распознавания образов

Персептрон обучают, подавая множество образов по одному на его вход и подстраивая веса до тех пор, пока для всех образов не будет достигнут требуемый выход.

Пусть, для простоты, изображение будет монохромным. Тогда если пиксель содержит точку изображения, то от него подается единица, в противном случае – ноль. Множество пикселей создает вектор входа персептрона.

Допустим, что вектор $\boldsymbol{X}$ содержит значения признаков распознаваемого образа (пикселей) – (x_1, x_2, …, x_n), которые умножаются на соответствующие компоненты вектора весов $\boldsymbol{W}$ – (w_1, w_2, ..., w_n). Эти произведения суммируются. Если сумма превышает порог Θ, то выход нейрона *OUT* равен единице, в противном случае – нулю.

Для обучения сети образ $\boldsymbol{X}$ подается на вход и вычисляется выход *OUT*. Если *OUT* правилен, то ничего не меняется. Однако если выход неправилен, то веса, присоединенные к входам, усиливающим ошибочный результат, модифицируются, чтобы уменьшить ошибку.

Метод обучения состоит из следующих шагов:

1. Выбирается очередной входной образ.
2. Значения признаков образа подаются на персептрон и вычисляется *OUT*.
3. Если выход правильный, то переходят на первый шаг.
4. Если выход неправильный и равен нулю, то добавить значения всех входов к соответствующим им весам; или если выход неправильный и равен единице, то вычесть значение каждого входа из соответствующего ему веса.
5. Перейти на первый шаг.

За конечное число шагов сеть научится разделять образы. Отметим, что это обучение *глобально*, т. е. сеть обучается на всем множестве образов. Возникает вопрос о том, как это множество должно предъявляться, чтобы минимизировать время обучения. Должны ли элементы множества предъявляться последовательно друг за другом или их следует выбирать случайно?

Существует несколько правил изменения шага и подачи образов.

Дельта-правило. Важное обобщение алгоритма обучения персептрона, называемое дельта-правилом, переносит этот метод на непрерывные входы и выходы. Вводится величина δ, которая равна разности между требуемым или целевым выходом *T* и реальным выходом *OUT*

$$\delta = (T - OUT). \qquad (3.5)$$

Случай, когда $\delta = 0$, соответствует шагу 3, т.е. когда выход правилен и в сети ничего не изменяется. Шаг 4 соответствует случаю $\delta <> 0$.

В любом из этих случаев персептронный алгоритм обучения сохраняется: δ умножается на величину каждого входа x_i и это произведение добавляется к соответствующему весу. С целью обобщения вводится коэффициент «скорости обучения» η, который умножается на δx_i, что позволяет управлять средней величиной изменения весов.

В алгебраической форме записи

$$\Delta_i = \eta \delta x_i,$$

$$w(n+1) = w(n) + \Delta_i,$$

где Δ_i – коррекция, связанная с *i*-м входом x_i; $w_i(n+1)$ – значение *i*-го веса после коррекции; $w_i\{n)$ – значение *i*-го веса до коррекции.

Дельта-правило модифицирует веса в соответствии с требуемым и действительным значениями выхода каждой полярности как для непрерывных, так и для бинарных входов и выходов.

3.6. Процедура обратного распространения

На рис. 3.13 изображена двухслойная сеть, которая может обучаться с помощью процедуры обратного распространения (рисунок упрощен).

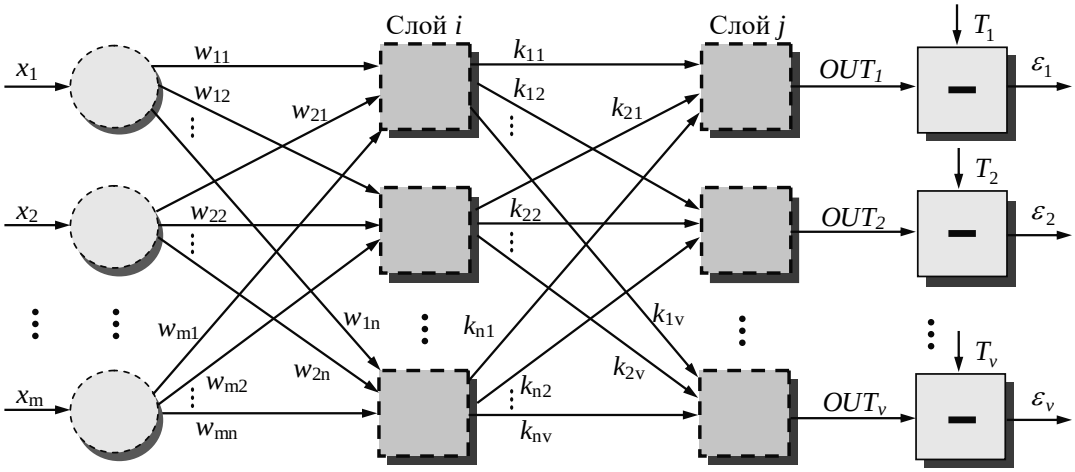


Рис. 3.13. Двухслойная сеть обратного распространения

На вход сети поступает входной вектор сигналов $\boldsymbol{X}$ (x_1, x_2, …, x_m), который умножается на матрицу весов $\boldsymbol{W}$ (на рис.3.13 показаны все возможные соединения, хотя в реальности ряд из них может отсутствовать). На рис.3.13 пунктирными кружками обозначены блоки умножения скаляра на вектор. Далее сигналы поступают на первый слой нейронной сети (слой *i*), где они суммируются и модифицируются активационными функциями каждого нейрона, а затем полученные выходы умножаются на вектор весов для передачи сигналов на второй слой нейронной сети (слой *j*). В слое *j* нет блока умножения выхода нейронов на вектор. Каждый из нейронов в этих слоях на рисунке представлен пунктирным квадратом.

Процедура обратного распространения применима к сетям с любым числом слоев. Однако для того, чтобы продемонстрировать алгоритм, достаточно двух слоев. Сейчас будут рассматриваться лишь сети прямого действия, хотя обратное распространение применимо и к сетям с обратными связями.

Фактически процедура обратного распространения является алгоритмом обучения сети без учителя.

Перед началом обучения всем весам должны быть присвоены небольшие начальные значения, выбранные случайным образом. Это гарантирует, что в сети не произойдет насыщения большими значениями весов, и предотвращает ряд других патологических случаев. Например, если всем весам придать одинаковые начальные значения, а для требуемого функционирования нужны неравные значения, то сеть не сможет обучиться.

Обучение сети процедурой обратного распространения требует выполнения следующих операций:

1. Выбрать очередную обучающую пару из обучающего множества и подать входной вектор на вход сети.
2. Вычислить выход сети.
3. Вычислить разность между выходом сети и требуемым выходом (целевым вектором обучающей пары).
4. Подкорректировать веса сети так, чтобы минимизировать ошибку.
5. Повторять шаги с 1 по 4 для каждого вектора обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемого уровня.

Операции, выполняемые шагами 1 и 2, сходны с теми, которые выполняются при функционировании уже обученной сети, т. е. подается входной вектор и вычисляется получающийся выход. Вычисления выполняются послойно. На рис. 3.13 сначала вычисляются выходы нейронов слоя *i*, затем они используются с некоторыми коэффициентами в качестве входов слоя *j*. После вычисляются выходы нейронов слоя *j*, которые и образуют выходной вектор сети.

На шаге 3 каждый из выходов сети, которые на рис. 13 обозначены OUT_q, вычитается из соответствующей компоненты целевого вектора T_q, чтобы получить ошибку ε_q. Эта ошибка используется на шаге 4 для коррекции весов сети, причем знак и величина изменений весов определяются алгоритмом обучения.

После достаточного числа повторений этих четырех шагов разность между действительными выходами и целевыми выходами должна уменьшиться до приемлемой величины, при этом говорят, что сеть обучилась. Тогда сеть может использоваться для распознавания и веса больше не изменяются.

Шаги 1 и 2 можно рассматривать как «проход вперед», так как сигнал распространяется по сети от входа к выходу. Шаги 3, 4 составляют «обратный проход», здесь вычисляемый сигнал ошибки распространяется обратно по сети и используется для подстройки весов.

Проход вперед. Шаги 1 и 2 могут быть выражены в векторной форме следующим образом: подается входной вектор $\boldsymbol{X}$ и на выходе получается вектор $\boldsymbol{OUT}$. Векторная пара вход-цель $\boldsymbol{X}$ и $\boldsymbol{T}$ берется из обучающего множества. Вычисления проводятся над вектором $\boldsymbol{X}$, чтобы получить выходной вектор $\boldsymbol{OUT}$.

Величина *NET* каждого нейрона первого слоя вычисляется как взвешенная сумма входов нейрона. Затем активационная функция *F* «сжимает» *NET* и дает величину *OUT* для каждого нейрона в этом слое. Когда множество выходов слоя получено, оно является входным множеством для следующего слоя. Процесс повторяется слой за слоем, пока не будет получено заключительное множество выходов сети.

$$OUT = F_2(F_1(\boldsymbol{X}\boldsymbol{W_1})\boldsymbol{W_2}), \qquad (3.6)$$

где $\boldsymbol{W_1}$ – матрица весовых коэффициентов входного слоя, $\boldsymbol{W_2}$ – матрица весовых коэффициентов передачи выходов первого слоя нейронной сети на входы второго слоя, F_1 – вектор активационных функций нейронов первого слоя, F_2 – вектор активационных функций нейронов второго слоя.

Обратный проход. На данном этапе осуществляется *подстройка весов выходного слоя.* Так как для каждого нейрона выходного слоя задано целевое значение, то подстройка весов этого слоя легко осуществляется с использованием модифицированного дельта-правила (5). Внутренние слои называют «скрытыми слоями», для их выходов не имеется целевых значений для сравнения. Поэтому обучение усложняется.

Выход нейрона слоя *j*, вычитаясь из целевого значения T_q, дает сигнал ошибки ε_q. Он умножается на производную активационной функции (для логистической функции (3.1) она равна $OUT_q\ (1 - OUT_q)$) вычисленную для этого нейрона q слоя j, давая, таким образом, величину δ_q

$$\delta_q = OUT_q\ (1 - OUT_q)(T_q - OUT_q). \qquad (3.7)$$

Затем δ_q умножается на величину OUT_p нейрона слоя i, из которого выходит рассматриваемый вес. Это произведение в свою очередь умножается на коэффициент скорости обучения η (обычно от 0.01 до 1.0), и результат прибавляется к весу. Такая же процедура выполняется для каждого веса от нейрона скрытого слоя к нейрону в выходном слое.

Следующие уравнения иллюстрируют это вычисление

$$\Delta k_{pq} = \eta\, \delta_q\, OUT_p, \tag{3.8}$$

$$k_{pq}(\mathrm{n+1}) = k_{pq}(\mathrm{n}) + \Delta k_{pq}, \tag{3.9}$$

где $k_{pq}(\mathrm{n})$ – величина веса от нейрона p в скрытом слое к нейрону q в выходном слое на шаге n (до коррекции); $k_{pq}(\mathrm{n+1})$ – величина веса на шаге n + 1 (после коррекции); δ_q – величина δ для нейрона q, в выходном слое j; OUT_p – величина OUT для нейрона p в скрытом слое i.

Подстройка весов скрытого слоя. Рассмотрим один нейрон в скрытом слое i, предшествующем выходному слою j. При проходе вперед этот нейрон передает свой выходной сигнал нейронам в выходном слое через соединяющие их веса. Во время обучения эти веса функционируют в обратном порядке, пропуская величину δ_q от выходного слоя назад к скрытому слою. Каждый из этих весов умножается на величину δ_q нейрона, к которому он присоединен в выходном слое. Величина δ_p, необходимая для нейрона скрытого слоя, получается суммированием всех таких произведений и умножением на производную сжимающей функции

$$\delta_p = OUT_p\left(1 - OUT_p\right)\left[\sum_q \delta_q k_{pq}\right]. \tag{3.10}$$

Когда значение δ_p получено, веса, первого скрытого слоя, могут быть подкорректированы с помощью уравнений

$$\Delta w_{xp} = \eta\, \delta_p\, x_p, \tag{3.11}$$

$$w_{xp}(\mathrm{n+1}) = w_{xp}(\mathrm{n}) + \Delta w_{xp}. \tag{3.12}$$

где $w_{xp}(\mathrm{n})$ – величина веса от входа x к нейрону p в скрытом слое на шаге n (до коррекции); $w_{xp}(\mathrm{n+1})$ – величина веса на шаге n + 1 (после коррекции); δ_p – величина δ для нейрона p, в скрытом слое i; x_p – величина входа для нейрона p в скрытом слое i.

Для каждого нейрона в данном скрытом слое должно быть вычислено δ_p и подстроены все веса, ассоциированные с этим слоем. Этот процесс повторяется слой за слоем по направлению к входу, пока все веса не будут подкорректированы.

С помощью векторных обозначений операция обратного распространения ошибки может быть записана значительно компактнее. Обозначим множество величин δ_q выходного слоя через $\boldsymbol{D}_q$ и множество весов выходного слоя как массив $\boldsymbol{K}_q$. Чтобы получить $\boldsymbol{D}_j$, δ-вектор выходного слоя, достаточно следующих двух операций:

1. Умножить вектор выходного слоя $\boldsymbol{D}_q$ на транспонированную матрицу весов K_q^T, соединяющую скрытый уровень с выходным уровнем.

2. Умножить каждую компоненту полученного произведения на производную сжимающей функции соответствующего нейрона в скрытом слое.

В символьной записи

$$D_j = D_q K_q^T * \left[OUT_p * (I - OUT_p)\right], \qquad (3.13)$$

где * – оператор поэлементного произведения векторов, OUT_p – выходной вектор слоя i, $\boldsymbol{I}$ – вектор, все элементы которого равны единице.

Импульс. В работе [6] описан метод ускорения обучения для алгоритма обратного распространения, увеличивающий также устойчивость процесса. Этот метод, названный импульсом, заключается в добавлении к коррекции веса члена, пропорционального величине предыдущего изменения веса. Как только происходит коррекция, она «запоминается» и служит для модификации всех последующих коррекций. Уравнения коррекции (3.8), (3.9) и по аналогии (3.11), (3.12) модифицируются следующим образом:

$$\Delta k_{pq}(\mathrm{n+1}) = \eta\, \delta_q\, OUT_p + \alpha\, \Delta k_{pq}(\mathrm{n}), \qquad (3.14)$$

$$k_{pq}(\mathrm{n+1}) = k_{pq}(\mathrm{n}) + \Delta k_{pq}(\mathrm{n+1}), \qquad (3.15)$$

где α – коэффициент импульса, обычно устанавливается около 0.9.

Используя метод импульса, сеть стремится идти по дну узких оврагов поверхности ошибки (если таковые имеются), а не двигаться от склона к склону. Этот метод, по-видимому, хорошо работает на некоторых задачах, но дает слабый или даже отрицательный эффект на других.

В работе [67] описан сходный метод, основанный на экспоненциальном сглаживании, который может иметь преимущество в ряде приложений. В этом случае формулы (3.14), (3.15) будут иметь вид

$$\Delta k_{pq}(\mathrm{n+1}) = (1\text{-}\alpha)\, \delta_q\, OUT_p + \alpha\, \Delta k_{pq}(\mathrm{n}), \qquad (3.16)$$

$$k_{pq}(n+1) = k_{pq}(n) + \eta\Delta k_{pq}(n+1), \qquad (3.17)$$

где α – коэффициент сглаживания, варьируемый и диапазоне от 0.0 до 1.0. Если α равен 1.0, то новая коррекция игнорируется и повторяется предыдущая. В области между 0 и 1 коррекция веса сглаживается величиной, пропорциональной α.

Метод обратного распространения имеет ряд недостатков:

– **паралич сети** – когда в процессе обучения значения весов становятся большими величинами, т.е. нейроны будут иметь большие значения выходов, а поскольку производная активационной функции при больших значениях мала, то производная в (3.7), (3.10), (3.13) будет также мала. В результате процесс обучения «замрет». Для предотвращения паралича обычно уменьшают размер шага η, но это увеличивает время обучения. Можно также попытаться ввести эвристические правила обнаружения паралича;

– **локальные минимумы** – в процессе обучения будут найдены не наилучшие значения весов, поскольку обратное распространение представляет собой разновидность градиентного спуска, т. е. осуществляет спуск вниз по поверхности ошибки, непрерывно подстраивая веса в направлении к минимуму. Если поверхность ошибки имеет сложную форму, то обучение застревает в локальном экстремуме. Для избегания подобной ситуации используют статистические методы обучения, но они медленны;

– **размер шага** – в [66] говорится о том, что коррекция весов должна осуществляться бесконечно малым шагом, но на практике шаг должен быть конечным и достаточно большим. В тоже время размер шага не должен приводить к параличу сети и постоянной неустойчивости;

– **временная неустойчивость** – заключается в том, что нейронная сеть забывает, чему ее уже обучили, при дальнейшем обучении. Процесс обучения должен быть таким, чтобы сеть обучалась на всем обучающем множестве без пропусков того, что уже выучено. Рекомендует перед коррекцией весов на вход сети подать все обучающее множество. К сожалению, если входные образы не могут быть повторены данный подход не применим, поскольку будут постоянные осцилляции.

3.6.1. Программная реализация нейросети обратного распространения ошибки

Алгоритм реализован в виде класса NeuroNetwork. Для описания нейрона создана структура neuron_type, содержащая следующие поля:

- список весовых коэффициентов для связей между данным нейроном и всеми нейронами предыдущего слоя (или входными данными, если нейрон находится во входном слое). Каждый весовой коэффициент - действительное число (по одному весовому коэффициенту на нейрон предыдущего слоя);
- пороговый уровень;
- значение ошибки, используется только на стадии обучения;
- изменение ошибки, также используется только во время обучения.

Выходной сигнал нейрона хранится в поле *a* (так называемая активность нейрона). Нейрон должен отреагировать на входной сигнал, поступающий по взвешенным связям, вычислив при этом выходной сигнал. Для трансформации входных сигналов в выходные необходима функция активации. Функция активации $x_i^{[n+1]} = f\left(a_i^{[n]}\right)$ реализована в открытом методе sigmoid. В данном методе используется формула $f(x) = \frac{1}{1+e^{-\alpha x}}$.

Биологические нейроны не срабатывают (не выдают выходной сигнал) до тех пор, пока уровень входного сигнала не достигнет некоторого порогового значения, т.е. на вход нейрона поступает сумма взвешенных сигналов минус некоторая величина. Полученное значение проходит через активационную функцию.

Каждая активность нейрона предыдущего слоя умножается на соответствующий весовой коэффициент, результаты умножения суммируются $a_i^{[n]} = \sum_j w_{ij}^{[n]} x_j^{[n]}$, вычитается пороговое значение, вычисляется значение сигмоидной функции $x_i^{[n+1]} = f\left(\sum_j w_{ij}^{[n]} x_j^{[n]}\right)$.

Вычисление суммы взвешенных сигналов для скрытого и выходного слоев (методы run_hidden_layer и run_output_layer) производится аналогично.

Каждый слой нейронов базируется на выходе предыдущего слоя (за исключением входного слоя, базирующегося непосредственно на предъявляемых сети входных данных (в коде – массив test_pat). Это значит, что значения входного слоя должны быть полностью рассчитаны до вычисления значений скрытого слоя, которые в свою очередь должны быть рассчитаны до вычисления значений выходного слоя.

Выходы нейронной сети – значения активностей (поле *a*) нейронов выходного слоя. Программа, симулирующая работу нейронной сети, в процессе обучения будет сравнивать их со значениями, которые должны быть на выходе сети.

Полный алгоритм обучения НС с помощью процедуры обратного распространения строится следующим образом:

1. Инициализировать пороговые значения и весовые коэффициенты небольшими случайными величинами (не более 0.4). Инициализация весовых коэффициентов случайными вещественными значениями с помощью класса Random производится в функции random_weights.

2. Подать на входы сети один из возможных образов и в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних. Метод run_the_networ.

3. Вычислить ошибки для выходного слоя (calculate_output_layer_errors). При этом используем формулу $E_i=(t_i-a_i).a_i.(1-a_i)$ (здесь E_i – ошибка для i-го узла выходного слоя, a_i – активность данного узла, t_i – требуемая активность для требуемого выходного значения) для каждого i-го значения выходного слоя. Ниже представлена соответствующая программа в виде функции.

Вычисление ошибки для скрытого и входного слоев производится методами calculate_input_layer_errors и calculate_hidden_layer_errors по формуле $E_i = a_i. (1 – a_i). У_j E_j.w_{ij}$.

Используя формулы, получаем функцию, обучающую весовые коэффициенты и пороговые уровни.

Далее объединяем указанные выше функции в одном методе back_propogate().

Для очистки предыдущих значений используется функция blank_changes.

Для упрощения временной сложности работы сети полученные весовые коэффициенты будем записывать в отдельные файлы (метод AddWeightsToFile()), имена которым даются программой автоматически Для считывания сохраненных параметров будет применяться метод ExtractWeights().

Класс перевода текста в двоичный вид предназначен для бинаризации исходных данных (слов), т.е. перевода слов с естественного языка в набор единиц и нулей. Данная процедура является необходимой, так как нейронная сеть обратного распространения ошибки работает только с двоичными данными.

Параметром конструктора является массив строк для перевода в двоичный вид.

Алгоритм работы методов GetBinarizeWord и GetBinarizeText данного класса в общем состоит из следующих этапов:

- кодировка слова: суммирование произведений ASCII-кодов букв на i+4, где i – номер буквы в слове;
- перевод полученного десятичного числа в двоичный вид при помощи метода DecToBin;
- обработка полученных данных.

Класс распознавания является главным. Его задача – распознавание. В нем используются объекты всех описанных выше классов.

Параметрами конструктора являются:

- текст для анализа – sText;
- параметры нейросети N_HID, beta, m, Epoch;
- индикатор необходимости обучения нейросети – flag.

Из всех перечисленных выше методов наиболее важными являются конструктор, Scaning и GetNeuroResult.

Анализируемый текст сначала подается в конструктор. Там он разбивается на отдельные лексемы, тип которых либо определяется сразу (если это знак препинания или имя собственное), либо посредством метода Scaning. Метод Scaning логически можно разделить на три блока:

- поиск слова в хеш-таблицах;

- идентификация слова по окончанию и определение его типа;

- идентификация слова по окончанию и проведение углубленного анализа с помощью нейронной сети.

При решении задачи распознавания печатных символов можно выделить два этапа:

- локализация символа на изображении;

- распознавание символов на знаке.

Для упрощения дальнейших расчетов и программной реализации преобразуем исходное цветное изображение в полутоновое (рис. 3.14).

Рис. 3.14. Преобразование изображения в оттенки серого

Теперь, как и при решении большинства других задач, необходимо провести предварительную обработку исходных изображений. Этот этап особенно важен при решении задач распознавания. От качества решения этапа предварительной обработки во многом зависит эффективность решения задачи распознавания в целом.

К наиболее распространенным дефектам, которые могут присутствовать на исходных изображениях, относятся шум и низкий уровень контрастности. Для устранения импульсных выбросов используем медианную фильтрацию.

Результат обработки представлен на рис. 3.15.

Рис. 3.15. Изображение с импульсными шумами и
после устранения импульсных шумов

Фильтр, повышающий резкость изображения, имеет маску, определяемую следующим выражением:

$$h = \frac{1}{a+1}\begin{bmatrix} -a & a-1 & -a \\ a-1 & a+5 & a-1 \\ -a & a-1 & -a \end{bmatrix},$$

где параметр *a* выбирается в пределах от 0 до 1.

Кроме этого, для улучшения качества изображения можно воспользоваться гистограммой. На рис. 3.16 показано, как устраняется размытость изображения.

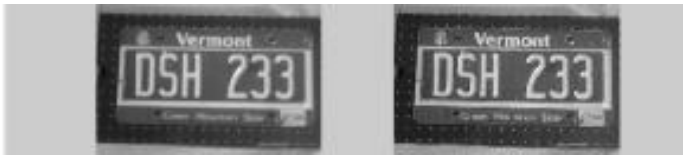


Рис. 3.16. Устранение размытости изображения

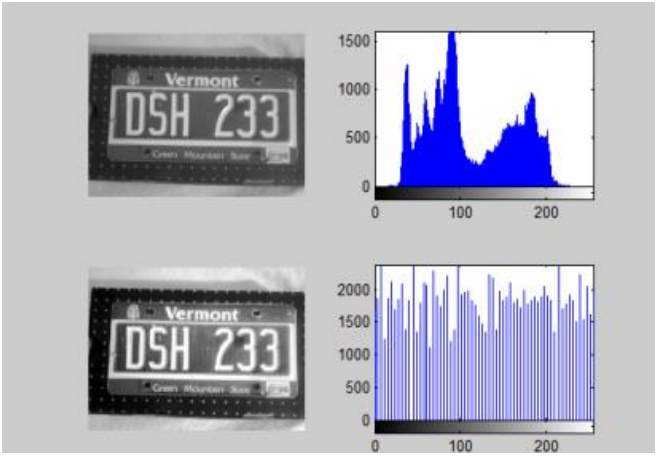


Рис. 3.17. Выравнивание гистограммы

Гистограмма улучшает контраст изображения с помощью преобразования значений пикселей исходного изображения таким образом, чтобы гистограмма яркостей пикселей результирующего изображения соответствовала некото-рой предопределенной гистограмме (рис. 3.17). Данная функция пред-назначена для преобразования полутоновых изображений или палитровых изображений.

Определение расположения печатных символов на изображении. После проведения предварительной обработки изображения необходимо определить расположение печатных символов. Для этого на изображении ищутся так называемые связанные области пикселей и создается матрица *L*, каждый элемент которой равен номеру объекта, которому принадлежит соответствующий пиксель изображения. Размер матрицы номеров объектов *L* равен размеру изображения. Объекты нумеруются по порядку, начиная с 1. Элементы, имеющие значение 1 , относятся к первому объекту, имеющие значение 2 - ко второму объекту и т.д. Если элемент в матрице *L* равен 0, то это означает, что соответствующий пиксель исходного изображения относится к фону. Параметр *n* указывает критерий связности, используемый для нахождения связных областей-объектов. Параметр *n* может принимать значения 4 или 8 (значение по умолчанию).

На рис. 3.18 показан результат выполнения программы.

Рис. 3.18. Результат выполнения программы

Следующий этап – вычисление признаков найденных объектов. Элементы матрицы *L*, имеющие значение 1, относятся к первому объекту, имеющие значение 2 - ко второму объекту и т.д. Если элемент в матрице *L* равен 0, то он относится к фону.

Пусть *N* - количество пикселей, относящихся к объекту. Все множество пикселей *p(x, y),* относящихся к объекту, обозначим через *Q*. Тогда координаты центра масс объекта вычисляются как

$$x_c = \frac{1}{N} \sum_{p(x,y)\in\Omega} x \quad y_c = \frac{1}{N} \sum_{p(x,y)\in\Omega} y.$$

Затем вычисляется несколько вспомогательных величин:

$$U_x = \frac{1}{12} + \frac{1}{N} \sum_{p(x,y)\in\Omega} (x - x_c)^2,$$

$$U_y = \frac{1}{12} + \frac{1}{N} \sum_{p(x,y)\in\Omega} (y - y_c)^2,$$

$$c = \sqrt{(U_x - U_y)^2 + 4 \cdot U_{xy}^2}.$$

Тогда длины максимальной $A_{\max}$ и минимальной $A_{\min}$ осей инерции вычисляются следующим образом:

$$A_{\min} = 2\sqrt{2} \cdot \sqrt{U_x + U_y - C}.$$

Длины главных осей инерции используются для вычисления эксцентриситета и ориентации объекта. Эксцентриситет определяется с помощью соотношения

$$E = \frac{2 \cdot \sqrt{(0.5 \cdot A_{\max})^2 - (0.5 \cdot A_{\min})^2}}{A_{\max}}.$$

Ориентация определяется как угол в градусах между максимальной осью инерции и осью *X*. Если $U_y > U_x$, то ориентация *O* вычисляется с помощью формулы

$$O = \frac{180}{\pi} \cdot arctg\left(\frac{U_y - U_x + C}{2 \cdot U_{xy}}\right),$$

в противном случае *O* вычисляется как

$$O = \frac{180}{\pi} \cdot arctg\left(\frac{2 \cdot U_{xy}}{U_y - U_x + C}\right).$$

Рис. 3.19. Цифрами обозначены найденные объекты

Приведем результаты вычислений признаков для всех объектов изображения. В качестве параметра был выбран коэффициент заполнения, который равен отношению площади объекта к площади ограничивающего прямоугольника.

3.6.2. Распознавание символов

После локализации печатных символов на изображении выполняется второй этап – распознавание символов. Первый шаг – выделение части изображения. Результат выполнения функции показан на рис. 3.20.

Рис. 3.20. Исходное изображение с номерным знаком и изображение с выделенной областью

Получаем координаты расположения первого символа на изображении (рис. 3.21).

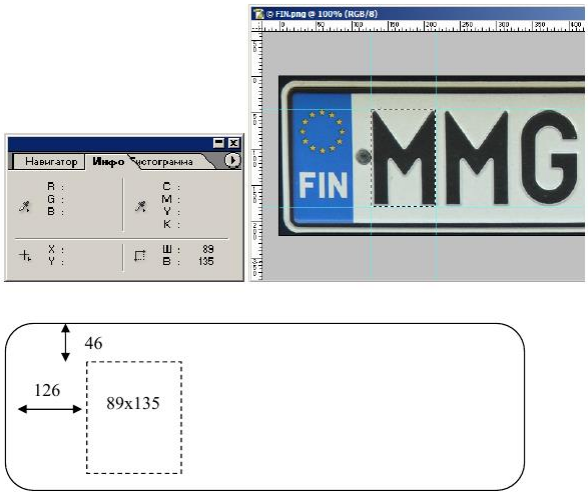


Рис. 3.21. Определение координат первого символа

Используя исходное изображение и программу Photoshop, создаем эталонное изображение (рис. 3.22).

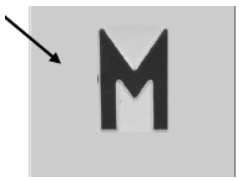


Рис. 3.22. Исходное и эталонное изображения

В настоящей работе распознаются только бинарные образы, поэтому на втором этапе после получения картинки она

бинаризуется. При работе с цветной камерой преобразование из цвета в черно-белый происходит по стандартной формуле:

$$Y:=0.3*R+0.59*G+0.11*B.$$

Далее алгоритм довольно прост: есть некоторая планка, если цвет оттенка серого выше - он считается белым, если ниже - считается черным. На стадии бинаризации преобразуется объект изображения в бинарную матрицу данных, после чего начинается работа уже не с объектом картинки, а с бинарной матрицей.

Если не предусматривать некоторое разбиение общего изображения на части, то ни один из описанных выше алгоритмов не сможет корректно работать. Разбиение изображения на части, каждая из которых содержит свой уникальных объект, называется сегментацией.

Следует заметить, что в сегментации четко разделяются черно-белые изображения на бинарные и с оттенками серого. Здесь работают совершенно разные по быстроте и сложности алгоритмы, однако интуитивно понятно, что любое изображение с оттенками серого можно бинаризовать по некоторым правилам.

После успешного завершения сегментации каждый сегмент попадает в модуль распознавания. Для того чтобы образы распознавались инвариантно относительно положения и поворота, надо привязаться к их структуре. У каждого бинарного образа можно вычислить несколько признаков, не зависящих от его поворота или размера.

Опишем применение нейронных сетей (НС) для распознавания изображений.

НС состоит из элементов, называемых формальными нейронами, которые сами по себе очень просты и связаны с другими нейронами. Каждый нейрон преобразует набор сигналов, поступающих к нему на вход в выходной сигнал. Именно связи между нейронами, кодируемые весами, играют ключевую роль. Одно из преимуществ НС (а также недостаток при реализации их на последовательной архитектуре) это то, что все элементы могут функционировать параллельно, тем самым существенно повышая эффективность решения задачи, особенно в обработке изображений. Кроме того, что НС позволяют эффективно решать многие задачи, они предоставляют мощные гибкие и универсальные механизмы обучения, а это является их главным преимуществом перед другими

методами [61, 62] (вероятностные методы, линейные разделители, решающие деревья и т.п.). Обучение избавляет от необходимости выбирать ключевые признаки, их значимость и отношения между признаками. Но, тем не менее, выбор исходного представления входных данных (вектор в *n*-мерном пространстве, частотные характеристики, вэйвлеты и т.п.) существенно влияет на качество решения и является отдельной темой. НС обладают хорошей обобщающей способностью (лучше, чем у решающих деревьев [62]), т.е. могут успешно распространять опыт, полученный на конечном обучающем наборе, на все множество образов.

Исследование методов и программно-аппаратных систем оптического распознавания символов позволяет сформулировать следующие выводы:

1. Современное состояние технологии автоматического распознавания печатных текстов (OCR) позволяет решать задачу автоматизации ввода информации при необходимом уровне надежности.

2. При построении системы OCR, включающей оптическое устройство оцифровки изображений, блоки локализации и выделения элементов текста, предобработки изображения, выделения признаков, распознавания символов и постобработки результатов распознавания, необходимо использовать методы и алгоритмы, обладающие высокой робастностью к яркостно-геометрическим искажениям и сложным текстурным фонам.

3. В качестве таких методов и алгоритмов могут быть использованы: процедуры определения строк, знакомест на основе модификаций преобразования Hough; методы, основанные на исследовании устойчивых статистических распределений точек; методы, использующие интегральные преобразования, а также структурный анализ символов.

4. При разработке современных систем OCR для повышения качества распознавания символов и слов необходимо учитывать контекстную информацию, использование которой позволяет не только находить ошибки, но и исправлять их.

Переходя к программе, разработанной в ходе выполнения исследований, необходимо отметить, что хотя в ней и не применяются системы искусственного интеллекта (перцептроны и нейросети), а используется довольно простой метод сравнения с

эталонными символами, алгоритм дает приемлемый результат на заранее известном наборе эталонов.

Применение этого метода уместно в тех случаях, когда необходимо распознавать большие объемы текстов, напечатанных одним шрифтом в едином размере. При таких условиях результаты распознавания могут конкурировать с методами, основанными на использовании нейросетей, и не уступать им по скорости распознавания.

При всем этом метод сравнения с эталоном гораздо проще других алгоритмов, использует простой математический аппарат. Однако небольшие отклонения входных данных от эталонных значений приводят к резкому падению качества распознавания.

3.7. Сети встречного распространения

Возможности сети встречного распространения превосходят возможности однослойных сетей, а время обучения, по сравнению с обратным распространением, может уменьшаться на несколько порядков. Встречное распространение не столь общо, как обратное распространение, но оно может давать решение в тех приложениях, где долгая обучающая процедура невозможна.

Во встречном распространении объединены два хорошо известных алгоритма: самоорганизующаяся карта Кохонена [69] и звезда Гроссберга [68]. Их объединение ведет к свойствам, которых нет ни у одного из них в отдельности.

Сеть встречного распространения обладает одним из важных свойств для распознавания образов – обобщающая способность сети позволяет получать правильный выход даже при приложении входного вектора, который является неполным или слегка неверным.

Структура сети. На рис. 3.23 показана упрощенная версия сети встречного распространения. На нем иллюстрируются функциональные свойства этой парадигмы.

Как видно из рис. 3.23 сеть встречного распространения очень похожа на двухслойную сеть обратного распространения. Различие заключается в операциях, выполняемых нейронами Кохонена и Гроссберга.

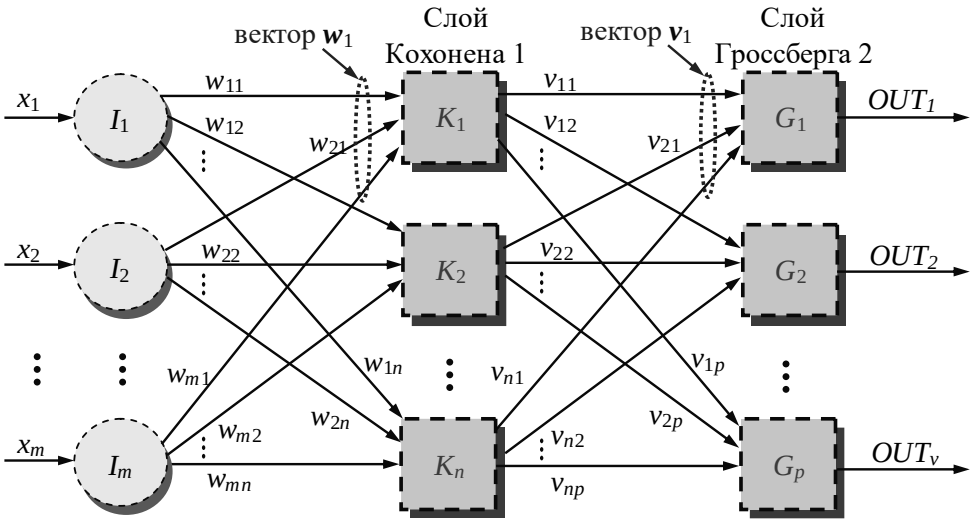


Рис. 3.23. Сеть встречного распространения без обратных связей

Слой Кохонена. В своей простейшей форме слой Кохонена функционирует следующим образом: только один нейрон из слоя выдает на выходе логическую единицу, все остальные выдают ноль для заданного входного вектора.

Ассоциированное с каждым нейроном Кохонена множество весов соединяет его с каждым входом. Например, на рис. 3.23 нейрон Кохонена K_1 имеет веса w_{11}, w_{21}, …, w_{m1}, составляющие весовой вектор $\boldsymbol{w}_1$. Они соединяются через входной слой с входными сигналами x_1, x_2, …, x_m, составляющими входной вектор $\boldsymbol{X}$. Подобно нейронам большинства сетей выход NET каждого нейрона Кохонена является просто суммой взвешенных входов

$$NET_j = \sum_i x_i w_{ij}, \tag{3.18}$$

где NET_j – это выход NET нейрона Кохонена j.

Выход нейрона Кохонена с максимальным значением NET равен единице, у остальных он равен нулю.

Слой Гроссберга. Слой Гроссберга функционирует в сходной манере. Его выход OUT является взвешенной суммой выходов нейронов K_1, K_2, ..., K_n слоя Кохонена

$$OUT_j = \sum_i NET_i v_{ij}, \tag{3.19}$$

где NET_i – выход i-го нейрона Кохонена, OUT_j – выход j-го нейрона Гроссберга.

Если слой Кохонена функционирует таким образом, что лишь у одного нейрона величина NET равна единице, а у остальных равна

нулю, то лишь один элемент векторов $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, …, $\boldsymbol{v}_p$ отличен от нуля, и вычисления очень просты. Фактически каждый нейрон слоя Гроссберга лишь выдает величину веса, который связывает этот нейрон с единственным ненулевым нейроном Кохонена.

Обучение слоя Кохонена. Слой Кохонена классифицирует входные векторы в группы схожих. Это достигается с помощью такой подстройки весов слоя Кохонена, что близкие входные векторы активируют один и тот же нейрон данного слоя. Затем задачей слоя Гроссберга является получение требуемых выходов.

Обучение Кохонена является самообучением, протекающим без учителя. Поэтому трудно (и не нужно) предсказывать, какой именно нейрон Кохонена будет активироваться для заданного входного вектора. Необходимо лишь гарантировать, чтобы в результате обучения разделялись несхожие входные векторы.

Предварительная обработка входных векторов. Весьма желательно (хотя и не обязательно) нормализовать входные векторы перед тем, как предъявлять их сети. Это выполняется с помощью деления каждой компоненты входного вектора на длину вектора. Эта длина находится извлечением квадратного корня из суммы квадратов компонент вектора. В алгебраической записи

$$x_i' = \frac{x_i}{\sqrt{x_1^2 + x_2^2 + \ldots + x_m^2}}. \qquad (3.20)$$

Это превращает входной вектор в единичный вектор с тем же самым направлением, т. е. в вектор единичной длины в m-мерном пространстве.

При обучении слоя Кохонена на вход подается входной вектор и вычисляются его скалярные произведения с векторами весов, связанными со всеми нейронами Кохонена. Далее веса нейрона с максимальным значением скалярного произведения подстраиваются. Так как скалярное произведение, используемое для вычисления величин *NET*, является мерой сходства между входным вектором и вектором весов, то процесс обучения состоит в выборе нейрона Кохонена с весовым вектором, наиболее близким к входному вектору, и дальнейшем приближении весового вектора к входному. Сеть самоорганизуется таким образом, что данный нейрон Кохонена имеет максимальный выход для данного входного вектора. Уравнение, описывающее процесс обучения имеет следующий вид:

$$w_н = w_c + \alpha(x - w_c), \qquad (3.21)$$

где $w_н$ – новое значение веса, соединяющего входную компоненту x с нейроном, имеющим единицу на выходе; $w_с$ – предыдущее значение этого веса; α – коэффициент скорости обучения, который может варьироваться в процессе обучения.

Каждый вес, связанный с нейроном Кохонена, дающим единицу на выходе, изменяется пропорционально разности между его величиной и величиной входа, к которому он присоединен. Направление изменения минимизирует разность между весом и его входом.

На рис. 3.24 этот процесс показан геометрически в двумерном виде. Сначала находится вектор $\boldsymbol{X} - \boldsymbol{W}_c$. Затем этот вектор укорачивается умножением его на скалярную величину α, меньшую единицы, в результате чего получается вектор изменения δ. Окончательно новый весовой вектор $\boldsymbol{W}_н$ является отрезком, направленным из начала координат в конец вектора δ. Отсюда можно видеть, что эффект обучения состоит во вращении весового вектора в направлении входного вектора без существенного изменения его длины.

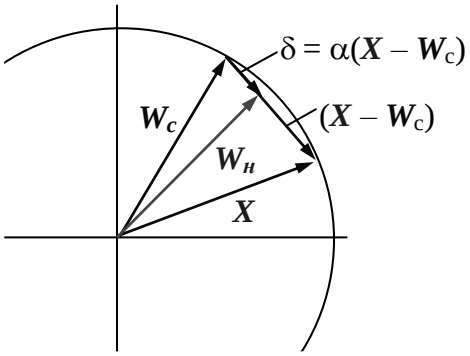


Рис. 3.24. Вращение весового вектора в процессе обучения

Коэффициентом скорости обучения вначале обычно равен 0,7 и может постепенно уменьшаться в процессе обучения. Это позволяет делать большие начальные шаги для быстрого грубого обучения и меньшие шаги при подходе к окончательной величине.

Если бы с каждым нейроном Кохонена ассоциировался один входной вектор, то слой Кохонена мог бы быть обучен с помощью одного вычисления на вес. Веса нейрона с единицей на выходе приравнивались бы к компонентам обучающего вектора ($\alpha = 1$). Как правило, обучающее множество включает много сходных между собой входных векторов, и сеть должна быть обучена активировать один и тот же нейрон Кохонена для каждого из них. В этом случае

веса этого нейрона должны получаться усреднением входных векторов, которые должны его активировать. Постепенное уменьшение величины α уменьшает воздействие каждого обучающего шага, так что окончательное значение будет средней величиной от входных векторов, на которых происходит обучение. Таким образом, веса, ассоциированные с нейроном, примут значение вблизи «центра» входных векторов, для которых данный нейрон дает единицу на выходе.

Выбор начальных значений весовых векторов. Всем весам сети перед началом обучения следует придать начальные значения. Общепринятой практикой при работе с нейронными сетями является присваивание весам небольших случайных значений. При обучении слоя Кохонена случайно выбранные весовые векторы следует нормализовать. Окончательные значения весовых векторов после обучения совпадают с нормализованными входными векторами. Поэтому нормализация перед началом обучения приближает весовые векторы к их окончательным значениям, сокращая, таким образом, обучающий процесс.

Рандомизация весов слоя Кохонена может породить проблему в процессе обучения, так как в результате ее весовые векторы распределяются равномерно по поверхности гиперсферы. Из-за того, что входные векторы, как правило, распределены неравномерно и имеют тенденцию группироваться на относительно малой части поверхности гиперсферы, большинство весовых векторов будут так удалены от любого входного вектора, что они никогда не будут давать наилучшего соответствия. Эти нейроны Кохонена будут всегда иметь нулевой выход и окажутся бесполезными. Более того, если начальная плотность весовых векторов в окрестности обучающих векторов слишком мала, то может оказаться невозможным разделить сходные классы из-за того, что не будет достаточного количества весовых векторов на интересующей поверхности гиперсферы, чтобы приписать по одному из них каждому классу входных векторов. В тоже время, если несколько входных векторов получены незначительными изменениями из одного и того же образца и должны быть объединены в один класс, то они должны включать один и тот же нейрон Кохонена. Если же плотность весовых векторов очень высока вблизи группы слегка различных входных векторов, то каждый входной вектор может

активировать отдельный нейрон Кохонена. Это не приводит к проблеме, поскольку слой Гроссберга может отобразить различные нейроны Кохонена в один и тот же выход, но это расточительная трата нейронов Кохонена.

Наиболее желательное решение состоит в том, чтобы распределять весовые векторы в соответствии с плотностью входных векторов, которые должны быть разделены, помещая тем самым больше весовых векторов в окрестности большого числа входных векторов. На практике это невыполнимо, однако существует несколько методов приближенного достижения тех же целей.

Одно из решений, известное под названием метода выпуклой комбинации (convex combination method), состоит в том, что все веса приравниваются одной и той же величине $w_i = 1/\sqrt{m}$, где m – число входов и, следовательно, число компонент каждого весового вектора. Благодаря этому все весовые векторы совпадают и имеют единичную длину. Каждой же элементу входного вектора $\boldsymbol{X}$ придается значение $x_i = \alpha x_i + (1-\alpha)/\sqrt{m}$, где m – число входов. В начале α очень мало, вследствие чего все входные векторы имеют длину, близкую к $1/\sqrt{m}$, и почти совпадают с векторами весов. В процессе обучения сети α постепенно возрастает, приближаясь к единице. Это позволяет разделять входные векторы и окончательно приписывает им их истинные значения. Весовые векторы отслеживают один или небольшую группу входных векторов и в конце обучения дают требуемую картину выходов. Метод выпуклой комбинации хорошо работает, но замедляет процесс обучения, так как весовые векторы подстраиваются к изменяющейся цели.

Другой подход состоит в добавлении шума к входным векторам. Тем самым они подвергаются случайным изменениям, схватывая в конце концов весовой вектор. Этот метод также работоспособен, но еще более медленен, чем метод выпуклой комбинации.

Третий метод начинает со случайных весов, но на начальной стадии обучающего процесса подстраивает все веса, а не только связанные с нейроном Кохонена, дающем на выходе единицу. Тем самым весовые векторы перемещаются ближе к области входных векторов. В процессе обучения коррекция весов начинает производиться лишь для ближайших к нейрону с выходом равным единице. Этот радиус коррекции постепенно уменьшается, так что в

конце концов корректируются только веса, связанные с нейроном имеющим единицу на выходе.

Еще один метод разделяет время обучения. Если нейрон Кохонена чаще других дает единицу на выходе (более $1/n$ раз, где n – число нейронов Кохонена), он временно увеличивает свой порог, что уменьшает его шансы иметь единицу на выходе, давая тем самым возможность обучаться и другим нейронам.

Во многих приложениях точность результата существенно зависит от распределения весов. К сожалению, эффективность различных решений исчерпывающим образом не оценена и остается проблемой.

Обучение слоя Гроссберга. Слой Гроссберга обучается относительно просто. Входной вектор, являющийся выходом слоя Кохонена, подается на слой нейронов Гроссберга, и выходы слоя Гроссберга вычисляются, как при нормальном функционировании. Далее, каждый вес корректируется лишь в том случае, если он соединен с нейроном Кохонена, имеющим ненулевой выход. Величина коррекции веса пропорциональна разности между весом и требуемым выходом нейрона Гроссберга, с которым он соединен. В символьной записи

$$v_{ijн} = v_{ijc} + \beta(OUT_j - v_{ijc})NET_i, \qquad (3.22)$$

где NET_i – выход i-го нейрона Кохонена (только для одного нейрона Кохонена он отличен от нуля); OUT_j – j-ая компонента вектора желаемых выходов, v_{ijc} – старое значение настраиваемого веса, $v_{ijн}$ – новое значение веса.

Первоначально β берется равным около 0,1 и затем постепенно уменьшается в процессе обучения.

Из (3.22) видно, что веса слоя Гроссберга будут сходиться к средним величинам от желаемых выходов, тогда как веса слоя Кохонена обучаются на средних значениях входов. Обучение слоя Гроссберга – это обучение с учителем, алгоритм располагает желаемым выходом, по которому он обучается. Обучающийся без учителя, самоорганизующийся слой Кохонена дает выходы в недетерминированных позициях. Они отображаются в желаемые выходы слоем Гроссберга.

3.8. Нейронные сети Хопфилда и Хэмминга

Среди различных конфигураций искуственных нейронных сетей (НС) встречаются такие, при классификации которых по принципу обучения, строго говоря, не подходят ни обучение с учителем [61], ни обучение без учителя [62]. В таких сетях весовые коэффициенты синапсов рассчитываются только однажды перед началом функционирования сети на основе информации об обрабатываемых данных, и все обучение сети сводится именно к этому расчету. С одной стороны, предъявление априорной информации можно расценивать, как помощь учителя, но с другой – сеть фактически просто запоминает образцы до того, как на ее вход поступают реальные данные, и не может изменять свое поведение, поэтому говорить о звене обратной связи с "миром" (учителем) не приходится. Из сетей с подобной логикой работы наиболее известны сеть Хопфилда и сеть Хэмминга, которые обычно используются для организации ассоциативной памяти. Далее речь пойдет именно о них.

Структурная схема сети Хопфилда приведена на рис.3.25. Она состоит из единственного слоя нейронов, число которых является одновременно числом входов и выходов сети. Каждый нейрон связан синапсами со всеми остальными нейронами, а также имеет один входной синапс, через который осуществляется ввод сигнала. Выходные сигналы, как обычно, образуются на аксонах.

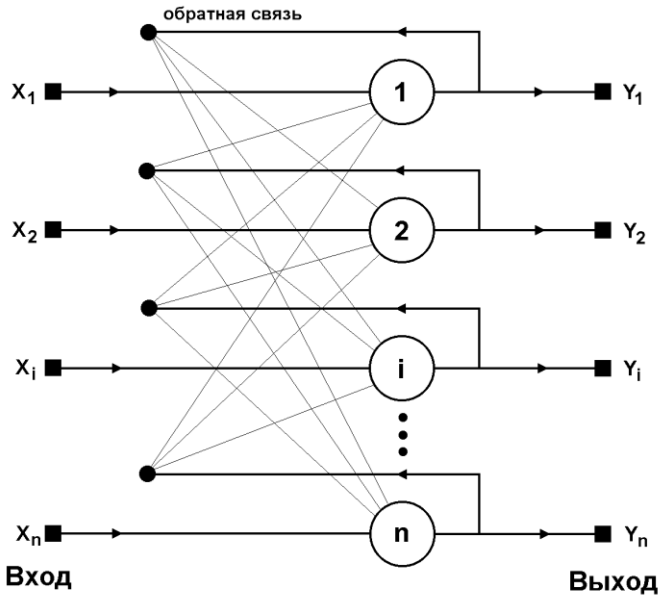


Рис.3.25. Структурная схема сети Хопфилда

Задача, решаемая данной сетью в качестве ассоциативной памяти, как правило, формулируется следующим образом. Известен некоторый набор двоичных сигналов (изображений, звуковых оцифровок, прочих данных, описывающих некие объекты или характеристики процессов), которые считаются образцовыми. Сеть должна уметь из произвольного неидеального сигнала, поданного на ее вход, выделить ("вспомнить" по частичной информации) соответствующий образец (если такой есть) или "дать заключение" о том, что входные данные не соответствуют ни одному из образцов. В общем случае, любой сигнал может быть описан вектором $\mathbf{X} = \{ x_i: i=0...n-1\}$, n – число нейронов в сети и размерность входных и выходных векторов. Каждый элемент x_i равен либо +1, либо -1. Обозначим вектор, описывающий k-ый образец, через $\mathbf{X^k}$, а его компоненты, соответственно, – x_i^k, k=0...m-1, m – число образцов. Когда сеть распозна́ет (или "вспомнит") какой-либо образец на основе предъявленных ей данных, ее выходы будут содержать именно его, то есть $\mathbf{Y} = \mathbf{X^k}$, где $\mathbf{Y}$ – вектор выходных значений сети: $\mathbf{Y} = \{ y_i: i=0,...n-1\}$. В противном случае, выходной вектор не совпадет ни с одним образцовым.

Если, например, сигналы представляют собой некие изображения, то, отобразив в графическом виде данные с выхода сети, можно будет увидеть картинку, полностью совпадающую с одной из образцовых (в случае успеха) или же "вольную импровизацию" сети (в случае неудачи).

На стадии инициализации сети весовые коэффициенты синапсов устанавливаются следующим образом [63]:

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, i \neq j \\ 0, \qquad i = j \end{cases} \tag{3.23}$$

Здесь i и j – индексы, соответственно, предсинаптического и постсинаптического нейронов; x_i^k, x_j^k – i-ый и j-ый элементы вектора k-ого образца.

Алгоритм функционирования сети следующий (p – номер итерации):

1. На входы сети подается неизвестный сигнал. Фактически его ввод осуществляется непосредственной установкой значений аксонов:

$$y_i(0) = x_i, \; i = 0...n-1, \tag{3.24}$$

поэтому обозначение на схеме сети входных синапсов в явном виде носит чисто условный характер. Ноль в скобке справа от y_i означает нулевую итерацию в цикле работы сети.

2. Рассчитывается новое состояние нейронов

$$s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p) \text{ , } j=0...n-1 \qquad (3.25)$$

и новые значения аксонов

$$y_j(p+1) = f\left[s_j(p+1)\right] \qquad (3.26)$$

где f – активационная функция в виде скачка, приведенная на рис.3.26а.

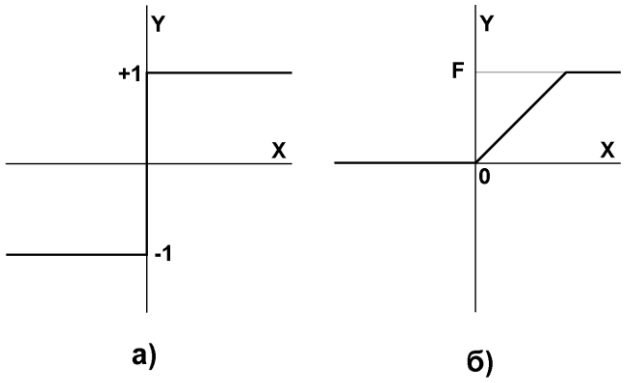


Рис.3.26. Активационные функции

3. Проверка, изменились ли выходные значения аксонов за последнюю итерацию. Если да – переход к пункту 2, иначе (если выходы застабилизировались) – конец. При этом выходной вектор представляет собой образец, наилучшим образом сочетающийся с входными данными.

Как говорилось выше, иногда сеть не может провести распознавание и выдает на выходе несуществующий образ. Это связано с проблемой ограниченности возможностей сети. Для сети Хопфилда число запоминаемых образов m не должно превышать величины, примерно равной 0.15•n. Кроме того, если два образа А и Б сильно похожи, они, возможно, будут вызывать у сети перекрестные ассоциации, то есть предъявление на входы сети вектора А приведет к появлению на ее выходах вектора Б и наоборот.

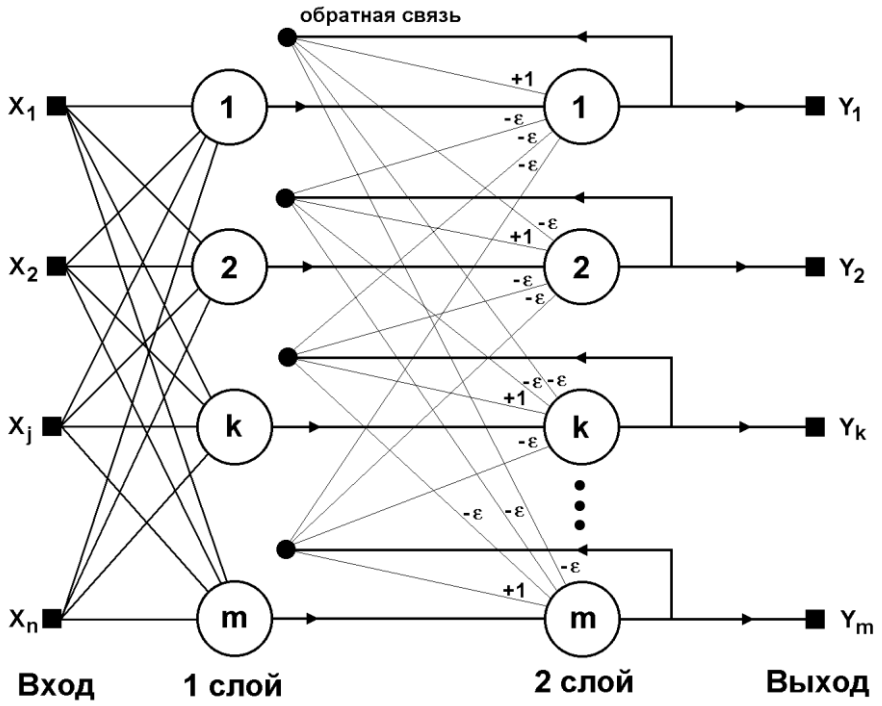


Рис.3.27 Структурная схема сети Хэмминга

Когда нет необходимости, чтобы сеть в явном виде выдавала образец, то есть достаточно, скажем, получать номер образца, ассоциативную память успешно реализует сеть Хэмминга. Данная сеть характеризуется, по сравнению с сетью Хопфилда, меньшими затратами на память и объемом вычислений, что становится очевидным из ее структуры (рис. 3.27).

Сеть состоит из двух слоев. Первый и второй слои имеют по m нейронов, где m – число образцов. Нейроны первого слоя имеют по n синапсов, соединенных со входами сети (образующими фиктивный нулевой слой). Нейроны второго слоя связаны между собой ингибиторными (отрицательными обратными) синаптическими связями. Единственный синапс с положительной обратной связью для каждого нейрона соединен с его же аксоном.

Идея работы сети состоит в нахождении расстояния Хэмминга от тестируемого образа до всех образцов. Расстоянием Хэмминга называется число отличающихся битов в двух бинарных векторах. Сеть должна выбрать образец с минимальным расстоянием Хэмминга до неизвестного входного сигнала, в результате чего будет активизирован только один выход сети, соответствующий этому образцу.

На стадии инициализации весовым коэффициентам первого слоя и порогу активационной функции присваиваются следующие значения:

$$w_{ik} = \frac{x_i^k}{2},\ i=0...n-1,\ k=0...m-1 \qquad (3.27)$$

$$T_k = n / 2,\ k = 0...m-1 \qquad (3.28)$$

Здесь x_i^k – i-ый элемент k-ого образца.

Весовые коэффициенты тормозящих синапсов во втором слое берут равными некоторой величине $0 < \varepsilon < 1/m$. Синапс нейрона, связанный с его же аксоном имеет вес +1.

Алгоритм функционирования сети Хэмминга следующий:

1. На входы сети подается неизвестный вектор $\mathbf{X} = \{x_i : i=0...n-1\}$, исходя из которого рассчитываются состояния нейронов первого слоя (верхний индекс в скобках указывает номер слоя):

$$y_j^{(1)} = s_j^{(1)} = \sum_{i=0}^{n-1} w_{ij} x_i + T_j,\ j=0...m-1 \qquad (3.29)$$

После этого полученными значениями инициализируются значения аксонов второго слоя:

$$y_j^{(2)} = y_j^{(1)},\ j = 0...m-1 \qquad (3.30)$$

2. Вычислить новые состояния нейронов второго слоя:

$$s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum_{k=0}^{m-1} y_k^{(2)}(p), k \neq j,\ j = 0...m-1 \qquad (3.31)$$

и значения их аксонов:

$$y_j^{(2)}(p+1) = f\left[s_j^{(2)}(p+1)\right],\ j = 0...m-1 \qquad (3.32)$$

Активационная функция f имеет вид порога (рис. 3.26б), причем величина F должна быть достаточно большой, чтобы любые возможные значения аргумента не приводили к насыщению.

3. Проверить, изменились ли выходы нейронов второго слоя за последнюю итерацию. Если да – перейди к шагу 2. Иначе – конец.

Из оценки алгоритма видно, что роль первого слоя весьма условна: воспользовавшись один раз на шаге 1 значениями его весовых коэффициентов, сеть больше не обращается к нему, поэтому первый слой может быть вообще исключен из сети (заменен на матрицу весовых коэффициентов), что и было сделано в ее конкретной реализации, описанной ниже.

Приведена тестовая программа для проверки сети. Здесь конструируется сеть со вторым слоем из пяти нейронов, выполняющая распознавание пяти входных образов, которые представляют собой схематичные изображения букв размером 5 на 6 точек (см.рис.3.28а). Обучение сети фактически сводится к загрузке и запоминанию идеальных изображений. Затем на ее вход поочередно подаются зашумленные на 8/30 образы (см.рис.3.28б), которые она успешно различает.

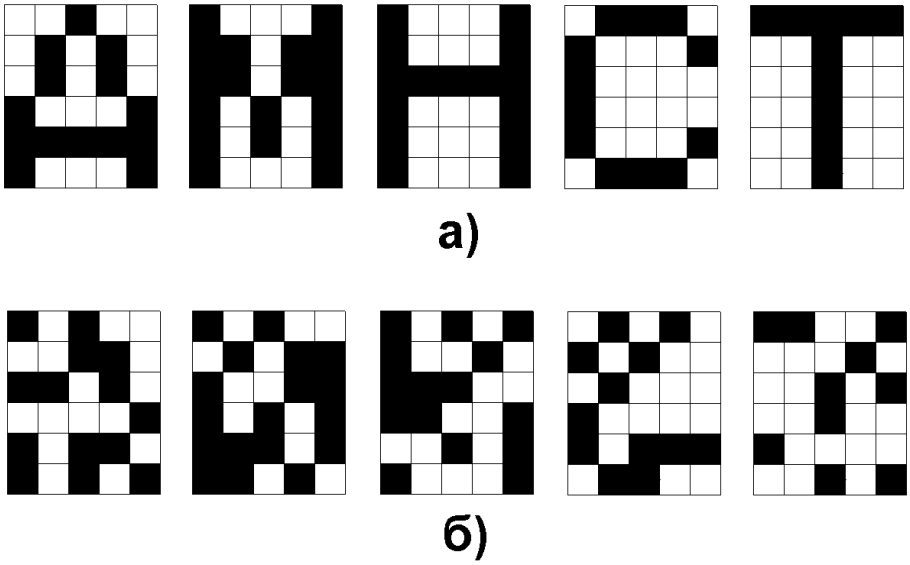


Рис. 3.28. Образцовые и тестовые образы

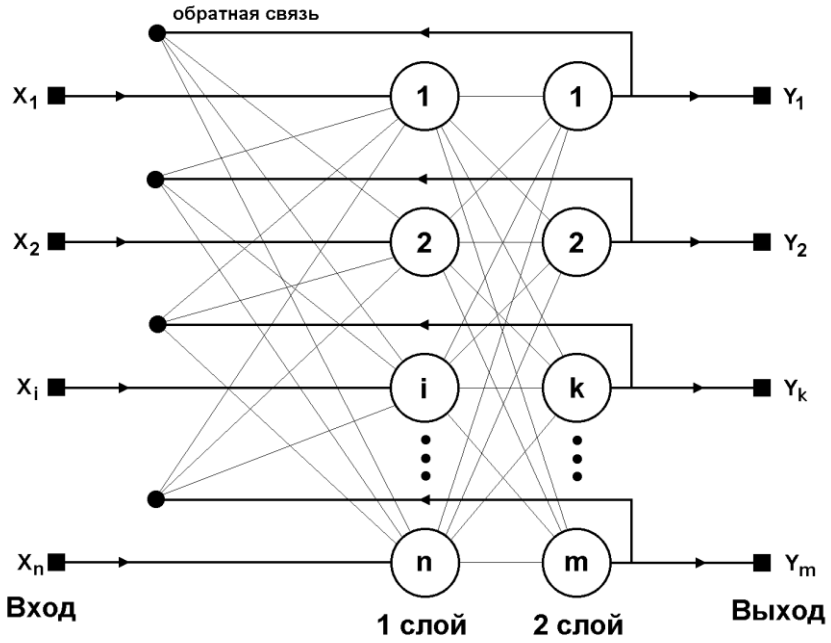


Рис.3.29. Структурная схема ДАП

В проект кроме файлов NEURO_HN и NEUROHAM входят также SUBFUN и NEURO_FF, описанные в [1]. Программа тестировалась в среде Borland C++ 3.1.

Предложенные классы позволяют моделировать и более крупные сети Хэмминга. Увеличение числа и сложности распознаваемых образов ограничивается фактически только объемом ОЗУ. Следует отметить, что обучение сети Хэмминга представляет самый простой алгоритм из всех рассмотренных до настоящего времени алгоритмов.

Обсуждение сетей, реализующих ассоциативную память, было бы неполным без хотя бы краткого упоминания о двунаправленной ассоциативной памяти (ДАП). Она является логичным развитием парадигмы сети Хопфилда, к которой для этого достаточно добавить второй слой. Структура ДАП представлена на рис.3.29. Сеть способна запоминать пары ассоциированных друг с другом образов. Пусть пары образов записываются в виде векторов $X^k = \{x_i^k : i=0...n-1\}$ и $Y^k = \{y_j^k : j=0...m-1\}$, $k=0...r-1$, где r – число пар. Подача на вход первого слоя некоторого вектора $P = \{p_i : i=0...n-1\}$ вызывает образование на входе второго слоя некоего другого вектора $Q = \{q_j : j=0...m-1\}$, который затем снова поступает на вход первого слоя. При каждом таком цикле вектора на выходах обоих слоев приближаются к паре образцовых векторов, первый из которых – X – наиболее походит на P, который был подан на вход сети в самом начале, а второй – Y – ассоциирован с ним. Ассоциации между векторами кодируются в весовой матрице $W^{(1)}$ первого слоя. Весовая матрица второго слоя $W^{(2)}$ равна транспонированной первой $(W^{(1)})^T$. Процесс обучения, также как и в случае сети Хопфилда, заключается в предварительном расчете элементов матрицы W (и соответственно W^T) по формуле:

$$w_{ij} = \sum_k x_i y_j,\ i = 0...n-1,\ j = 0...m-1. \qquad (3.33)$$

Эта формула является развернутой записью матричного уравнения

$$\mathbf{W} = \sum_k \mathbf{X}^{\mathrm{T}} \mathbf{Y} \qquad (3.34)$$

для частного случая, когда образы записаны в виде векторов, при этом произведение двух матриц размером соответственно [n*1] и [1*m] приводит к (3.33).

В заключении можно сделать следующее обобщение. Сети Хопфилда, Хэмминга и ДАП позволяют просто и эффективно

разрешить задачу воссоздания образов по неполной и искаженной информации. Невысокая емкость сетей (число запоминаемых образов) объясняется тем, что, сети не просто запоминают образы, а позволяют проводить их обобщение, например, с помощью сети Хэмминга возможна классификация по критерию максимального правдоподобия [663]. Вместе с тем, легкость построения программных и аппаратных моделей делают эти сети привлекательными для многих применений.

3.9. Нейросетевое распознавание объектов на изображениях топологических слоев интегральных микросхем

Составной частью современной технологии проектирования и производства интегральных сем (ИС) являются системы технического зрения. Несмотря на ряд успехов в области идентификации и распознавания объектов на основе цифровой обработки изображений, к числу нерешенных можно отнести проблему использования теории нейронных сетей (НС) для решения задач обработки изображений топологии ИС. Основные нерешенные задачи здесь состоят в адекватном отображении предметной области на нейронную систему, выборе моделей используемых НС и их интеграции в единую интеллектуальную систему.

При обработке изображений топологии требуется по некоторым признакам выделять некоторые однородные области изображения, причем, как правило, подобие нечеткое и часто нарушается. Этапы предварительной обработки изображения позволяют уменьшить влияние искажений на процесс распознавания [1, 2]. Тем не менее, имеет место распознавание в условиях неполной и нечеткой информации. Наиболее подходят для ее решения нейросетевые технологии, НС при этом . выступает в роли классификатора. Применение НС в задачах обработки визуальной информации обосновывается также свойством обучаемости или адаптивности НС к новым задачам, при этом сохраняются архитектура сети и алгоритм ее функционирования.

Концепция нейросетевой обработки и идентификации видеоизображений предполагает использование следующих подходов к обработке изображений топологических слоев ИС:

1) проблемно-ориентированной предварительной обработки, сохраняющей информационные признаки топологических объектов,

что позволяет сократить число связей НС, упростить и ускорить процесс обучения;

2) выделения (идентификации) объекта на изображении для нейросетевой обработки за счет использования информации об иерархии признаков, что сокращает затраты времени на обработку (осуществляется поиск лишь в идентифицированной области);

3) использования набора классификаторов, в котором по результатам классификации на тестовой выборке производится выбор наилучшей модели классификатора для обработки всех изображений.

Существует ряд НС, позволяющих решать задачу распознавания образов с определенной степенью точности [63-69]: многослойный персептрон, сеть на основе радиальной базисной функции, сети АРТ, сети Хопфилда, самоорганизующееся отображение Кохонена, неокогнитрон. Наиболее подходящим для этой цели является неокогнитрон. Для повышения точности иерархического распознавания и увеличения производительности НС исследования проводятся в следующих направлениях: модификация правил выделения признаков за счет введения новых дополнительных инвариантов относительно искажений [70]; модификация структуры и принципов послойной обработки [71]; разработка алгоритмов самообучения [72-74].

В этом параграфе рассмотрены нейросетевая реализация названных выше второго и третьего подходов, а также представлена структура системы, реализующей нейросетевую технологию распознавания объектов топологии интегральных микросхем.

3.10. Выделение объекта на изображении

Нейросетевой подход к идентификации (поиску) объектов предполагает разработку НС, алгоритма ее обучения и технологии ее применения. Поиск элементов топологии на основе НС состоит в сканировании изображения окном, размер которого равен размеру обучающего изображения, и определения по отклику НС, является ли изображение внутри окна элементом. Выбор архитектуры НС определяется типом обрабатываемого слоя и формой его представления на разных этапах обработки и осуществляется, как правило, путем экспериментальных просчетов и оценок их качества проектировщиком ИС. Учитывая многообразие топологических слоев предлагается для решения задачи идентификации совокупность НС,

включающая многослойный персептрон, НС с радиальной базисной функцией и неокогнитрон, которые апробированы и рекомендуются для использования.

Многослойный персептрон состоит из множества слоев нейронных элементов, причем известно, что достаточно трех слоев для создания сколь угодно сложной решающей граничной поверхности в пространстве обучающих образов. При ее построении количество n нейронов первого слоя определяется в соответствии с количеством информативных признаков (яркостные значения (r, g, b), семантические дескрипторы и др.). Количество нейронов промежуточных слоев определяется эмпирически. Выходной слой имеет p нейронов, активность которых определяет принадлежность изображения окна к элементу в виде некоторой функции, значение которой сравнивается с некоторым заданным порогом. Каноническим методом обучения такой сети считается алгоритм обратного распространения ошибки.

При использовании НС с радиальной базисной функцией не существует проблемы выбора количества скрытых нейронов, так как они определяются количеством групп (кластеров), на которые разбивается пространство обучающих образов при помощи радиальной базисной функции, что положительно отличает их от сетей обратного распространения ошибки.

Неокогнитрон (рис. 3.30) позволяет выполнить при поиске элементов сравнение изображения в окне сканирования с эталонным образцом с учетом иерархии признаков, формируемой в процессе обучения.

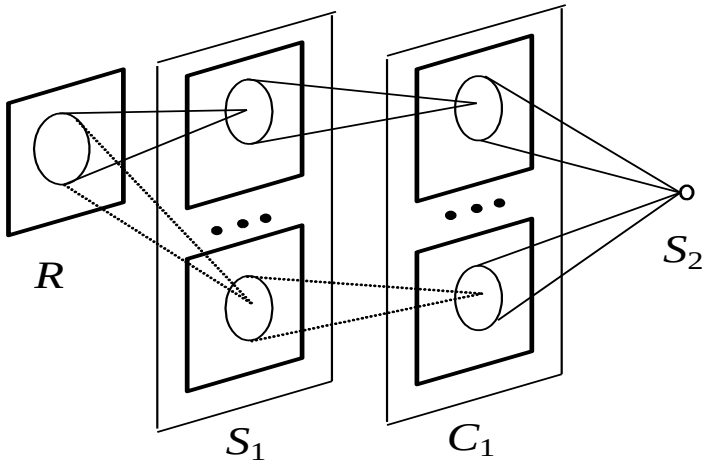


Рис. 3.30. Используемая архитектура сети

Слой *R* является входным. S_1-слой предназначен для выделения общих признаков, встречающихся у всех элементов ИС, таких, как линейные границы перепадов яркостей различной ориентации. Все подслои этого слоя состоят из нейронов с одинаковыми размерами подгрупп рецепторного поля 4×4 нейрона. Обучающие образы для этих подслоев изображены на рис. 3.31.

Рис. 3.31. Обучающие образы S_1-слоя

C_1-слой предназначен для обобщения признаков, выделяемых в S_1-слое, а рецепторные подгруппы его нейронов организованы таким образом, чтобы объединять такие парные признаки, как перепады яркости с темного на светлый или наоборот (объединены на рис. 3.31 горизонтальной прямоугольной скобкой). В результате получаем четыре подслоя в C_1-слое. Размер подгрупп рецепторного поля выбран такой, чтобы активность нейронов из этого слоя была инвариантна по отношению к малым сдвигам признаков, выделяемых в предыдущем слое, и составляет поле 2×2. S_2-слой предназначен для выделения совокупности признаков, присущих одному конкретному элементу ИС. Он формирует выходное значение сети и состоит из одного нейрона, а размеры подгрупп рецепторного поля этого нейрона совпадают с размерами подслоев C_1-слоя.

Входными данными для тестирования являлись полутоновые фрагменты фотоизображения поликремниевого слоя ИС (рис. 3.32). Идентифицировались два типа фрагментов. Соответственно производились два варианта обучения НС, обучающие выборки для которых показаны на рис. 3.33.

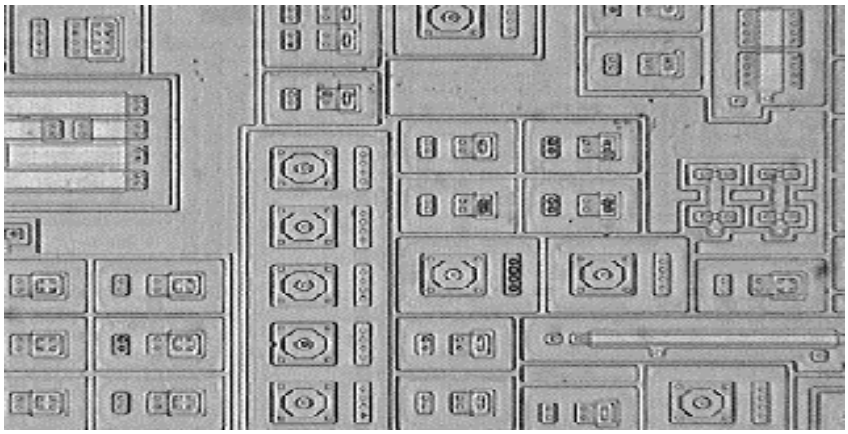
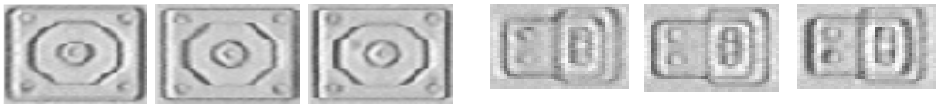


Рис. 3.32. Фрагмент поликремниевого слоя ИС



Вариант 1 Вариант 2

Рис. 3.33. Обучающие образы S_2-слоя НС

Результаты тестирования показали, что метод формирования критерия схожести двух изображений с помощью описанной НС является лучшим по сравнению с методом расчета корреляции этих двух изображений по формуле:

$$\mu = \frac{\sum_{x,y} a_{x,y} \cdot b_{x,y}}{\sqrt{\sum_{x,y} a_{x,y}^2 \cdot \sum_{x,y} b_{x,y}^2}}, \qquad (3.35)$$

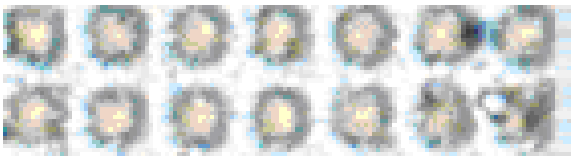
где x, y – координаты точки сравниваемых изображений; a, b – значение цвета точки для усредненного изображения и изображения в скользящем окне соответственно (табл. 3.3) . Этот результат был достигнут вследствие того, что НС является инвариантной к искажениям формы и яркости изображения распознаваемого объекта, а также требует меньшего количества обучающих образцов для достижения необходимой точности идентификации.

Для демонстрации пригодности персептрона ниже приведены тесты с сетью с сигмоидной функцией активации для идентификации контактных площадок (КП) двух типов (рис. 3.34). НС состояла из 35 входных нейронов, скрытого слоя из 3.34 нейронов и выходного слоя из одного нейрона (для поиска каждого типа объекта формируется своя сеть).

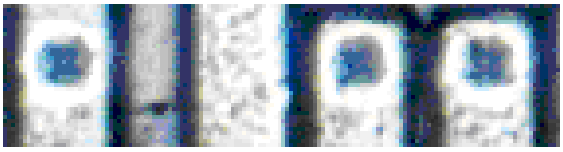
Таблица 3.3

Результаты тестирования

Объем обучающей выборки	Точность идентификации			
	Корреляционный метод		Нейронная сеть	
	Вариант 1	Вариант 2	Вариант 1	Вариант 2
1	63,1	67,4	87,0	89,0
3	72,3	73,6	96,3	97,0
6	81,4	84,2	98,5	99,0



а)



б)

Рис. 3.34. Примеры исходных обучающих изображений КП: *а*) первый тип; *б*) второй тип

В табл. 3.4 приведены результаты идентификации на пяти тестах, время обучения НС, которые показывают стабильность и высокое качество идентификации.

Таблица 3.4

Результаты тестирования

Объем обучающей выборки	Количество тестируемых КП	Число итераций обучения	Суммарная квадратич-ная ошибка	Процент идентифи-кации
18	20	2868	0,098	95
30	20	4225	0,099	100
40	50	3656	0,099	100
70	70	19906	0,099	100
70	70	18091	0,099	100

Для эффективной идентификации топологических объектов целесообразно сформировать библиотеку элементов, которая включает в себя различные варианты их расположения (ориентации). Так, предварительный анализ изображений показал целесообразность определить следующие типы элементов металлизации, показанные на рис. 3.35.

Под элементами металлизации понимается часть изображения, которая вмещает в себя изображение типа: контактная площадка, пересечение дорожек, дорожка. Каждый тип в свою очередь может быть представлен множеством его модификаций. К примеру, в зависимости от расположения на дорожке, КП могут быть левые или правые (расположены на левом или правом концах конце горизонтальной дорожки) и др. (рис. 3.36).

а) *б*) *в*) *г*) *д*)

Рис. 3.35. Типы элементов изображения металлизации:
а) КП; *б)* часть дорожки; *в)* изгиб дорожки; *г)* дефект дорожки; *д)* пересечение дорожек текущего и предыдущего слоя

а) *б*) *в*) *г*) *д*)

Рис. 3.36. Виды КП изображения металлизации:
а) верхняя; *б*) правая; *в*) левая; *г*) нижняя; *д)* нижняя правая

Результаты идентификации КП различных видов трехслойным персептроном с числом входов, равным числу пикселей эталонного изображения КП (40×40), скрытым слоем из 10 нейронов и выходным слоем, количество нейронов которого определяется числом видов идентифицируемых КП (до 10) показали, что большинство КП были правильно идентифицированы (77 из 83 КП, что составляет 93,9 %).

Типовой алгоритм идентификации (поиска) объектов на изображениях топологических слоев ИС имеет следующий вид.

1. Начало. Подготовить (выбрать из БД) образцы изображений искомых объектов.
2. Сформировать (выбрать из БД) архитектуру НС для поиска элементов на изображении ИС и выполнить ее настройку (обучение).
3. Сохранить информацию об архитектуре НС в БД в виде таких ее полей, как размер рецепторного слоя и матрицы весовых коэффициентов.
4. Выполнить сканирование изображения топологии ИС окном соответствующего размера и определить координаты мест расположения искомых объектов топологии.
5. Конец.

3.10.1.Анализ и классификация

С целью улучшения характеристик обработки информации и точности распознавания был проведен анализ структуры неокогнитрона и разработана архитектура многослойной НС, которая реализует многоуровневый процесс распознавания. В основе правила активации нейронов лежит метод сравнения матриц яркостей изображений, который позволяет сравнивать матрицы с нечетким позиционированием соответствующих элементов матриц [15]. При этом функция активации определяется формулой

$$c = 1 + \frac{n\sum_{x,y}\left(w_{x,y}a'_{x,y}\right) - \sum_{x,y}w_{x,y}\sum_{x,y}a'_{x,y}}{\sqrt{\left(n\sum_{x,y}\left(w_{x,y}a'_{x,y}\right) - \left(\sum_{x,y}w_{x,y}\right)^2\right)\left(n\sum_{x,y}\left(w_{x,y}a'_{x,y}\right) - \left(\sum_{x,y}a'_{x,y}\right)^2\right)}}, \quad (3.36)$$

где n – количество точек в эталонной матрице; $w_{x,y}$ – значения точек в эталонной матрице; $a'_{x,y}$ – значения соответствующих точек на анализируемом изображении,

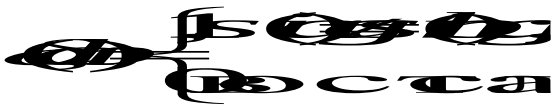
$$a'_{x,y} = \begin{cases} 0, \left[\min\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (a_{x+rx,y+ry}); \max\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (a_{x+rx,y+ry}) \right] \cap \\ \cap \left[\min\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (w_{x+rx,y+ry}); \max\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (w_{x+rx,y+ry}) \right] \neq \varnothing; \\ a_{x,y}, \left[\min\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (a_{x+rx,y+ry}); \max\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (a_{x+rx,y+ry}) \right] \cap \\ \cap \left[\min\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (w_{x+rx,y+ry}); \max\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (w_{x+rx,y+ry}) \right] = \varnothing. \end{cases}$$

Параметр R – радиус геометрических искажений, определяющий максимально допустимое смещение пикселей эталона на изображении (образе), вычисляется по формуле R= [0,3×1/F], где $F = \frac{F_h + F_v}{2}$,

$$F_h = \frac{1}{X}\left(1 + \frac{1}{Y}\sum_{x=1,y=1}^{X-1,Y} \delta(g^h_{x,y}, g^h_{x+1,y})\right),$$

$$F_v = \frac{1}{Y}\left(1 + \frac{1}{X}\sum_{x=1,y=1}^{X,Y-1} \delta(g^v_{x,y}, g^v_{x,y+1}),\right),$$

$g^h_{x,y}$, $g^v_{x,y}$ – значения вертикального и горизонтального градиентов в точке (x, y),



В результате применения (3.36) получаем значения в интервале $c \in [0; 2]$, где значения $c \in [0; 1)$ говорят об обратной корреляции, значения с = 1 – об отсутствии корреляции, а значения $c \in (1; 2]$ – о прямой корреляции.

В рассматриваемой здесь постановке задачи распознавания предполагается, что существует набор изображений объектов, в котором каждый объект представлен несколькими изображениями, отличающимися друг от друга. Чтобы сформировать такое множество, разделим все изображения объектов на три множества: обучающее, тестовое для обучения и тестовое. Обучающее множество – это множество изображений объектов, которое формируется как результат алгоритма формирования обучающего

множества. Главной характеристикой является его репрезентативность, т.е. достаточность различных представлений объекта на изображении для того, чтобы классификатор смог распознать все остальные возможные вариации представления объекта. Тестовое множество для обучения – это некоторая небольшая часть общего множества изображений, которая используется при формировании обучающего множества. Будем считать, что тестовое множество для обучения является репрезентативным, но в то же время избыточным. Избыточность заключается во включении в множество таких элементов, которые можно исключить из процесса обучения. Тестовое множество – это большая часть общего множества изображений.

Теперь для формирования обучающего множества выполняются следующие действия.

1. Выбрать в обучающее множество по одному экземпляру изображений каждого класса из тестового множества для обучения.

2. Произвести обучение.

3. Выполнить распознавание на тестовом множестве для обучения.

4. Добавить в обучающее множество экземпляры изображений каждого класса, которые не были распознаны.

5. Повторить пп. 2-4 до тех пор, пока не будет достигнут требуемый уровень распознавания.

После построения обучающего множества для каждого слоя решается задача кластеризации входных данных слоя, к которой сводится задача обучения слоя.

Поиск лучшей архитектуры НС осуществлялся следующим методом. Для различных вариантов архитектуры она обучалась на обучающем множестве, затем тестировалась на тестовом множестве для обучения. По результатам тестов выбиралась лучшая архитектура.

Структура системы обработки изображений показана на рис. 3.37.

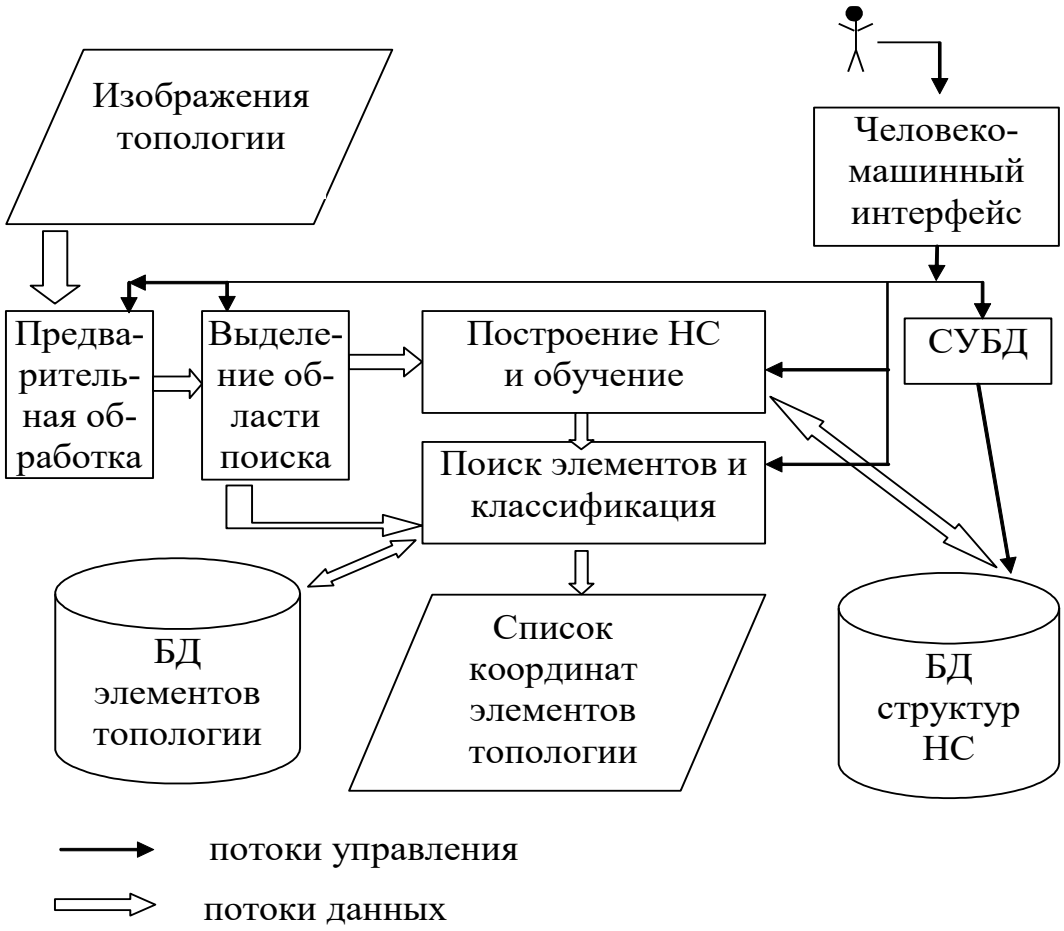


Рис. 3.37. Структура нейросетевой системы

Разработаны алгоритмы идентификации и распознавания объектов топологии на основе нейросетевого подхода с использованием следующих НС: многослойного персептрона, НС с радиальной базисной функцией и неокогнитрона, которые обеспечивают стабильность и снижение вычислительной сложности идентификации при нечеткой информации об объектах топологии. Предложенные алгоритмы является составной частью нейросетевой технологии обработки топологии ИС и используется как для поиска собственно объектов топологии (которые затем классифицируются, и выполняется их анализ на наличие дефектов), так и для поиска специфических областей топологии для проведения их детального анализа.

3.10.2. Методы видеонаблюдения, сегментации и сопровождения движущихся объектов

Главным признаком, положенным в основу разрабатываемых методов видеонаблюдения, сегментации и сопровождения, является движение. Его наиболее информативная оценка – вектора движения.

Основой определения векторов движения является уравнение оптического потока:

$$< \nabla \mathbf{L}, \mathbf{V} > + I_t = 0, \qquad (3.37)$$

где $\nabla \mathbf{L} = (\frac{\partial L}{\partial x}, \frac{\partial L}{\partial y})$ – яркостный вектор-градиент по пространственным координатам; $I_t = \frac{\partial L}{\partial t}$ – производная яркости по времени; $\mathbf{V} = (\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t})$ – вектор движения.

Анализ уравнения оптического потока (3.37) показывает:

1. Уравнение является плохо обусловленным.

2. Однозначное определение вектора движения возможно только в случае, если компоненты яркостного вектора - градиента $\nabla \mathbf{L} = (\frac{\partial L}{\partial x}, \frac{\partial L}{\partial y})$ отличны от нуля (имеют место изменения яркости по горизонтали и вертикали). В случае однородной, нетекстурированной поверхности или текстуры только в одном направлении достоверную оценку векторов найти нельзя.

3. Уравнение (3.37) получено из предположения о постоянной яркости точки (пикселя) при ее движении. Подсветка, тени, блики, прозрачные и зеркальные поверхности нарушают это утверждение и приводят к ошибкам при определении векторов движения.

Вектора движения, методы и алгоритмы их определения были разработаны для стандартов видеокомпрессии MPEG 1,2,4 для обработки изображений студийного качества, где соблюдаются требования текстурированности и постоянства яркости.

При работе в сложных условиях наблюдения с определенным в рамках решаемой задачи классом объектов требования существования пространственных яркостных производных и постоянства яркости при движении точки (пикселя) вдоль траектории

нарушаются. Изображение объектов интереса имеет низкую текстурированность. При видеонаблюдении на открытом воздухе высока вероятность изменения уровня освещенности, появления солнечных бликов, теней и др. Не соблюдение обоих требований порождает появление аномальных векторов – векторов, не отражающих реальное движение в кадре.

Тяжесть последствий от аномальных векторов движения в прикладных ТВ системах значительна: рассегментация объектов интереса, снижение точности моделей движения, потеря объекта при сопровождении и др. Это определяет критерий оценки эффективности методов и алгоритмов определения векторов движения в прикладных ТВ системах – уровень достоверности:

$$K_{\partial} = 1 - \frac{Q_{ан}}{Q},$$

где Q – общее число найденных векторов, $Q_{ан}$ – число аномальных векторов в общем числе найденных

Проведенные исследования показали, что даже при использовании корреляционных методов (наиболее робастных по данным литературы) с алгоритмом полного перебора для поиска лучшего соответствия уровень достоверности $K_{\partial} = 0{,}6$. Этот уровень недостаточен для эффективного применения векторов движения в прикладных ТВ системах. Необходима разработка новых методов определения векторов движения, учитывающих особенности объектов интереса, и свойства видеоданных, получаемых в сложных условиях наблюдения.

В методе совмещения блоков задачу определения векторов движения решают путем минимизации целевой функции, характеризующей степень соответствия (совпадения) двух блоков, на множестве различных положений обрабатываемого блока в области поиска. Результаты решения этой задачи определены видом целевой функции, которая зависит от уровня детальности изображения в блоке (таблица 3.5).

Под уровнем детальности изображения в блоке будем понимать:

$$D(k,l) = \sum_{j=1}^{M}\sum_{i=1}^{N} \Lambda(x_k + i, y_l + j) \ , \qquad (3.38)$$

где $\Lambda(x,y)$ – яркость пикселя в препарате, полученном из исходного изображения $L(x,y)$ в результате операции подчеркивания высокочастотной составляющей (пространственного дифференцирования, многомасштабного морфологического дифференцирования и др.); x_k, y_l – координаты левого верхнего угла блока изображения; k,l – номер блока по горизонтали и вертикали; N,M – число пикселей блока по горизонтали и вертикали.

На основе проведенных экспериментальных исследований и регрессионного анализа найдена функция достоверности, позволяющая определить априорную вероятность корректного нахождения вектора движения $P_{c\_v}(k,l)$ по модифицированной оценке уровня детальности в блоке $D_M(k,l)$:

$$P_{c\_v}(k,l) = 1 - \exp\left[-\eta * D^4{}_M(k,l)\right], \qquad (3.39)$$

где $\eta = 0{,}1$ – параметр модели, численное значение которого определено на основе экспериментальных данных методом нелинейного программирования.

Модифицированная оценка уровня детальности в блоке D_M:

$$D_M = \frac{D(k,l) - D_{\min}}{D_{\min}},$$

где $D_{\min}$ – минимальный уровень детальности в блоке, обусловленный шумами

$$D_{\min} = \operatorname{moda}\{D(k,l)\}, \quad k = 1..KK, l = 1..LL,$$

где $KK * LL$ – число блоков в изображении.

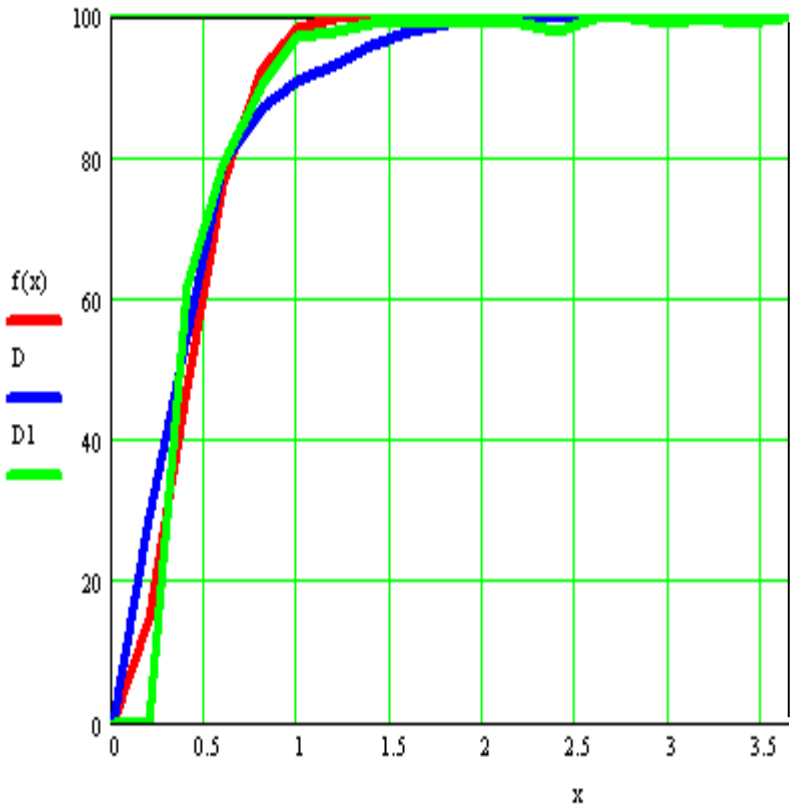
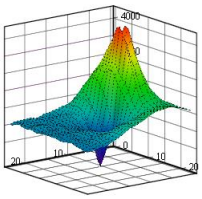
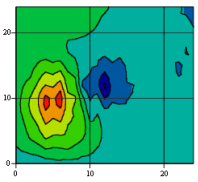
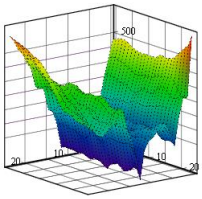
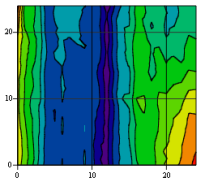
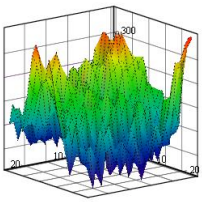
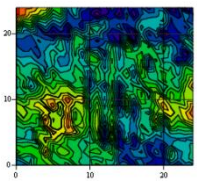


Рис. 3.38 Вид аналитической кривой функции значимости $f(x)$ и экспериментальных зависимостей вероятности существования значимого вектора от уровня модифицированной оценки абсолютной межкадровой разности, полученных для тестовых сюжетов

Таблица 3.5

$D(k,l)$ ● $D_m(k,l)$	Описание блока	Рельеф (ось X – смещение блока по горизонтали; ось Y – смещение блока по вертикали; ось Z – значение целевой функции)	Линии равного уровня (ось X – смещение блока по горизонтали; ось Y – смещение блока по вертикали)	Характер функции
1546 ● 7	Блок изображения объекта «самолет» тестовая последовательность «Самолеты»			Унимодальная целевая функция с ярко выраженным минимумом
137 ● 1	Блок фона «море» тестовая последовательность «Корабль - катер»			Унимодальная целевая функция с существенно плоским участком в области минимума («овраг»)
181 ● 1	Блок фона «небо» тестовая последовательность «Самолеты»			Существенно мультимодальная целевая функция с плохо выраженным глобальным минимумом

Определение векторов движения для всех блоков кадра или области поиска является избыточным: если в блоке кадра t нет значимых изменений относительно кадра $t-1$, то с большой вероятностью вектор движения равен нулю. Искать векторы движения целесообразно только в тех блоках, где произошли какие-либо изменения.

На основе экспериментальных исследований и регрессионного анализа найдена функция значимости, позволяющая определить

априорную вероятность существования значимого (ненулевого) вектора движения $P_{z\_v}(k,l)$ на основе модифицированной оценки абсолютной межкадровой разности в блоке MAD_M:

$$P_{z\_v}(k,l) = 1 - \exp\left[-\beta * MAD_M^2(k,l)\right],$$

где $\beta = 4$ – параметр модели, численное значение которого найдено на основе экспериментальных данных методом нелинейного программирования (рис.3.38).

Модифицированная оценка уровня абсолютной межкадровой разности:

$$MAD_M(k,l) = \left| \frac{MAD(k,l) - MAD_{\min}}{MAD_{\min}} \right|,$$

собственно уровень абсолютной межкадровой разности

$$MAD(k,l) = \sum_{j=1}^{N} \sum_{i=1}^{M} \left| L(x_k + j, y_l + i, t) - L(x_k + j, y_l + i, t-1) \right|$$

где $L(\ )$ – яркости пикселя в текущем t и предыдущем $t-1$ кадрах; $MAD_{\min}$ – оценка абсолютной межкадровой разности блока, обусловленная присутствием шумов:

$$MAD_{\min} = \text{moda}\{MAD(k,l)\} \; k = 1..KK, l = 1..LL$$

На основе найденных функций предложен метод определения векторов движения с учетом их априорных оценок достоверности и значимости.

На множестве всех блоков изображения G сформировано два нечетких множества. Первое G_{DH} – множество блоков с «высокой детальностью». В качестве функции принадлежности к этому блоку использована функция достоверности (3.39) $\mu_{G_{DH}} = P_{c\_v}$. Второе нечеткое множество G_{MH} – блоки « с высоким уровнем абсолютной межкадровой разности» с функцией значимости (3.40) в качестве функции принадлежности $\mu_{G_{MH}} = P_{z\_v}$. Интересующая совокупность блоков - пересечение этих двух множеств – нечеткое множество G_{DMH}.

Пересечение нечетких множеств выполняют в соответствии с t-нормой: «вероятностное пересечение» – норма задана перемножением функций принадлежности $T(G_{DH}, G_{MH}) = \mu_{G_{DH}} * \mu_{G_{MH}}$.

Вектора движения ищут для блоков, принадлежащих α-сечению нечеткого множества G_{DMH}. В результате поиска им присваивают

трехкомпонентный вектор движения $(v_{xkl}, v_{ykl}, p_{kl})^{\mathrm{T}}$, где значения p_{kl} – это вероятность верного определения вектора движения, определяемая функцией достоверности (3.39). Блокам, принадлежащим α-сечению только одного нечеткого множества G_{DH} (это блоки с высоким уровнем детальности, но низкой абсолютной межкадровой разностью) без выполнения процедуры поиска присваивают нулевые векторы движения.

В типичных для прикладных ТВ систем сюжетах при установленном уровне достоверности вектора движения 0,95 для дальнейшего анализа оставляют в среднем 20% блоков изображения; собственно процедуру поиска вектора движения выполняют в среднем для 5%-10% блоков изображения (рис. 3.39). Найденная для каждого вектора вероятность верного определения (оценка достоверности) p_{kl} представляет самостоятельную ценность: ее учет на дальнейших этапах обработки позволяет реализовать взвешенную оценку признака движения.

Таким образом, предложенный метод позволяет находить трехкомпонентные вектора движения с заданным уровнем достоверности, одновременно, снижая вычислительные затраты по определению поля векторов движения в 10 – 20 раз.

Необходимость обработки изображений объектов, обладающих медленным движением (видимая скорость движения в плоскости кадра менее 1 пикселя), движением со скоростью, не кратной целому числу пикселей, а также различение объектов с близкими скоростями движения требует субпиксельной оценки векторов движения.

Для субпиксельной оценки векторов движения предложено два метода: интерполяционный метод и метод, основанный на многомасштабной межкадровой разности.

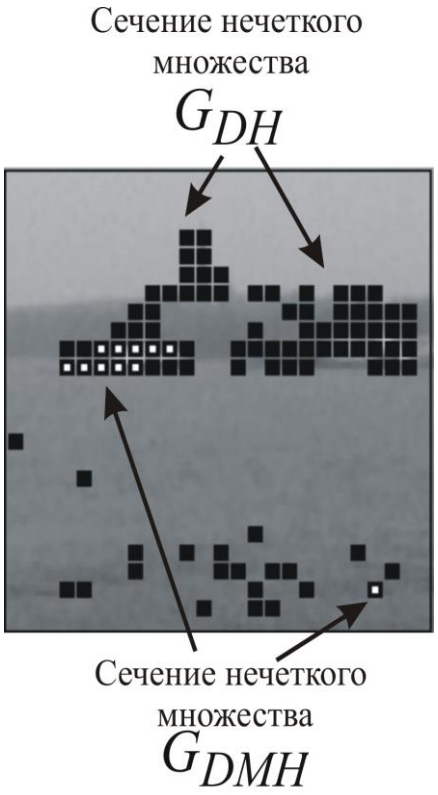


Рис. 3.39 Сечения нечетких множеств G_{DH} и G_{DMH}

Интерполяционный метод использует несимметрию в области точки оптимума целевой функции, минимизируемой в процессе нахождения векторов движения.

Метод, основанный на многомасштабной межкадровой разности, предполагает использование для определения векторов движения N кадров. Вектор движения находят в (N-1) паре кадров: k-ая пара включает в себя кадр t и кадр t-k, k=1..N. Результатом является (N-1) оценка вектора движения $\mathbf{v}_k^0, k = 1..N-1$. Для обеспечения соизмеримости полученных оценок выполняют пересчет:

$$V_{kx} = \frac{V_{kx}^0}{d} \; ; \quad V_{ky} = \frac{V_{ky}^0}{d},$$

где (V_{kx}, V_{ky}) – k-ый приведенный вектор пакета, (V_{kx}^0, V_{ky}^0) – вектор, найденный методом совмещения блоков по кадрам t и t-k, d – расстояние между кадрами t и t-k.

Предложено два варианта получения субпиксельной оценки $\mathbf{V}_p = (V_{px}, V_{py})$ на основе логической фильтрации полученного пакета векторов. Минимально отличный от других вектор пакета:

$$L_{V_{kx}} = \sum_{i=1}^{N} |V_{kx} - V_{ix}|; V_{px} = \arg\min_{V_{kx}} \{L_{V_{kx}}\};$$

$$L_{V_{ky}} = \sum_{i=1}^{N} |V_{ky} - V_{iy}|; V_{py} = \arg\min_{V_{ky}} \{L_{V_{ky}}\}$$

или медиана пакета:

$$V_{px} = V_{(N/2)x} \text{ при условии } V_{1x} < V_{2x} < .... < V_{Nx};$$

$$V_{py} = V_{(N/2)y} \text{ при условии } V_{1y} < V_{2y} < .... < V_{Ny}.$$

Экспериментальное исследование предложенных методов выполнено по двум критериям: уровень достоверности и точность. При оценке уровня достоверности аномальным считают вектор, отличный от корректного на заданную величину $\|\mathbf{V} - \mathbf{V}_{cor}\| > \Delta\xi$. Учитывая субпиксельный характер оценки вектора движения, $\Delta\xi = \pm 0{,}1$пикселя. Точность – величина СКО в полученной выборке верных векторов движения.

Согласно полученным данным установлено, наиболее эффективным методом является метод на основе многомасштабной межкадровой разности с минимально отличным вектором в качестве субпиксельной оценки. Его точность 0,07 пикселя. Уровень достоверности $K_д = 0{,}8$. Полученный уровень достоверности поля векторов движения на 20% выше по отношению к полю, найденному методом полного перебора.

Предложенные методы оценки векторов движения в значительной степени устраняют противоречие между жесткими требованиями основного уравнения оптического потока (3.37) и особенностями видеоматериала, полученного в сложных условиях наблюдения. Обеспеченный уровень достоверности и точность определения поля векторов движения снимают ограничения на их применение в прикладных ТВ системах.

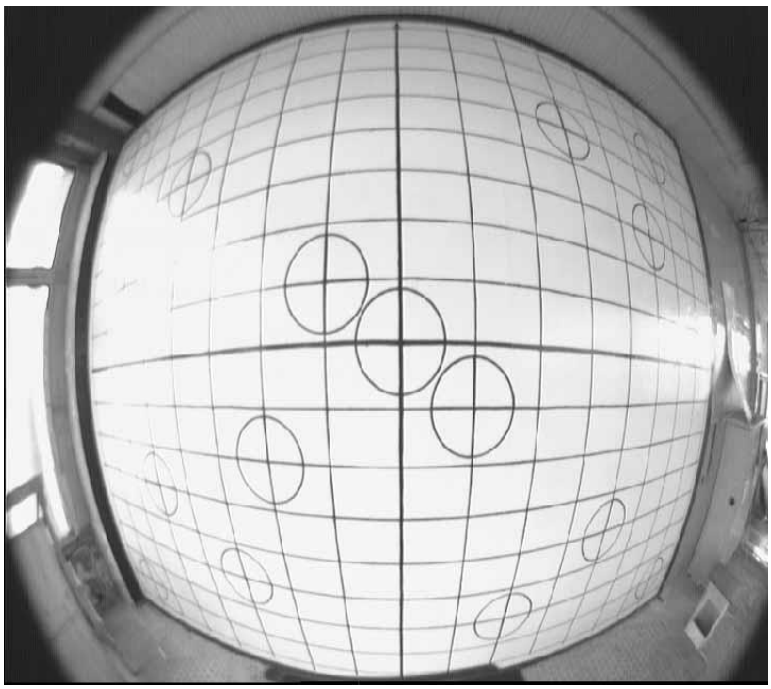


Рис. 3.40 Изображение, полученное сверхширокоугольным объективом

При видеонаблюдении протяженных объектов в силу целевого назначения прикладной ТВ системы используют широкоугольные объективы. Их применение обуславливает появление пространственных искажений, приводящих к существенному нарушению геометрического подобия в сформированных изображениях (геометрические искажения). Внесенные искажения составляют: нелинейность кубическая до 120%, искажения типа «бочка/подушка» до 30% (рис.3.40). В этом случае цифровая коррекция пространственных искажений является малоэффективной. Изображения, полученные в результате цифровой коррекции, имеют значительные остаточные искажения и существенную потерю разрешения. Потери разрешения на краях растра превышают 70%, в центре 40%.

Реализация функциональных особенностей интеллектуальной системы видеонаблюдения требует принципиально нового метода обработки видеоданных. Метод должен носить комплексный характер - одновременно обеспечивать решение целого ряда задач: коррекцию пространственных искажений; восстановление

построчного растра из чересстрочного; возможность просмотра видеоданных с произвольной скоростью; семантическое сжатие; извлечение дополнительной информации об объекте интереса.

Учитывая ключевую особенность объекта интереса – движение, предложено синтезировать его панорамное изображение. Панораму составляют из фрагментов, вырезанных в центральной части кадра. Ширина фрагмента определена скоростью движения объекта. Принцип синтеза панорамного изображения позволяет одновременно решить все перечисленные выше задачи.

Изображение в центральной части кадра имеет наилучшее качество: высокую четкость, минимальные искажения по горизонтали и вертикали. Использование фрагментов, вырезанных в центре кадра, обеспечивает формирование панорамного изображения, у которого искажения в направлении движения объекта соответствуют величине искажений вырезанного фрагмента, а искажения в перпендикулярном направлении могут быть скорректированы до низкого остаточного уровня.

Четкость изображения по площади синтезированного кадра выше, чем у исходного: в центральной части разрешающая способность короткофокусного объектива больше, чем на краях.

При синтезе панорамного изображения постоянно определяют скорость движения протяженного объекта. Эта информация позволяет корректно выполнить операцию совмещения четных и нечетных полей при использовании камер с чересстрочной разверткой. Сформированная панорама имеет разрешение по вертикали равное разрешению полного кадра без эффекта «ступенек».

Предложенный принцип формирования изображения обеспечивает высокий коэффициент сжатия без потери информации об объекте интереса. Коэффициент сжатия

$$k_{compr} = \frac{W}{w},$$

где W – ширина кадра (или высота H в случае движения объекта интереса по вертикали), w – ширина вырезаемого фрагмента по направлению движения.

При практической реализации небольшой размер фрагментов кадров необходимых для построения панорамы позволяет разместить их в буферной памяти ОЗУ и регулировать скорость анимации синтезированного изображения на экране. Это обеспечивает

возможность установки удобной для анализа скорости просмотра материала в режиме реального времени.

Найденные оценки скорости при построении панорамы позволяют извлечь дополнительную информацию об объекте: зафиксировать моменты начала и завершения движения, оценить форму объекта на основе пространственного среза скоростей и др.

Главной задачей при формировании панорамного изображения является определение видимой скорости движения объекта или камеры на основе анализа видеоданных. Для получения оценки видимой скорости использованы вектора движения. В этом случае их определение имеет следующие особенности:

- наличие ярко выраженной функциональной зависимости видимой скорости движения различных частей объекта от расстояния между ними и камерой - пространственный срез скоростей;
- зависимость точности оценки скорости от местоположения блока на изображении, обусловленная значимыми пространственными искажениями;
- существенное число аномальных векторов вследствие низкой детальности исходного изображения.

Даны следующие рекомендации по местоположению и размерам блоков при определении векторов движения.

- Блоки изображения, в которых ищут вектора движения, располагают в центре растра: при движении в горизонтальном направлении – вертикальной полосой, при движении по вертикали – горизонтальной полосой.
- Максимальный размер блока в направлении, совпадающем с направлением движения, определяют в соответствии с уровнем пространственных искажений в кадре и допустимой ошибкой при оценке максимальной скорости.
- Размер блока в направлении, перпендикулярном к направлению движения определен размером зон L пространственного среза скоростей.

Величина зон L обусловлена конкурирующими показателями: качеством построения панорамы $k_1 = f_1(L)$ и точностью оценки векторов движения $k_2 = f_2(L)$. Уменьшение размера зоны L повышает точность формируемого среза скоростей и улучшает качество панорамы. Размер блока $b = L * L$ определяет интервал усреднения при определении векторов движения, его уменьшение снижает точность

найденной оценки. Дополнительно, для получения равноточных оценок вектора движения размер всех зон должен быть одинаков.

Показатель k_2 может быть переведен в ограничения: размер зоны L должен обеспечивать потенциальную точность не ниже заданного уровня $\sigma < \sigma_p$.

$$\begin{cases} f_1(L) \to \min\limits_{L} \\ f_2(L) \le \sigma_p \\ L_1 = L_2 = ,\ldots, = L_K \end{cases},$$

где K – число зон в пространственном срезе скоростей.

Нахождение L является задачей поиска минимума функции с ограничениями в виде равенств.

Для снижения числа аномальных векторов предложено проводить временную фильтрацию на основе усреднения по мажоритарному принципу. Согласно экспериментальным данным, в результате временной фильтрации число аномальных векторов падает в 2 раза, обеспечиваемый уровень достоверности K_∂ составляет 0,8.

Для повышения уровня достоверности и дополнительного сглаживания поступающей информации введена регрессионная предсказывающая полиномиальная модель

$$\hat{U}_j = \sum_{k=0}^{K} a_k \varphi_k(j), \qquad (3.40)$$

где j – номер зоны пространственного среза скоростей j=0..N; $\hat{U}_j$ – оценка вектора движения U для j –го блока; α_k – параметры (коэффициенты) регрессионной модели; $\varphi_k(x)$ – базисные функции k = 0..K, ортонормированные на системе из N точек с весами

$$w_j = \frac{1}{\sqrt{1 - x_j^2}} \quad , \text{где } x_j = \frac{\left(j - \frac{N}{2}\right)}{\frac{N}{2}};$$

Базисные функции получены из последовательности 1, x, x^2, … методом ортогонализации Грамма-Шмидта, согласно рекуррентным выражениям, которые при симметричной относительно начала координат области изменений аргумента имеют вид:

$$\lambda_k \varphi_{k+1}(x) = x\varphi_k(x) - b_k \varphi_{k-1}(x) \ ; \ b_k = \sum_{i=1}^{N} w_i x_i \varphi_k(x_i)\varphi_{k-1}(x_i).$$

Для расчета полагают $\varphi_{-1}(x) \equiv 0,$ $\varphi_0(x) = \left(\sum_{i=1}^{N} w_i \right)^{-1/2}$, а коэффициенты λ_k определяют из условий нормировки.

Коэффициенты a_k определяют из условия минимизации взвешенной квадратической ошибки $\sigma^2 = \sum w_j (\hat{U}(x_j) - U(x_j))^2 = \min$ согласно выражению $\mathbf{a} = \mathbf{\Phi}^t \mathbf{W} \mathbf{U}$, где $\mathbf{\Phi}$ – матрица, составленная из отсчетов базисных функций $\varphi_k(x)$; $\mathbf{W}$ - диагональная матрица весовых коэффициентов.

На основе регрессионной модели построен предсказывающий фильтр. Для формирования панорамы используют оценки векторов движения $\hat{\mathrm{u}}$, рассчитанные по модели (3.40).

Алгоритм определения оценок векторов движения для каждого блока на основе предсказывающей регрессионной модели включает в себя три основных шага:

Шаг 1: Масштабирование установленных априорно параметров модели $\mathbf{a}$ по найденной средней скорости движения протяженного объекта интереса.

Шаг 2: Идентификация методом наименьших квадратов параметров модели a. Определение параметров выполняют по найденным векторам движения U. При этом вычисляют разницу между измеренным вектором движения и его оценкой. Чем меньшей погрешностью характеризуются измерения, тем больший вес

$$Weight_j = \exp\left[koff * (U_j - \hat{U}_j)^2 \right]$$

получают найденные вектора при идентификации параметров модели.

Шаг 3: Расчет по модели значений векторов скорости для каждого из блоков.

Масштабирование модели выполняют один раз после обнаружения начала движения объекта. Обновления параметров модели и расчет по модели оценок векторов движения проводят в течение всего времени формирования панорамы, через заданное число кадров, определяемое средней скоростью движения объекта.

На основе введенного пространственного среза скоростей, временной фильтрации и регрессионной предсказывающей модели разработана процедура определения скорости протяженного

движущегося объекта. Согласно экспериментальным данным средняя ошибка при оценке скорости 2,7 % , уровень достоверности $K_{\partial} = 0{,}999985$.

Полученные показатели обеспечивают все заявленные выше характеристики формируемого панорамного изображения и позволяют реализовать основные функциональные особенности интеллектуальной системы видеонаблюдения.

Функциональные особенности интеллектуальной ТВ системы сегментации и сопровождения определяют совокупность требований к применяемым в них методам обработки видеоданных: необходимо обеспечить минимальную ошибку сегментации, сегментацию изображений объектов, находящихся в непосредственной близости друг к другу и на сложном фоне; на этапе сопровождения разрешить ситуации окклюзии, слияния и разделения объектов интереса.

Для реализации перечисленных задач предложен метод сегментации и сопровождения неточечных объектов с жестким движением (неподвижные объекты рассматриваются как частный случай: движение с нулевой скоростью), основанный на совокупности признаков с использованием методов нечеткой логики.

Использование математического аппарата нечеткой логики обусловлено отсутствием априорной информации об объектах интереса: количество объектов, данные об их форме и размерах неизвестны.

Как наиболее информативные выделены признаки: детальность, движение и форма.

Признаки детальности, движения и пространственной связности используют для первоначальной сегментации объектов интереса, на этапе сопровождения дополнительно применяют признак формы.

На основании признака детальности, под которым понимают уровень высокочастотной энергии в фрагменте изображения заданного размера (3.38) выполняют предварительную классификацию.

Процедура предварительной классификации включает: предобработку изображения с целью подчеркивания высокочастотной составляющей; оценку уровня детальности блоков изображения; пороговую обработку.

Для предобработки изображения применяют многомасштабный морфологический градиент

$$MG(L) = \frac{1}{3}\sum_{i=1}^{3}\left[\left((L \oplus S_i) - (L \Theta S_i)\right) \Theta S_{i-1}\right],$$

где $\oplus$ и Θ – морфологические операции наращивания и эрозии; S_i - квадратная группа структурных элементов. Размер S_i равен $(2i + 1)(2i + 1)$ пикселей для $0 \le i \le 3$. В соответствии с выражением (3.40) значения градиентов рассчитывают трижды с использованием структурных элементов различной размерности, а затем результаты складывают.

Данные экспериментов показывают, что применение многомасштабного морфологического градиента обеспечивает максимальную эффективность по сравнению с такими методами предобработки, как пространственное дифференцирование, оператор Превитта и вейвлет фильтр Добеши. Критерии оценки эффективности: K_1 – степень выделения объектов интереса; K_2 – степень выделения фона, K_3 – результативность анализа и K_4 – инвариантность к свойствам изображения.

Распределение оценок детальности блоков изображения в кадре имеет зависимость Релея. Бинарный характер проводимой классификации и известный закон распределения оценок детальности позволяет реализовать пороговое ограничение.

В результате предварительной классификации выделяют множество блоков G_d с высоким уровнем детальности для дальнейшего анализа. Для каждого блока этого множества определяют трехкомпонентный вектор движения $(v_{xk}, v_{yk}, p_k)^{\mathrm{T}}$, где v_{xk}, v_{yk} – составляющие k-го вектора движения по направлениям x и y, p_k - оценка его достоверности

Вектор признаков $\mathbf{q}_k = (x_k, y_k, v_{kx}, v_{ky}, p_k)$, сопоставленный каждому блоку множества G_d, позволяет выделить объекты интереса (принять решение к какому объекту принадлежит k-ый блок с вектором признаков $\mathbf{q}_k$).

Предложено сегментировать объекты, объединяя блоки в группы на основе схожести признаков для блоков одной группы и отличий между группами. Этот подход в аппарате нечеткой логики называется кластеризацией. В терминах нечеткой логики блоки – это элементы, подлежащие кластеризации (объединению), объекты интереса – кластеры.

Задачу сегментации объектов интереса при отсутствии априорной информации следует рассматривать как нечеткую кластеризацию при неизвестном числе кластеров.

Исходной информацией для кластеризации является матрица наблюдений

$$\mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ q_{m1} & q_{m2} & \cdots & q_{mn} \end{bmatrix},$$

каждая строка которой представляет собой значения n признаков одного из M элементов кластеризации.

Нечеткие кластеры описывают матрицей возможностного нечеткого разбиения:

$$\mathbf{F} = [\mu_{ki}],\ \mu_{ki} \in [0,1],\ \ k = 1..M,\ \ i = 1..c$$

в которой k-ая строка содержит степени принадлежности элемента k к кластерам $O_1, O_2, ..., O_c$. При нечетком разбиении степень принадлежности объекта к кластеру μ_{ki} принимает значения из интервала [0, 1].

Матрица возможностного нечеткого разбиения должна обладать следующими свойствами

$$\exists i,\ \mu_{ki} > 0,\ \forall k \quad k = 1..M;$$

$$0 < \sum_{k=1}^{M} \mu_{ki} < M,\ i = 1..c.$$

Для определения центров потенциальных кластеров применен алгоритм горной кластеризации.

Потенциал центров кластеров определен выражением:

$$P(\mathbf{Z}_h) = \sum_{k=1}^{M} \exp(-\alpha \cdot D(\mathbf{Z}_h, \mathbf{q}_k)),$$

где $\mathbf{Z}_h = (z_{1h}, z_{2h}, ..., z_{nh})$ – вектор признаков потенциального центра h-го кластера, $h = 1..c$; α – положительная константа; $D(\mathbf{Z}_h, \mathbf{q}_k)$ – расстояние между вектором признаков потенциального центра кластера $\mathbf{Z}_h$ и вектором признаков элемента кластеризации $\mathbf{q}_k$.

Расстояние между вектором признаков потенциального центра и элемента кластеризации:

$$D(\mathbf{Z}_h, \mathbf{q}_k) = \left\| \mathbf{d}_k^{rv} \right\|; \quad \mathbf{d}_k^{rv} = (d_r, d_v);$$

$$d_r = \sqrt{(x_k - x_c)^2 + (y_k - y_c)^2}\,; \quad d_v = p_k p_c \sqrt{(v_{xk} - v_{xc})^2 + (v_{yk} - v_{yc})^2}\,,$$

где x_c, y_c – координаты левого верхнего угла, v_{cx}, v_{cy} и p_c – найденный вектор движения и его достоверность для блока, являющегося потенциальным центром кластера; x_k, y_k – координаты левого верхнего угла, v_{kx}, v_{ky} и p_k – найденный вектор движения и его достоверность для блока, являющегося элементом кластеризации.

В качестве центров кластеров выбирают точки с максимальным потенциалом («горные вершины»). Центром первого кластера назначают точку с наибольшим потенциалом. При выборе следующего центра кластера исключают влияние найденного кластера. Для этого значения потенциала оставшихся возможных центров кластеров пересчитывают:

$$P_2(\mathbf{Z}_h) = P_1(\mathbf{Z}_h) - P_1(\mathbf{V}_1) \cdot \exp(-\beta \cdot D(\mathbf{Z}_h, \mathbf{V}_1)),$$

где $P_1(\,)$ – потенциал на 1-й итерации; $P_2(\,)$ – потенциал на 2-й итерации; $\mathbf{V}_1$ – вектор признаков первого найденного кластера: $\underset{Z_1, Z_2, \ldots, Z_x}{\arg\max}\,(P_1(\mathbf{Z}_1), P_1(\mathbf{Z}_2), \ldots, P_1(\mathbf{Z}_c))$; β – положительная константа.

Центр второго кластера определяется по максимальному значению обновленного потенциала: $\underset{Z_1, Z_2, \ldots, Z_x}{\arg\max}\,(P_2(\mathbf{Z}_1), P_2(\mathbf{Z}_2), \ldots, P_2(\mathbf{Z}_c))$

После определения центров кластеров формируют матрицу нечеткого разбиения. Значение элементов матрицы μ_{ki} определены степенью принадлежности элемента (блока) кластеру (объекту), которое задано расстоянием между вектором признаков блока и вектором признаков центра кластера.

При сегментации объектов учитывают признак пространственной связности – блоки должны образовывать связную группу и иметь схожие вектора движения. Первым признаком r_{kc1} принадлежности блока k к кластеру Ω_c является минимальное расстояние $d_{\min} = \min(d_{kl}),$ где $l = 1..m,\ k = 1..m; l \neq k$ до одного из ранее включенных в кластер блоков (на первом шаге – к центру кластера).

Вторым признаком r_{kc2} является взвешенная норма разностного вектора $p_k \|\mathbf{v}_k - \mathbf{v}_c\|$, где $\mathbf{v}_c$ – наиболее вероятный вектор движения для данного кластера, а p_k – оценка достоверности вектора движения $\mathbf{v}_k$.

Для k-го блока формируют обобщенный признак в виде нормы вектора $\mathbf{r}_{kc}$, составленного из взвешенных значений частных признаков

$$d_{kc} = \|\mathbf{r}_{kc}\| = \mathbf{r}_{kc}{}^T \mathbf{W}\mathbf{r}_{kc},$$

где матрица $\mathbf{W}$ переменных весовых коэффициентов учитывает динамику свойств объектов интереса и окружающей обстановки.

Для определения степени принадлежности блока кластеру введена экспоненциальная функция на основании обобщенного признака d :

$$\mu = \exp(-d^2 / \beta^2),$$

где β – масштабный коэффициент.

Сопровождение объектов интереса реализовано на основе обобщенной модели формы Гаусса и с использованием теории нечетких множеств. Анализируют соответствие вектора признаков фрагмента изображения (блока) векторам признаков объектов, сегментированных на предыдущем шаге.

Представление в виде эллипсов рассевания Гаусса является наиболее общим и несет информацию о центре тяжести объекта μ_x, μ_y (центр эллипса), его линейных размерах L_1, L_2 (длины полуосей эллипса), коэффициенте элонгации $elong = \frac{L_1}{L_2}$, угле наклона φ, мере несимметрии объекта E.

Построение модели формы и движения для каждого сегментированного объекта позволяет использовать для решения задачи сопровождения математический аппарат нечетких множеств. На множестве всех блоков кадра сформированы нечеткие множества объектов O_g $g = 1..G$, где G – число объектов, сегментированных в предыдущем кадре t. Функция принадлежности нечеткого множества O_g получена на основе t-нормы вероятностного пересечения нечетких множеств G_{fog} и G_{vog}.

$$\eta_{Og}(b_k \in O_g) = \eta_{vog}\eta_{fog},$$

где η_{fog} – функция принадлежности нечеткого множества G_{fog} блоков, соответствующих модели формы объекта g; η_{vog} – функция

принадлежности нечеткого множество G_{vog} блоков, соответствующих модели движения объекта g

$$\eta_{fog}(b)=\frac{1}{2\pi\sigma_{xg}\sigma_{yg}}\exp\left\{-\frac{1}{2(1-\rho)}\left[\frac{(x_k-\hat{\mu}_{xg})^2}{\sigma_{xg}^2}-2\rho\frac{(x_k-\hat{\mu}_{xg})(y_k-\hat{\mu}_{yg})}{\sigma_{xg}\sigma_{yg}}+\frac{(y_k-\hat{\mu}_{yg})^2}{\sigma_{yg}^2}\right]\right\},$$

$$\sigma_{xg}=\sqrt{D_{xg}}\ ;\ \sigma_{yg}=\sqrt{D_{yg}}\ ;\ \rho=\sqrt{K_{xyg}}$$

где b – блок с координатами (x_k, y_k); D_{xg}, D_{yg}, K_{xyg} – значения дисперсии и ковариации, полученные в кадре t при определении параметров модели формы объекта в виде эллипсов рассеивания Гаусса; $(\hat{\mu}_{xg}, \hat{\mu}_{yg})$ – оценка координат центра тяжести объекта в кадре $t+1$ на основе координат центра тяжести в кадре t.

$$\eta_{vog}(b)=\exp\left(-\left[w_g\left\|\mathbf{v}_k-\mathbf{v}_{og}\right\|\right]\right),$$

где $\mathbf{v_{og}}$ – вектор движения объекта, найденный согласно модели движения объекта O_g, w_g – весовой коэффициент, учитывающий выраженность признака движения и особенности окружающей обстановки.

При сопровождении нескольких близко расположенных объектов функция принадлежности имеет вид

$$\eta(O_g)=\eta^2(O_g)/\sum_{g=1}^{G}\eta(O_g).$$

Сопровождение объектов выполняют выявлением принадлежности блоков кадра t + 1, сформированным нечетким множествам O_g кадра t. Для однозначного определения объектов введены α-сечения каждого из множеств O_g.

В случае длительного исчезновения объекта необходимо отождествить потерянный объект с одним из вновь захваченных объектов. Операцию отождествления выполняют по критерию минимума расстояния между векторами взвешенных параметров потерянного объекта и одного из вновь захваченных объектов

$$d_{gb}=\langle(\mathbf{f}_g-\mathbf{f}_b),\mathbf{WW}(\mathbf{f}_g-\mathbf{f}_b)\rangle,$$

где $\mathbf{f}_g$ – вектор параметров потерянного объекта $g = 1...G$, G – число объектов, сегментированных в кадре t; $\mathbf{f}_g = (\boldsymbol{\mu}_g, \mathbf{L}_g, \varphi_g, elong_g, \mathbf{v}_g, \mathbf{a}_g)$, $\boldsymbol{\mu}_g$ – центр тяжести, $\mathbf{L}_g$ – линейные размеры, φ_g – угол поворота, $elong_g$ – коэффициент элонгации, $\mathbf{v}_g$ и $\mathbf{a}_g$ – вектора скорости и ускорения объекта интереса; $\mathbf{f}_b$ – вектор параметров найденного объекта; $b = 1...K$, K – число вновь захваченных объектов в кадре $t+1$; $\mathbf{WW}$ – весовая матрица, в общем случае отличная от диагональной.

Как показали экспериментальные исследования, предложенный метод позволяет реализовать автоматический захват до 50 объектов интереса на сложном фоне и обеспечивает устойчивое сопровождение объектов интереса при существенной динамике их свойств и окружающей обстановки, а также при взаимодействии изображений объектов с фоном и друг с другом.

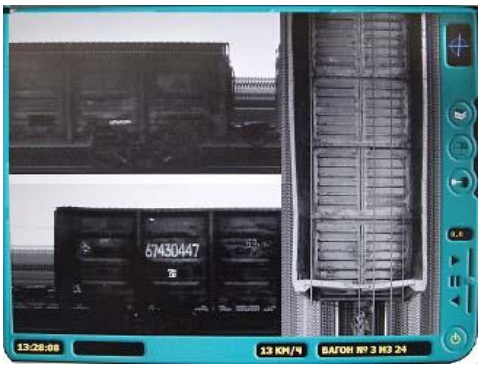


Рис. 3.41. Вид экрана монитора

Многоцелевой телевизионно-компьютерный комплекс видеомониторинга железнодорожных составов предназначен для обеспечения осмотра и выявления негабаритных грузов движущегося товарного состава с последующим сбором, обработкой, хранением и документированием информации о коммерческом состоянии вагонов и грузов, а также передачей ее в автоматизированную систему управления перевозками.

В состав системы входят сенсоры, электронная система ввода видеоизображений в компьютер, персональный компьютер, программное обеспечение. Сенсорами системы являются три видеокамеры, дающие изображения левого и правого по ходу поезда бортов вагонов, а также вид сверху; габаритные электронные ворота для выявления негабаритного груза.

Комплекс позволяет:

- На одном экране представить панорамные изображения всех трех изображений проходящих вагонов и дать дополнительную информацию о текущем времени, скорости движения состава и порядковом номере вагона (рис. 3.41).
- Предоставить оператору возможность самому устанавливать комфортную скорость просмотра состава с помощью движка, останавливать движение (стоп-кадр) и возвращаться назад.
- Записать синтезированные изображения в архив с высоким коэффициентом сжатия без потери информации об объекте интереса.

Экспериментальное исследование и результаты, полученные в процессе апробации комплекса, показывают:

1. Синтезированное панорамное изображение обеспечивает эффективную коррекцию пространственных искажений (рис. 5):

нелинейные искажения третьего порядка:

исходный уровень 120%; уровень после коррекции менее 5%;

искажения типа «бочка/подушка»:

исходный уровень 20%; уровень после коррекции менее 1%.

2. Полученный размер панорамы одного вагона позволяет разместить ее целиком на экране монитора.

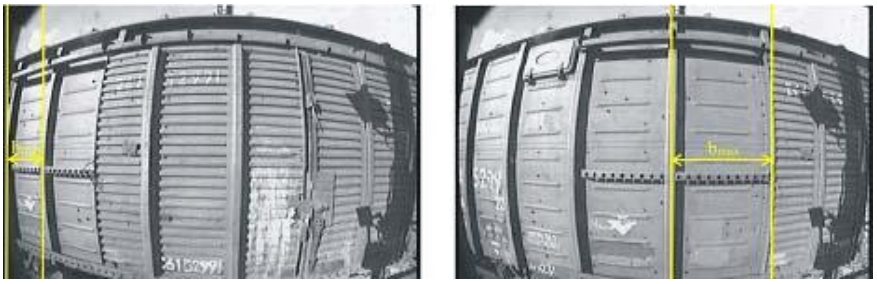
3. При движении протяженного объекта со скоростью от 20 до 60 км/ час и стандартной ширине кадра 768 пикселей по горизонтали достигаемый коэффициент сжатия от 80 до 25 раз, соответственно.

4. Средняя ошибка нарушения геометрического подобия (соответствия с определенным масштабным коэффициентом размеров объекта интереса на созданной панораме размерам реального объекта) составляет 2,7%. Достигнутая точность обеспечивает восприятие оператором синтезированного изображения, как изображения без искажений. Искажения менее 5% не воспринимаются зрительным анализатором человека.

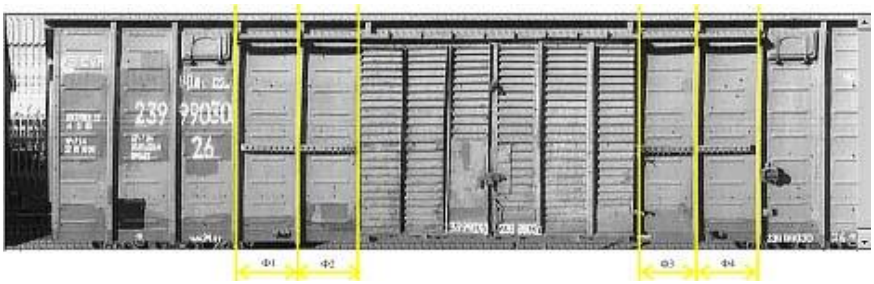
5. Информация о скорости движения протяженного объекта при построении панорамного изображения, одновременно с синтезом изображения объекта интереса, позволяет зафиксировать момент начала и окончания движения железнодорожного состава, определить направление и скорость движения, реализовать счет вагонов и выполнить идентификацию типа вагона.

Полученные результаты показывают, что синтезированное панорамное изображение позволяет реализовать основные функциональные особенности интеллектуальной системы видеонаблюдения протяженных объектов.

В многофункциональном оптико–электронном комплексе решены задачи автоматизированного захвата и автоматического



a)



b)

Рис. 3.42 а) Исходное изображение.

Коэффициент нелинейных искажений 109%

($b_{\min} = 56$; $b_{\max} = 191$).

сопровождения неточечных объектов интереса. Размеры объектов от 4*4 до 100*100 пикселей. Скорость движения от 0 до 10 пикселей за кадр. Движение объектов произвольное, возможно исчезновение длительностью до 10 сек. Число главных объектов интереса от одного до четырех. Число второстепенных объектов неограниченно. Эффективное отношение сигнал/шум (отношение превышение сигнала цели над фоном к СКО шума) не менее 6 дБ. Автоматизированный захват выполняет оператор.

Система решает следующие задачи:

1. Один из главных объектов (по выбору оператора) всегда удерживают в центре растра с ошибкой, не превышающей 1 пиксель по каждой координате.

2. Остальные объекты сопровождают стробом. Реализуют непрерывную (с частотой кадров) подстройку стробов под изменяющиеся размеры и ракурсы для всех четырех объектов интереса.

3. Выполняют разрешение ситуации окклюзии и временного (до 10 сек) исчезновения объектов интереса.

Экспериментальное исследование показывает:

1. Предложенный метод обеспечивает систематическую ошибку сегментации 16% от площади объекта интереса. Указанное значение в 3 раза ниже средней ошибки при сегментации объекта прямоугольным стробом. Случайная ошибка зависит от размеров объекта интереса: для объектов размером до 5 блоков она составляет 15-20%, для объектов большего размера 3-5%.

2. При сопровождении объектов статичной камерой ошибка при определении центра тяжести не превышает 1 пиксель.

3. Разрешение ситуации окклюзии во всех экспериментах было выполнено корректно и не привело к ложному изменению размеров стробов объектов интереса.

В состав системы входят видеокамера; электронная система ввода видеоизображений в компьютер; поворотный стол; персональный компьютер; программное обеспечение.

В автоматизированном режиме выполняют захват объекта интереса: оператор формирует строб вокруг объекта, подлежащего дальнейшему сопровождению. В автоматическом режиме система реализует удерживание в центре кадра объекта, выделенного оператором. Скорость перемещения объекта интереса в экранной плоскости до 5 пикселей за кадр. Возможно исчезновение объекта интереса длительностью до 3 секунд.

Экспериментальные исследования показывают:

1. Средняя ошибка слежения, определяемая предложенным методом (без влияния инерционности поворотного стола), составляет 1,5 пикселя.

2. Момент начала движения объекта интереса был зафиксирован на всех тестовых сюжетах.

Полученные показатели и качественные характеристики показывают, что предложенные в работе методы представления и обработки видеоданных, позволяют реализовать основные функциональные особенности интеллектуальной системы видеонаблюдения за протяженными объектами и систем сегментации и сопровождения неточечных объектов интереса с жестким движением.

3.11.Применение генетического алгоритма для фрактального сжатия изображения

В последнее время изображения и иллюстрации стали использоваться повсеместно. Проблема, связанная с большим объемом для их обработки и хранения, появилась при работе и на рабочих станциях, и на персональных компьютерах. Разработано большое количество различных алгоритмов архивации графики.

Майкл Барнсли и Алан Слоун нашли новый метод решения данной задачи. В методе используется принципиально новая идея - не близость цветов в локальной области, а подобие разных по размеру областей изображения. Так с помощью стандартных приемов обработки изображений, таких, как выделение краев и анализ текстурных вариаций, изображение делится на сегменты и кодируется с помощью некоторого сжимающего аффинного преобразования. Восстановление изображения происходит с помощью многократного применения этого аффинного преобразования.

Целью этого параграфа является построение метода фрактального сжатия для статических изображений с использованием генетических алгоритмов (ГА). В работе рассматриваются основные принципы метода, его обоснование, несколько алгоритмов его реализации и модификация генетического алгоритма в применении к задаче поиска кодирующего преобразования.

В начале работы излагаются общие принципы и понятия, необходимые для описания некоторых методов сжатия изображений, приводятся несколько методов кодирования цифровой информации и схемы их реализации. Далее, описывается более подробно метод фрактального сжатия и его математическое обоснование. В заключении изложены результаты вычислительного эксперимента для задачи фрактального сжатия с помощью ГА.

3.11.1. Представление изображений конечным объемом данных

Компьютерное изображение в его цифровом представлении является набором значений интенсивностей светового потока, распределенных по конечной площади, имеющей обычно прямоугольную форму.

Для простоты рассмотрим сначала монохромные изображения. Тогда интенсивность излучаемой световой энергии с единицы поверхности в точке с координатами (ξ,η) изображения можно представить некоторым числом $B(\xi,\eta)$. Единичный элемент изображения, характеризуемый определенным значением (ξ,η), называется пикселем, а величина $z = f(\xi,\eta)$ - яркостью.

Статистическая и визуальная избыточность изображений

Поток данных об изображении имеет существенное количество излишней информации, которая может быть устранена практически без заметных для глаза искажений.

Существует два типа избыточности.

- Статистическая избыточность, связанная с корреляцией и предсказуемостью данных. Эта избыточность может быть устранена без потери информации, исходные данные при этом могут быть полностью восстановлены.
- Визуальная (субъективная) избыточность, которую можно устранить с частичной потерей данных, мало влияющих на качество воспроизводимых изображений; это - информация, которую можно изъять из изображения, не нарушая визуально воспринимаемое качество изображений.

3.11.2. Статистическая избыточность изображений

Пусть имеется дискретизированное $M \times N$ пикселей и квантованное с точностью K бит на пиксель монохромное изображение. Следовательно, для хранения этого изображения необходимо $M \times N \times K$ бит информации.

Если предположить, что квантованные значения яркости не равновероятны, то уменьшение информации возможно путем изменения количества бит информации для кодирования пикселей: более вероятные кодируются словами с меньшим количеством бит, менее вероятные - с большим. Этот метод называется кодированием словами переменной длины или энтропийным кодированием.

Пусть квантованный уровень яркости z имеет вероятность $P(z)$ и ему присваивается слово - код длины $L(z)$ бит. Тогда средняя длина кода для всего изображения составит $\hat{L} = \sum_b L(z) \cdot P(z)$ бит на пиксель. Нижняя граница для $\hat{L}$ определяется информационной теоремой и называется энтропией случайной величины:

$$H(f) = -\sum_z P(z) \cdot \log_2 P(z) \le \hat{L}.$$

Таким образом, энтропия - это мера количества информации, которую несет случайная величина уровня яркости z.

$H(z) \ge 0$, поскольку $P(z) \in [0,1]$. Из формулы $H(f)$ вытекает, что чем более неравномерно распределение $P(z)$, тем меньше энтропия и тем эффективнее может быть энтропийное кодирование.

Наиболее известные методы эффективного кодирования символов основаны на знании частоты каждого символа присутствующего в сообщении. Зная эти частоты, строят таблицу кодов, обладающую следующими свойствами:

- различные коды могут иметь различное количество бит
- коды символов с большей частотой встречаемости, имеют больше бит, чем коды символов с меньшей частотой
- хотя коды имеют различную битовую длину, они могут быть восстановлены единственным образом.

Этими свойствами обладает известный алгоритм Хаффмана [6].

3.11.3. Визуальная избыточность изображений

Устранение визуальной избыточности изображений является основным резервом сокращения передаваемой информации. Для оптимизации процесса кодирования с точки зрения обеспечения передачи наименьшего объема информации необходимо, с одной стороны, не передавать избыточную информацию, а, с другой, - не допустить чрезмерной потери качества изображения.

До сих пор не существует простой и адекватной модели визуального восприятия изображений, пригодной для оптимизации их кодирования [5].

3.11.4. О кодировании цветных изображений

Большинство цветных растровых изображений для сохранения деталей цвета пикселя применяют систему цветов *RGB*, прежде всего потому, что эта система используется для изображения цветов монитором компьютера. Для точного сжатия изображения с цветами

RGB необходимо все три компоненты сжимать с одинаковым уровнем точности. Комбинация этих трех компонент определяет относительную яркость пикселя, а также его цвет. Поэтому изменение любого из трех значений скажется и на яркости, и на цвете пикселя.

Так как глаз человека более чувствителен к изменению яркости, чем к изменению цвета, часто возникает желание преобразовать систему цветов *RGB* в такую систему, где информация о яркости пикселях запоминалась бы отдельно от информации о цвете. Общеупотребительной системой цветов, которая хранит яркость пикселя в виде отдельной компоненты данных о его цвете, является система *HSB* (тон, насыщенность, яркость) и *YUV* (*Y* - компонента яркости; *U* и *V* хранят характеристики цвета).

3.11.5. Обзор некоторых алгоритмов сжатия с потерями

Метод усеченного блочного кодирования (УБК)

Название метода отражает тот факт, что изображение разбивается на небольшие прямоугольные куски одинакового размера, называемые блоками. Этот метод в отличие от большинства других подстраивает параметры кодирования не под некоторую усредненную характеристику всего изображения, а под локальные особенности в пределах каждого блока. Это позволяет сохранить мелкие детали изображений. Метод не приводит к размыванию границ, что характерно для некоторых других алгоритмов. Метод УБК сопоставим с большинством других методов по эффективности сжатия данных и по объему вычислений, требуемых для кодирования, но не имеет конкурентов по простоте декодирования.

Базовый алгоритм УБК строится следующим образом. Изображение, представленное $M \times N$ - матрицей $\|b_{ij}\|$ яркостей пикселей, разбивается на небольшие прямоугольные блоки $m \times n$ элементов. Каждый такой блок обрабатывается независимо от других, поэтому опишем алгоритм обработки одного блока.

Обработка блока начинается с вычисления порога и двух уровней квантования (описанных ниже), затем проводится квантование блока на два уровня, после чего следует упаковка проквантованного блока. Для определения уровней квантования сначала вычисляются два первых выборочных момента - среднее значение C и средний квадрат E:

$$C = \frac{1}{m \times n} \sum_i \sum_i b_{ij} \ , \ E = \frac{1}{m \times n} \sum_i \sum_j b_{ij}^2$$

(где суммируются элементы изображения в пределах блока) и дисперсия

$$\sigma^2 = E - c^2 \cdot$$

Пороговая величина квантователя d полагается равной среднему C. Верхний a и нижний b уровни квантования вычисляются по следующим формулам:

$$a = C - \sigma\sqrt{q/(p-q)} \ , \ b = C + \sigma\sqrt{(p-q)/q} \ ,$$

где $p = m \times n$ - число элементов блока, q - число элементов блока, превышающих порог d.

Квантование проводится по правилу:

$$s_{ij} = \begin{cases} a, если \ b_{ij} < d \\ b, если \ b_{ij} \geq d \end{cases},$$

где s_{ij} - элементы изображения после квантования.

После квантования получается блок, содержащий только уровни a и b. Нетрудно показать, что среднее значение и средний квадрат исходного и проквантованного блоков совпадают. Практически для удобства последующей упаковки вместо a записывается нуль, вместо b - единица. Уровни a и b записываются отдельно.

Упаковка состоит в том, что блок, содержащий только нули и единицы, интерпретируется как двоичное число, имеющее $m \times n$ разрядов. Восстановление закодированного изображения также проводится поблочно и состоит в распаковке и обратной подстановке.

Из алгоритма видно, что степень сжатия непосредственно зависит от размеров блока. Наиболее удовлетворительные результаты, как по степени сжатия, так и по качеству восстановленного изображения были получены при использовании блоков размером 4×4 (см. [3]).

Описанный выше способ определения порога и уровней квантования не является единственным. Существует ряд других критериев. Важно, чтобы критерий соответствовал целям последующей обработки изображения и ее конкретным особенностям.

3.11.6. JPEG

JPEG - один из самых распространенных и достаточно мощных алгоритмов, представляет собой метод сжатия изображений,

реализуемый различными способами. Работает он как на черно-белых, так и на полноцветных изображениях.

Коэффициент архивации в *JPEG* может изменяться в пределах от 2 до 200 раз. Как и у любого другого алгоритма сжатия с потерями, у *JPEG* свои особенности. Наиболее известны ‘эффект Гиббса’ и дробление изображения на квадраты. Первый проявляется около резких границ предмета, образуя вокруг своеобразный ‘ореол’. Разбиение на квадраты происходит, когда задается слишком большой коэффициент сжатия для данной картинки. Тем не менее, не смотря на эти недостатки, для архивации изображений, предназначенных для просмотра человеком, он на данный момент является лучшим.

Широкое применение *JPEG* сдерживается, пожалуй, лишь тем, что он оперирует 24-битными изображениями. Поэтому для того, чтобы с приемлемым качеством посмотреть картинку на обычном мониторе в 256-цветной палитре, требуется применение соответствующих алгоритмов и, следовательно, определенное время. В приложениях, ориентированных на придирчивого пользователя, таких, например, как игры, подобные задержки неприемлемы. Кроме того, если имеющиеся изображения, допустим, в 8-битном формате *GIF* перевести в 24-битный *JPEG*, а потом обратно в *GIF* для просмотра, то потеря качества произойдет дважды при обоих преобразованиях. Тем не менее, выигрыш в размерах архивов зачастую настолько велик (3-20 раз), а потери качества настолько малы, что хранение изображений в *JPEG* оказывается очень эффективным.

3.11.7.Преобразование цветов RGB в цвета YUV

В сжатии *JPEG* применяется система цветов *YUV*. Разделение данных *RGB* на данные *YUV* позволяет программе сжатия уделять больше внимание данным о яркости *(Y)*, чем данным о цвете *(UV)*. Этот процесс называется подвыборкой, так как три компоненты выбираются с различной частотой. Так метод подвыборки, который называется *YUV411*, на каждую выборку цвета делает четыре выборки данных о яркости. Например, если рисунок не подвергался подвыборке, то величины *YUV* встречаются в нем с одинаковой частотой. При использовании подвыборки *YUV411* на шесть значений выборки обработанного файла приходится двенадцать значений выборки исходного файла. Таким образом, подвыборка данных сразу уменьшает размер файла изображения.

3.11.8.Метод сжатия JPEG

Процесс сжатия по схеме *JPEG* состоит из трех шагов (не считая подвыборку). Первый шаг - это запись изменения значений пикселей в виде изменения частот: как быстро меняются яркость и цвет пикселей. Второй шаг - группировка этих отдельных изменений частот по средним значениям (первый этап сжатия). И третий этап - сжатие этих усредненных данных с помощью модифицированного алгоритма кодирования Хаффмана.

3.11.9.Изменение частоты

JPEG определяет изменение частоты данных с помощью дискретного преобразования Фурье (*ДПФ*), которое применяется для каждой пиксельной компоненты в выбранной области пикселей. Например, выбирается группа из 8×8 пикселей, и к ней применяется преобразование *ДПФ*: сначала к величинам красных компонент во всей группе, затем зеленых, и, наконец, синих.

Вместо действительных значений пикселей величины *ДПФ* хранят скорость изменения интенсивности от пикселя к пикселю. На самом деле данных о частоте получается больше, чем исходных пиксельных данных, но следующие два шага это устраняют.

3.11.10. Усреднение

После вычисления с помощью *ДПФ* значений изменения частоты эти величины усредняются в соответствии с плавающей шкалой относительной важности. Это значит, что изменения частоты, которые меньше влияют на общий вид изображения (например, быстрые изменения частоты), усредняются больше других значений. Именно на этой стадии сжатия изображения происходят потери, так как величина применяемого усреднения может регулироваться.

3.11.11. Сжатие методом Хаффмана

Окончательно усредненные данные о частоте сжимаются с помощью модифицированного алгоритма кодирования Хаффмана, который особенно эффективен для этого типа данных, так как строит таблицы кодов, базирующиеся на частоте повторения величин.

3.11.12. Фрактальное сжатие

Эта группа алгоритмов является самой перспективной и развивается наиболее бурно. Основные идеи фрактального сжатия принадлежат Barnsley [7]. Первые практические результаты были получены Jacquin [8] в 1992 году. Алгоритм основывается на идее подобия между элементами изображения. Коэффициенты сжатия у

фрактальных алгоритмов варьируются в пределах 2-2000 раз. Причем большие коэффициенты достигаются на реальных изображениях, что нетипично для предшествующих алгоритмов. Кроме того, при разархивации изображение можно масштабировать. Уникальная особенность этого алгоритма заключается в том, что увеличенное изображение не дробится на квадраты.

Недостатком этого алгоритма является потребность в больших вычислительных мощностях при архивации. При этом распаковка требует меньше вычислений, чем у *JPEG*. Фактически это первый существенно несимметричный алгоритм. Причем, если у всех предшествующих алгоритмов коэффициент симметричности (отношение времени архивации ко времени разархивации) не превышает 3, то у фрактального алгоритма он колеблется от 1000 до 10000.

3.11.13. Метод фрактального сжатия изображений

Пусть задано изображение $M \times N$ пикселей. Обозначим через $\Omega = (0, M] \times (0, N] \in R^2$ множество точек изображения. При этом каждому пикселю (i, j) сопоставляется площадка:

$$\{(\xi, \eta) \in \Omega, \xi \in (j-1, j], \eta \in (i-1,]\},$$

точка $(i, j) \in \Omega$ - правый верхний угол площадки пикселя.

Основная идея фрактального сжатия состоит в представлении исходного изображения, заданного функцией $z = f(\xi, \eta)$, с помощью некоторого сжимающего отображения, неподвижная точка которого соответствует исходному изображению.

Пусть X - множество векторов из R^n, где $n = M \times N$ и i - я координата вектора соответствует одному пикселю изображения: $x_i = f(1 + (i-1) \bmod N, \left\lfloor \frac{i}{N} \right\rfloor + 1)$. Здесь $i = 1, \ldots, n$, $\lfloor \ \rfloor$ - символ округления вниз.

Нужное нам отображение будем искать в виде: $A: X \to X$.

Для задания такого отображения разобьем Ω на множество *R-блоков* $R_1, R_2, \ldots, R_r$, где $R_i \subseteq \Omega$, $i = 1, \ldots, r$ есть квадратный $B \times B$ пиксельный фрагмент изображения и $\bigcup_i R_i = \Omega$. Рассмотрим также множество *D-блоков*: $D_1, D_2, \ldots, D_d$, где $D_i \subseteq \Omega$ представляющие квадратные (возможно пересекающиеся) $2B \times 2B$ пиксельные фрагменты и $\bigcup_i D_i = \Omega$.

Процесс кодирования изображения заключается в поиске двух совокупностей аффинных преобразований W_i и F_i.

$W_i : D_{j(i)} \to R_i,\ i = 1,\ldots,r$ - биекция, действие которой описывается следующим образом: пусть $(\xi', \eta') = W_i(\xi, \eta)$ имеет вид

$$\begin{bmatrix} \xi' \\ \eta' \end{bmatrix} = \begin{bmatrix} \alpha_i & \beta_i \\ \gamma_i & \delta_i \end{bmatrix} \begin{bmatrix} \xi \\ \eta \end{bmatrix} + \begin{bmatrix} \varphi_i \\ \psi_i \end{bmatrix}.$$

Очевидно, существует всего восемь способов такого преобразования квадрата в квадрат.

Функционал $F_i : R_i \to R,\ i = 1,\ldots,r$ задает цветовые характеристики каждого пикселя (ξ, η) блока R_i и определяется как усредненные значения яркости пикселей прообраза (ξ, η):

$$F_i(\xi, \eta) = s_i \cdot \frac{1}{4} \sum \left(f(\tilde{\xi}, \tilde{\eta}) : (\tilde{\xi}, \tilde{\eta}) \in W^{-1}((\xi - 1, \xi] \times (\eta - 1, \eta]) \cap \mathrm{Z}^2 \right) + o_i, \quad (3.41)$$

где s_i отвечает за контрастность, а o_i за яркость.

Представим искомое отображение A, как результат действия совокупностей W_i и F_i следующим образом: пусть координаты вектора $x' = A(x)$ имеют вид $x'_k = F_{i(k)}(\xi(k), \eta(k))$, где $\xi(k) = 1 + (k-1) \bmod N$, $\eta(k) = \left\lfloor \frac{k}{N} \right\rfloor + 1$ - координаты правого верхнего угла площадки k - го пикселя, а $i(k)$ - номер R блока, содержащего точку $(\xi(k), \eta(k))$.

3.11.14. Обоснование метода фрактального сжатия

При фрактальном кодировании отображение A подбирается таким образом, чтобы минимизировать расстояние между вектором x и его образом $A(x)$ в некоторой метрике.

В соответствии с теоремой Банаха, если A - сжимающее отображение и (X, ρ) - полное метрическое пространство с метрикой ρ, тогда последовательность $\{x_k\}$, построенная по правилу

$$x_{k+1} = A(x_k), \quad (3.42)$$

сходится для произвольной точки $x_0 \in X$ к единственной неподвижной точке $x_A \in X$ преобразования A:

$$x_A = A(x_A). \quad (3.43).$$

Поэтому для восстановления изображения необходимо запомнить лишь отображение A.

Нетрудно видеть, что если семейство отображений F_i такое, что

$$L = \max_i |s_i| < 1, \text{ то } \rho(x, y) \le L \cdot \rho(A(x), A(y)), \quad (3.44)$$

где ρ метрика, порожденная нормой l^{∞}. Таким образом, для сходимости итерационного процесса (3.42) достаточно контролировать параметр s_i, чтобы $s_i < 1 \ \forall i = 1,...,r$. Тем не менее, необходимо отметить, что условие $L < 1$ не является необходимым для сходимости процесса восстановления.

Вообще говоря x_A не всегда будет точно совпадать с образом $A(x)$, но, при выполнении некоторых условий на оператор A можно гарантировать, что x_A будет «почти» также близко к x, как и $A(x)$.

Соответствующую оценку расстояния между x и $A(x)$ дает следующая теорема (Collage theorem) [9].

Теорема: $\rho(x, x_A) \le \frac{1}{1-L}\rho(x, A(x))$, где $L < 1$ - константа Липшица сжимающего оператора A в некоторой метрике ρ.

В случае использования евклидовой метрики константа Липшица оператора A может быть вычислена следующим образом [10]

$$L = \max_{1 \le j \le d} \sqrt{\frac{1}{4}\sum \left(s_i^2 \ : \ j(i) = j \right)} \ . \qquad (3.45)$$

В предлагаемом далее алгоритме в качестве критерия близости $A(x)$ к исходному вектору изображения x используется евклидова метрика. А именно, требуется найти такое отображение A, что $\rho(A(x), x)$ минимально. Это равносильно задаче: найти для каждого R-блока $R_i, i = 1,...,r$ такую тройку $(W_i, D_{j(i)}, F_i)$, что

$$\sigma^2_{R_i}(W_i, D_{j(i)}, F_i) = \sum \left(\left[f(\xi,\eta) - F_i(\xi,\eta) \right]^2 : (\xi,\eta) \in R_i \cap \mathrm{Z}^2 \right) \to \min , \quad (3.46)$$

где суммирование ведется по всем пикселям блока R_i.

При решении этой задачи для любых вариантов $D_{j(i)}$ и W_i можно подобрать оптимальные параметры s_i и o_i, используя метод наименьших квадратов для линейной регрессии.

Пусть даны две последовательности из B^2 значений цветов пикселей до и после применения W_i:

$$z_1 = f(\xi_1, \eta_1), z_2 f(\xi_2, \eta_2), ..., z_k = f(\xi_{B^2}, \eta_{B^2}) \text{ и}$$

$$z_1' = \frac{1}{4}\sum \left(f(\tilde{\xi}, \tilde{\eta}) : (\tilde{\xi}, \tilde{\eta}) \in W^{-1}\left((\xi_1 - 1, \xi_1] \times (\eta_1 - 1, \eta_1] \right) \cap \mathrm{Z}^2 \right),$$

$$\ldots,$$

$$z_k' = \frac{1}{4}\sum \left(f(\tilde{\xi}, \tilde{\eta}) : (\tilde{\xi}, \tilde{\eta}) \in W^{-1}\left((\xi_k - 1, \xi_k] \times (\eta_k - 1, \eta_k] \right) \cap \mathrm{Z}^2 \right),$$

где $(\xi_k, \eta_k) \in R_i$ и $z_k^{'}$ усредненные значения яркости пикселей прообраза (ξ_k, η_k) $\forall k = 1,..., B^2$.

Мы можем найти s_i и o_i для (3.41) минимизировав сумму

$$\sigma_{R_i}^2 = \frac{1}{B^2} \sum_{k=1}^{B^2} (s_i z_k^{'} + o_i - z_k)^2 . \qquad (3.47)$$

Функция (3.47) имеет минимум в тех точках, в которых частные производные от $\sigma_{R_i}^2$ по параметрам s_i и o_i обращаются в нуль. В результате дифференцирования и элементарных преобразований для определения параметров получаем систему двух линейных уравнений с двумя неизвестными s_i и o_i откуда получаем, что

$$s_i = \frac{\left[B^2 \cdot \left(\sum_{k=1}^{B^2} z_k^{'} z_k \right) - \left(\sum_{k=1}^{B^2} z_k^{'} \right) \cdot \left(\sum_{k=1}^{B^2} z_k \right) \right]}{B^2 \cdot \sum_{k=1}^{B^2} z_k^{' 2} - \left(\sum_{k=1}^{B^2} z_k^{'} \right)^2}, \qquad o_i = \frac{1}{B^2} \left(\sum_{k=1}^{B^2} z_k - s_i \cdot \sum_{k=1}^{B_2} z_k^{'} \right).$$

В частности, если $s_i = 0$, то $o_i = \frac{1}{B^2} \sum_{k=1}^{B^2} z_k$. Таким образом среднеквадратичная ошибка преобразованного блока от его исходного состояния

$$\sigma_{R_i}^2 = \frac{1}{B^2} \left[\sum_{k=1}^{B^2} z_k^2 + s \cdot \left(s \cdot \sum_{k=1}^{B^2} z_k^{' 2} - 2 \cdot \sum_{k=1}^{B^2} z_k^{'} z_k + 2 \cdot o \sum_{k=1}^{B^2} z_k^{'} \right) + o \cdot \left(o \cdot B^2 - 2 \cdot \sum_{k=1}^{B^2} z_k \right) \right].$$

Задача поиска D-блока $D_{j(i)}$ и его аффинного отображения W_i может быть решена различными способами. Например с помощью ГА, алгоритм Фишера [10], полный перебор и т.д.

3.11.15. Алгоритмы поиска фрактальной модели изображения Алгоритм Фишера

Идея алгоритма заключается в том, чтобы некоторым образом классифицировать D-блоки и R-блоки, а поиск близкого D-блока производить в том же классе, к которому относится ранговая область. Делается это следующим образом.

Исходные блоки разбиваются на четыре части. Для каждой из частей подсчитывается среднее значение A_i и дисперсия V_i пикселей. Далее блоки классифицируются по следующему принципу. Определим три базовых типа блоков

тип 1: $A_1 \geq A_2 \geq A_3 \leq A_4$,

тип 2: $A_1 \geq A_2 \geq A_4 \geq A_3$,

тип 3: $A_1 \geq A_4 \geq A_2 \geq A_3$.

Понятно, что любой блок при помощи соответствующего аффинного преобразования квадрата в квадрат можно привести к виду, соответствующему одному из указанных типов. После того, как мы зафиксировали три основных класса, блоки классифицируются по дисперсии. Таким образом, в каждом из трех классов появляются 24 подкласса, итого 72 класса. Поиск близкого к R-блоку D-блока производится перебором в соответствующем классе.

При использовании ГА для поиска оптимальных решений каждый элемент $x \in X$ пространства оптимизации должен быть представлен как вектор $b \in B$ из N символов двоичного алфавита A = {0,1}, где $B = A^N$. Необходимо также, чтобы пространство оптимизации X состояло из конечного числа элементов.

Популяцией $\Pi = (\chi^1, \chi^2, \ldots, \chi M)$ численности M считается вектор пространства B^M, координаты которого называются генотипами особей данной популяции.

Шагом ГА является переход от текущего поколения к следующему, т.е. получение новой популяции Π_{t+1} из Π_t. В построении очередной особи новой популяции участвуют операторы кроссинговера, мутации и случайный оператор отбора, *Select*: $B^M \to \{1,\ldots,M\}$ действие которого состоит в выборе номера особи родителя при порождении очередного потомка.

Для определения ГА в необходимо задать оператор кроссинговера (скрещивания) *Cross*: $B \times B \to B \times B$ и оператор мутации *Mut*: $B \to B$.

Действие кроссинговера $(\chi', \tau') = Cross(\chi, \tau)$ заключается в выборе случайным образом некоторой позиции *j*, равномерно распределенной от *1* до N-*1*, после чего результат формируется в виде

$$\chi' = (\chi_1, \chi_2, \ldots, \chi_j, \tau_{j+1}, \ldots, \tau_N), \ \tau' = (\tau_1, \tau_2, \ldots, \tau_j, \chi_{j+1}, \ldots, \chi_N).$$

Влияние кроссинговера регулируют с помощью вероятности P_{Cross} срабатывания этого оператора (в противном случае все остается без изменений).

Оператор мутации в каждой позиции аргумента с заданной вероятностью P_{mut} заменяет ее содержимое на случайный элемент двоичного алфавита A, выбранный в соответствии с равномерным распределением (в противном случае все остается без изменений).

Целевая функция исходной задачи, заменяется в ГА на неотрицательную функцию пригодности генотипа $\Phi(\chi)$, где $\chi \in B$.

Процесс работы алгоритма представляет собой последовательную смену поколений, на каждом шаге которой популяция Π_{t+1} наполняется парами потомков от особей популяции Π_t по формуле

$$(\chi_k^{t+1}, \chi_{k+1}^{t+1}) = Mut(Cross(\chi_{Select(\Pi_t)}^t, \chi_{Select(\Pi_t)}^t)),$$

где $(\chi_k^{t+1}, \chi_{k+1}^{t+1})$ - особи с наименьшей пригодностью популяции Π_t. То есть индивиды извлекаются попарно из Π_t и после кроссинговера и мутации помещаются в Π_{t+1}. Изменение вероятностей мутации и кроссинговера позволяет регулировать работу ГА и настраивать его на конкретные задачи.

Модификация генетического алгоритма для задачи фрактального сжатия

Опишем схему ГА в применении к задаче фрактального сжатия. В качестве генотипа ГА удобно взять вектор, компонентами которого будут пиксельные координаты области $D_{j(i)}$ исходного изображения, определенного на тороидальной поверхности, и число кодирующее аффинное преобразование W_i. Имеется восемь способов аффинного преобразования квадрата в квадрат: поворот на четыре стороны или зеркальное отражение и поворот на четыре стороны. Следовательно, на кодировку этого преобразования достаточно трех бит. Функцию пригодности положим равной

$$\Phi = \frac{1}{1 + \sum\left(\left[f(\xi,\eta) - F_i(\xi,\eta)\right]^2 : (\xi,\eta) \in R_i \cap Z^2\right)},$$

где в нижней части под знаком суммы – евклидово расстояние между исходным и преобразованным блоком. Данная функция удовлетворяет требования ГА (неотрицательна) и адекватна для оператора рулеточной селекции, при которой каждый индивид $\chi^{i,t}$ популяции Πt оказывается родителем при формировании очередной особи $\chi^{i,t+1}$ популяции $\Pi t+1$ с вероятностью

$$P_{select}(\chi^{i,t}) = \frac{\Phi(\chi^{i,t})}{\sum_j \Phi(\chi^{j,t})}.$$

При таком представлении хромосом, определяющих данный генотип, любой вектор пространства решений всегда допустим и имеет ненулевую пригодность.

Оператор мутации для данного алгоритма – стандартный, а оператор кроссинговера был модифицирован следующим образом:

- позиция кроссинговера может располагаться только в местах стыковки двоичного представления координат.
- выполняется только для близкородственных особей, т.е., если расстояние $\rho(\chi,\tau)$ мало, где ρ - некоторая метрика.

3.11.16. Увеличение быстродействия метода

Как уже было сказано выше, недостатком данного метода сжатия является очень большой объем вычислений для кодирования изображений. Для программ реализующих данный метод, применяются как обычные методы увеличения скорости программ: целочисленная арифметика, ассемблерная реализация и другие, так и некоторые специфические приемы: использование аффинных преобразований только одного типа, различные варианты разбиения изображения на области R_i.

Приведем схемы двух алгоритмов, которые для некоторых классов изображений могут значительно уменьшить объем вычислений. Параметрами первого алгоритма служат уровень потерь при кодировании и минимальный размер областей R_i. Этот алгоритм обеспечивает равномерное качество кодирования всего изображения. Параметром второго алгоритма является количество областей R_i, используемых для кодирования изображения, что прямо влияет на объем вычислений, но он не обеспечивает достаточной точности кодирования отдельных фрагментов изображения.

Алгоритм 1.

- Выберем допустимый уровень потерь при кодировании e.
- $R_1 = \Omega$ и пометим его как необработанный фрагмент.
- Пока есть необработанный фрагмент R_i выполнять:

1. Найти $D_{j(i)}$, W_i и F_i, которые наилучшим образом приближают R_i (на которых достигается минимум $\sigma^2_{R_i}(W_i, D_{j(i)}, F_i)$).

2. Если $\sigma^2_{R_i}(W_i, D_{j(i)}, F_i) < e$ или размер $R_i < \min$, то пометить R_i как обработанное. Иначе, разбить R_i на более мелкие фрагменты и пометить их как необработанные.

Алгоритм 2.

- Выберем максимальное число N фрагментов R_i.
- Добавим фрагмент $R_1 = \Omega$ в список преобразований и пометим его как необработанный.

- Пока есть необработанные фрагменты в списке выполнять:

1. Для каждого необработанного фрагмента найти соответствующие $D_{j(i)}$, W_i и F_i.

2. Найти в списке фрагмент R_i наибольшего размера и наибольшей метрикой $\sigma^2_{R_i}(W_i, D_{j(i)}, F_i)$.

3. Если число фрагментов в списке меньше N , тогда разбить фрагмент R_i на более мелкие и занести их в список как необработанные. Вычеркнуть из списка фрагмент R_i.

Вычислительный эксперимент

Описанный метод фрактального сжатия статических изображений был запрограммирован на языке C++ с применением объектно-ориентированного подхода. Программа работает как с черно-белыми, так и с полноцветными изображениями формата *Windows Bitmap*.

Для оценки качества закодированного 8-битного изображения или одного канала цветного использовалось соотношение сигнал-шум, измеряемое в децибелах (dB) и определенное как

$$PSNR = 10 \cdot \lg\left(\frac{255^2}{\sum (f(\xi,\eta) - \tilde{f}(\xi,\eta))^2}\right),$$

где суммирование ведется по всем пикселям исходного f и приближенного $\tilde{f}$ изображения.

Алгоритм кодирования реализован в двух вариантах.

1. *Алгоритм кодирования с фиксированным размером блоков.*

Алгоритм оперирует с прямоугольными областями одинакового размера. Размер областей фиксирован от начала работы алгоритма до конца.

Преимуществом данного алгоритма состоит в том, что при соответствующем выборе размеров обрабатываемых областей обеспечивается равномерное качество кодирования всего изображения. Недостатком алгоритма является малый коэффициент сжатия.

2. Алгоритм кодирования с разбиением изображения на дерево подблоков.

За основу был взят *алгоритм 1* из предыдущего параграфа. При недостаточной точности кодирования обрабатываемый фрагмент

разбивается на четыре части, каждая из которых обрабатывается так, как и все остальные.

Данное разбиение эффективно с точки зрения хранения его файле. Так любое такое разбиение можно представить в виде двоичной последовательности символов, где 1 - означает то, что блок разбит, 0 - иначе. При тестировании алгоритм показал лучшие результаты, чем его предшественник, по степени сжатия, но не всегда обеспечивает достаточной точности кодирования некоторых однотонных областей.

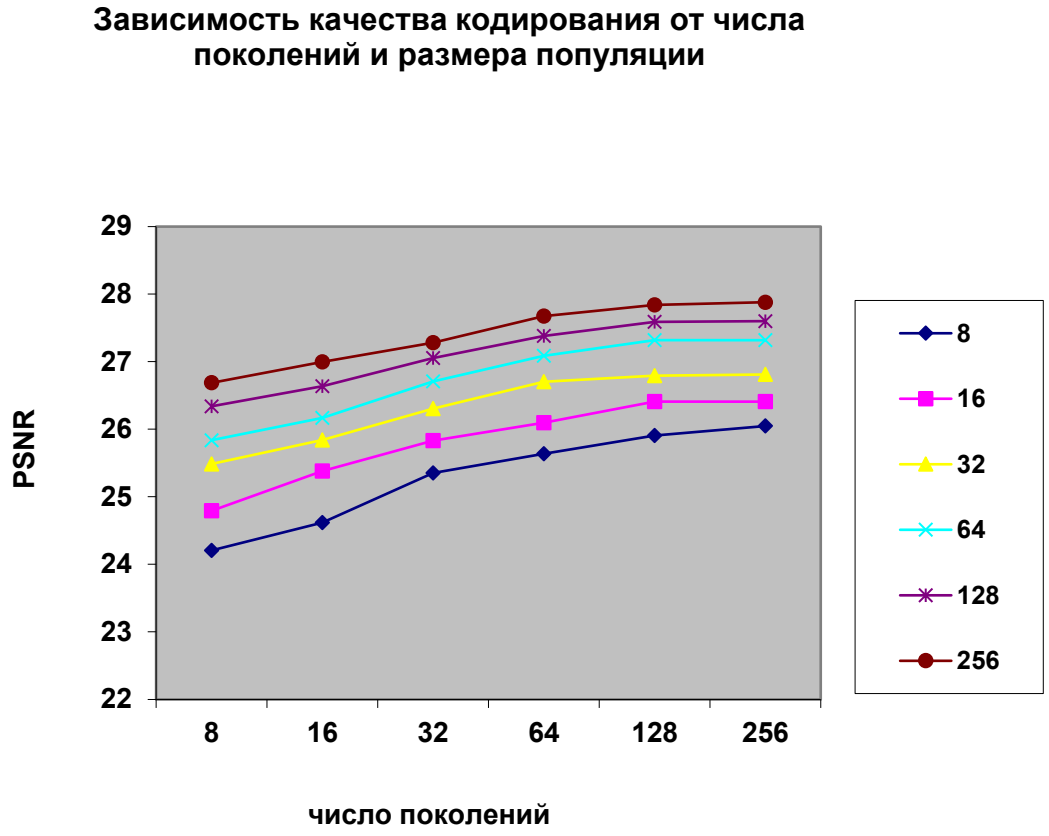
Степень сжатия изображения при фрактальном кодировании зависит от размеров обрабатываемых R-блоков, параметров s_i и o_i преобразования F_i. Для черно-белых изображений она может быть вычислена по формуле:

$$\frac{2^8 * \textit{размер блока в пикселях}}{\textit{число бит для } s + \textit{число бит для } o + \lceil \log_2(8 \times \textit{размер кодовой книги}) \rceil},$$

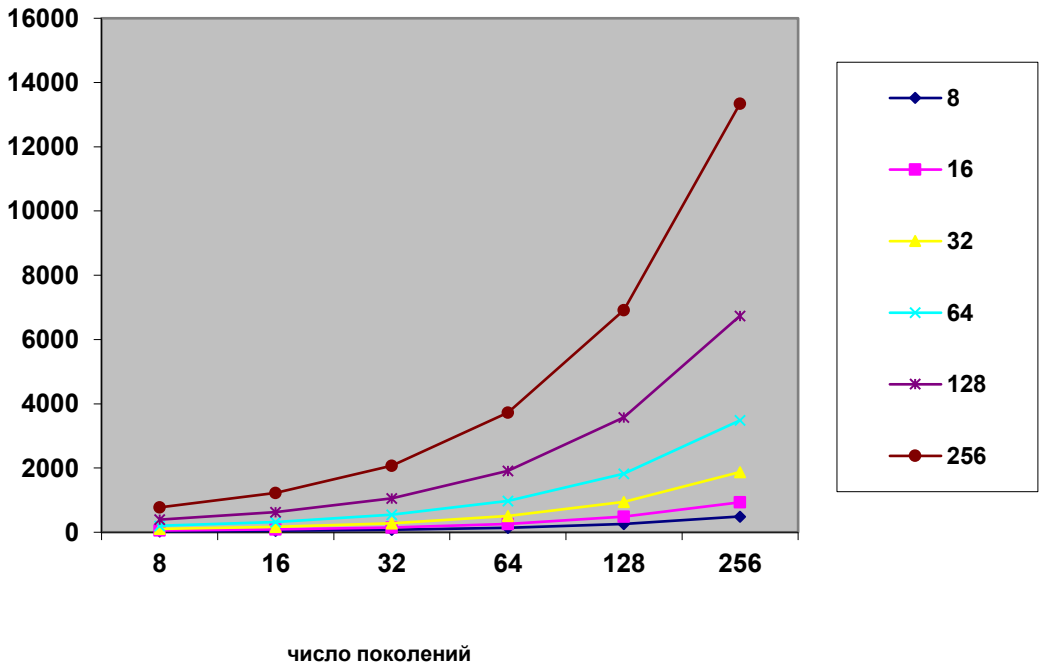
где $\lceil\ \rceil$ - символ округления вверх. Таким образом, мы можем получить дополнительную степень сжатия, если квантовать параметры s_i и o_i. Очевидно, что после квантования этих параметров среднеквадратическое отклонение между исходным и приближенным изображением возрастет на величину не большую, чем

$$\left(\frac{\max\limits_i(s_i) - \min\limits_i(s_i)}{2^{\textit{число бит для } s}} * 2^8 + \frac{\max\limits_i(o_i) - \min\limits_i(o_i)}{2^{\textit{число бит для } o}} \right)^2.$$

Поиск фрактальной модели *(для каждого R-блока соответствующий D-блок и преобразование W)* изображения осуществлялся с помощью модифицированного генетического алгоритма, описанного ранее. При проведении вычислительного эксперимента ГА прекращал свою работу по истечении заданного числа поколений или при достижении определенного качества кодирования, т.е если степень приспособленности наилучшей особи будет выше некоторого заранее заданного числа. Лучшие результаты были получены при запуске ГА со следующими параметрами $P_{Mut} = 0.1$, $P_{Cross} = 0.75$. Иллюстрации полученных результатов для различных вариантов алгоритма кодирования приведены в приложении. Представленные ниже статистические данные о работе ГА были получены при проведении эксперимента для изображения и алгоритма кодирования с фиксированным размером блоков 8×8 пикселей.



Следует отметить, что использование генетических алгоритмов для задачи фрактального сжатия является одним из вариантов ухода от полного перебора. Так для изображения размером 256×256 пикселей, для каждого R-блока нужно перебрать $2^8 * 2^8 * 2^3$ комбинаций D-блоков с аффинным преобразованием W. Следующая диаграмма показывает среднее число просчитанных вариантов при работе ГА в зависимости от числа поколений и размера популяции.



- Из этих графиков видно, что предложенный ГА имеет значительные преимущество по сравнению с алгоритмом полного перебора по времени работы.
- Представляется целесообразным использование данного алгоритма в сочетании с другими эвристиками такими, как алгоритм классификации Фишера, поиск с учетом фрактальной размерности и т.д.
- Приведенный вычислительный эксперимент показывает, что ГА может быть использован для сжатия графических изображений. В дальнейшем было бы интересно сравнить эффективность ГА с другими алгоритмами поиска кодирующего преобразования.

3.12. Улучшение качества изображений на основе применения эволюционирующей нейронной сети, вейвлет-преобразования и генетического алгоритма

Разработан двухэтапный метод улучшения качества полноцветных изображений на основе вейвлет-преобразования и генетического алгоритма. На первом этапе происходит удаление шумов. Для этой цели было выбрано вейвлет-преобразование, поскольку оно позволяет легко удалять высокочастотные компоненты. В течение второго этапа обработки происходит автоматическая настройка яркости и контраста на основе применения генетического алгоритма.

1. Применение нейроэволюционного алгоритма

В общем случае проблема попиксельной обработки изображений может быть представлена в виде проблемы поиска следующей функции преобразования $I^*=T(I,\boldsymbol{\Omega})$. Здесь I и I^* – интенсивность пикселей до и после обработки соответственно; $\boldsymbol{\Omega}$ – вектор параметров, определяющих локальные и глобальные характеристики каждого пикселя обрабатываемого изображения. В данной работе предлагается трехэтапный метод улучшения качества изображений посредством аппроксимации функции T на основании ее свойств [1]. Функция T аппроксимируется с помощью искусственной нейронной сети, которая обучается попиксельной обработке изображений с использованием разработанного эволюционного алгоритма (нейроэволюционный (НЭ) подход).

Разработанный НЭ алгоритм NEvA [2] позволяет одновременно настраивать веса и структуру ИНС. Каждая ИНС закодирована в генотипе в виде списка всех имеющихся связей. Веса связей представлены 19-битными числами, равномерно распределенными в интервале [-26,2144; +26,2143] с шагом 0,0001. Родительская подпопуляция формируется с использованием отбора усечением. Используются оригинальные операторы скрещивания и мутации, учитывающие структуру ИНС. Рассматриваются нейроны с лог-сигмоидной функцией активации. Размер популяции динамически настраивается в процессе эволюционного поиска.

При обработке изображений применяется локально-адаптивный подход. Особенность подхода заключается в независимой обработке каждого пикселя изображения, исходя из имеющегося набора его локальных и глобальных характеристик. Таким образом, ИНС

обучается обработке одного пикселя. При этом уменьшаются требования к объему оперативной памяти, необходимой для хранения информации об ИНС, и появляется возможность обрабатывать изображения произвольных размеров [73]. Проведенный анализ показал, что целесообразно рассматривать функцию преобразования интенсивности пикселей изображения в следующем виде $L^*(x,y) = T\left(L(x,y),\ D_{(x,y)},\ m_{(x,y)}\right)$. Здесь $L^*(x,y)$ и $L(x,y)$ – соответственно обработанное и исходное значение яркости пикселя $(x,\ y)$, $m_{(x,y)}$ и $D_{(x,y)}$ – соответственно средняя яркость и дисперсия яркости в локальной окрестности обрабатываемого пикселя. Таким образом, для рассматриваемого преобразования ИНС, аппроксимирующая функцию T, должна иметь 3 входа и 1 выход [73].

При обработке цветных изображений, сначала происходит преобразование изображений в полутоновые, затем осуществляется обработка с использованием ИНС, а после этого информация о цвете восстанавливается.

Во время обучения ИНС оцениваются в зависимости от визуального качества обработанных изображений. Используется приблизительная оценка качества изображений на основе модифицированной оценки из [74] следующего вида [73]: $f = \frac{N*M-\mu}{N*M} + \frac{256-\exp(H)}{192},\ H = -\sum_{i=1}^{256} l_i \log l_i$. Здесь N и M – соответственно ширина и высота изображения, μ – количество пикселей на границах перепадов уровней яркости, l_i – доля пикселей с i-м уровнем яркости. Значение функции f рассматривается в качестве ошибки ИНС, и, таким образом, задача эволюционной настройки ИНС заключается в минимизации функции f.

Первое слагаемое в выражении для функции f необходимо для максимизации числа пикселей на границах (краях) областей перепадов уровней яркости, для обеспечения детализированности обработанного изображения. Чем больше пикселей присутствует на краях изображения, тем более контрастным оно является. Второе слагаемое в указанном выражении препятствует «вырождению» обработанного изображения в бинарное, на котором присутствуют только черные и белые пиксели.

Оценивание функционирования ИНС осуществляется по последовательности $N*M$ ее выходных сигналов. Анализ получаемых нейросетевых решений и результаты экспериментов показали

необходимость использования пред- и постобработки изображений [73]. В результате проведенного исследования, для предварительной обработки была выбрана мультипликативная подстройка яркости исходного изображения [75].

ИНС обрабатывает пиксели с использованием локальных характеристик, поэтому для более эффективной обработки представляется разумным применение, в качестве алгоритма третьего этапа обработки, «глобального» алгоритма улучшения качества изображений. Для этой цели применяется алгоритм автоматической настройки уровней яркости, реализованный во многих графических пакетах. Таким образом, предлагаемый способ трехэтапной обработки включает в себя следующие этапы:

1. Предобработка изображений с помощью мультипликативной подстройки яркости.
2. Обработка на локальном уровне с использованием ИНС.
3. Глобальная обработка с применением алгоритма автонастройки уровней яркости.

В соответствии с предлагаемым подходом, применение обученных ИНС подразумевает использование локальных средней и дисперсии яркости, поэтому время обработки изображений существенно зависит от скорости вычисления этих характеристик. Ясно, что с увеличением размера окрестности вычислительная сложность также увеличивается, т.к. увеличивается количество обрабатываемых пикселей. В предположении скореллированности распределения яркости в соседних строках (т.е. рассматриваем случай достаточно «гладкого» распределения яркости на изображении) и достаточно малого размера окрестности, получены формулы для приближенных среднего $\tilde{m}_{(x,y)}$ и дисперсии $\tilde{D}_{(x,y)}$ в прямоугольной окрестности пикселя с координатами (x ,y) [75].

Для подсчета числа пикселей на границах областей различной яркости, необходимого для оценивания изображения, обработанного ИНС, применяется алгоритм детектора края Собеля, описанный в [6]. В качестве критерия прекращения работы программы используется значение $f_0 = 1{,}5$. Длительность эволюции составляет 25 поколений. Начальный размер популяции равен 50 особям и адаптивно изменяется во время запуска алгоритма NEvA. При обучении ИНС с целью повышения скорости обучения используется изображение с небольшими размерами (128x128 пикселей). По окончании процесса

обучения в популяции выбирается лучшая ИНС.

Время обучения ИНС составляет около 70 секунд на процессоре Интел Пентиум – 4 с частотой 3 ГГц. Полученная ИНС, как правило, содержит три входных нейрона, один или несколько скрытых нейронов и один выходной нейрон. Проведена трехэтапная обработка тестовых изображений и осуществлено сравнение с алгоритмом автоматической настройки уровней яркости и результатами обработки алгоритмом Multi-Scale Retinex (NASA) [77]. Время обработки цветного изображения размером 512x512 пикселей составляет около 1 секунды.

Ниже приведен результат улучшения качества исходного изображения (рис. 3.43) после применения трехэтапной обработки на основе алгоритма NEvA (рис. 3.44). Для сравнения на рис. 3.45 приведен результат улучшения исходного изображения на основе применения алгоритма Multi-Scale Retinex [77]. Следует отметить сопоставимое качество улучшенных изображений (рис. 3.44, 3.45), однако предложенный в данной работе трехэтапный метод обладает меньшей вычислительной сложностью по сравнению с алгоритмом Multi-Scale Retinex.

Рис. 3.43. Исходное изображение [77]

Рис. 3.44. Изображение после трехэтапной обработки на основе алгоритма NEvA

Рис. 3.45. Изображение после обработки алгоритмом Multi-Scale Retinex [77]

2. Применение вейвлет-преобразования и генетического алгоритма

Дискретное вейвлет-преобразование (Discrete Wavelet Transform, DWT) разработано для быстрого вычисления вейвлет-преобразования. Оно просто в выполнении и позволяет уменьшить время вычисления и количество требуемых ресурсов. В непрерывном вейвлет-преобразовании сигналы анализируются с использованием набора базисных функций, получающихся друг из друга путем сдвига

и масштабирования. В случае DWT-преобразования масштабное временное представление сигнала получается путем цифровой фильтрации. Анализируемый сигнал проходит через фильтры, на которых происходит «обрезание» определенных частот с различными масштабами [78].

Первоначально изображение, представленное в формате RGB переводится в цветовое пространство YUV по соответствующим формулам [79]. Компоненты *U* и *V* не подвергаются изменению во время фильтрации. Входным сигналом для системы является яркостная составляющая изображения (*Y*).

Вейвлет-преобразование в данном случае – это двумерное дискретное вейвлет-преобразование, основанное на материнском вейвлете «3-5», имеющем 3 высокочастотных коэффициента и 5 низкочастотных [79]. Фильтрация заключается в удалении из полученного набора вейвлет-коэффициентов высокочастотных компонент, которые и представляют собой шумы. Пользователь сам может определить порог при выполнении этой операции. После обратного вейвлет-преобразования изображение переводится в цветовое пространство RGB [79].

Для коррекции яркости и контраста применяется генетический алгоритм, позволяющий решить задачу оптимизации значений четырех параметров ядра улучшения. В разработанном алгоритме каждая особь имеет 4 хромосомы, которые представлены целыми числами (32 бита) [79]. Методы локального улучшения основаны на применении функции, зависящей от распределения яркости в окрестности каждой точки изображения. Одним из примеров таких методов является адаптивное выравнивание гистограмм, показавшее хорошие результаты в обработке медицинских изображений. Однако этот метод использует ядро, требующее больших вычислительных затрат. Ядро улучшения представляет собой преобразование, применяемое к каждой точке изображения с координатами *(x, y)*. Входным параметром для этого преобразования является исходная яркость точки полутонового изображения, выходное значение – итоговая яркость точки [79].

Для автоматизации процесса улучшения качества изображения необходимо определить объективный критерий для оценки качества улучшения изображения. Этот критерий используется для построения

целевой функции генетического алгоритма. Критерий качества в данном алгоритме включает 3 составляющих:

1. Количество и интенсивность краевых пикселей. Для выявления этих характеристик применяется детектор края Собеля [76].

2. Мера энтропии изображения. Необходимо оценить какое количество пикселей с различным уровнем яркости присутствуют на изображении.

3. Уровень адаптации к зрению человека по яркости. Человеческое зрение наиболее восприимчиво к изображениям, среднее значение яркости на которых соответствует середине всего диапазона возможных яркостей. Поэтому в целевую функцию был введен соответствующий множитель [79].

Разработана программа, реализующая данный метод улучшения изображений. Программа была применена для обработки набора тестовых изображений. Оказалось, что данный метод хорошо проявляет себя на двух классах изображений:

1. Слишком светлые или темные изображения. Данный метод позволяет добиться осветления слишком темных участков и затемнения слишком светлых. Кроме того, он позволяет проявить объекты, которые на исходном изображении были практически неразличимы.

2. Затуманенные изображения. Применение данного метода к затуманенным или нечетким изображениям позволяет повысить их четкость, а также снять затуманенность.

Реализованный генетический алгоритм при размере популяции 20 особей находит достаточно хорошее решение за 5 поколений. При этом обработка изображения в среднем занимает 10-20 секунд (например, для изображения размером 498x326 пикселей обработка заняла 16 секунд на компьютере с процессором Pentium 4 2,4 ГГц), что существенно превосходит скорость обработки изображений, достигнутую в работе [74].

Таким образом, в данной работе предложены трехэтапный и двухэтапный методы улучшения качества цифровых изображений. В основе трехэтапного метода лежит применение нейроэволюционного алгоритма, позволяющего определять оптимальные топологию и веса связей ИНС для решения задачи улучшения качества изображений.

Глава 4. КВАНТОВЫЕ МЕТОДЫ АНАЛИЗА И ОБРАБОТКИ ИЗОБРАЖЕНИЙ

4.1.Квантовое преобразование Фурье при обработке изображений

В данной работе представлен подход к применению квантового преобразования Фурье (QFT) для обработки изображений с использованием квантовых вычислений. Применение квантовых вычислений для анализа и обработки изображений становится все более актуальным в современной науке и технологиях. Представляется квантовая схему QFT, реализованную с использованием фреймворка Qiskit, который является инструментом для программирования квантовых компьютеров. Представлены основные шаги QFT и их применение к вектору состояния, представляющему интенсивности пикселей изображения. Исследованы влияние квантового преобразования на структуру изображения и представлены результаты в виде графиков и визуализаций. Кроме того, мы внедрены возможности вывода квантовой схемы QFT для более наглядного представления алгоритма. Полученные результаты подчеркивают потенциал квантовых вычислений в области обработки изображений и открывают новые перспективы для использования квантовых технологий в сфере компьютерного зрения.

С развитием квантовых технологий появляются новые возможности для решения вычислительных задач, в том числе и в области обработки изображений. Традиционные методы обработки изображений часто сталкиваются с ограничениями в скорости и эффективности, особенно при работе с большими объемами данных. В этом контексте применение квантовых вычислений для обработки изображений представляет собой перспективный направление исследований. Квантовые алгоритмы, такие как квантовое преобразование Фурье (QFT), могут обеспечить более эффективную обработку данных за счет использования принципов квантовой механики. В данной работе мы предлагаем исследование применения QFT к изображениям с использованием квантовых вычислений, что может привести к созданию более эффективных и быстрых методов обработки изображений. Этот подход актуален в свете постоянного

развития квантовых технологий и их потенциального влияния на область компьютерного зрения и анализа изображений [91].

С развитием квантовых вычислений и расширением области их применения возникают новые возможности в сфере обработки данных. Одним из увлекательных направлений в исследованиях является применение квантовых методов к обработке изображений. Традиционные методы, несмотря на свою эффективность, часто сталкиваются с ограничениями, особенно при работе с большими объемами информации. Квантовые вычисления представляют собой перспективный инструмент для создания новых, более эффективных методов обработки изображений. В частности, квантовое преобразование Фурье (QFT) привлекает внимание исследователей своей способностью эффективно обрабатывать данные в квантовой среде [92].

Целью данной работы является рассмотрение и анализ применения QFT к изображениям с использованием квантовых вычислений. Мы исследуем ключевые шаги квантового преобразования в контексте обработки изображений и рассматриваем влияние этого метода на структуру изображений. Также представляем новый аспект в виде вывода квантовой схемы QFT для наглядной демонстрации алгоритма [93].

Эта работа призвана расширить понимание возможностей квантовых вычислений в области обработки изображений и внести вклад в развитие новых методов анализа данных с использованием квантовых технологий. Преимущества квантовых вычислений, такие как параллелизм и суперпозиция, обещают ускорить решение определенных задач, таких как факторизация больших чисел или моделирование сложных молекулярных систем. Это может иметь огромное значение для различных областей, от криптографии до разработки новых материалов и лекарств. Однако на данный момент квантовые вычисления все еще находятся в стадии развития, и существует ряд технических и алгоритмических препятствий, которые необходимо преодолеть, прежде чем они станут широко доступными. Несмотря на это, потенциал квантовых вычислений для революционизации наших возможностей в предсказании физических явлений впечатляет [94,95].

Основная идея квантовых вычислений заключается в использовании квантовых битов, или кубитов, вместо классических

битов. В отличие от классических битов, которые могут находиться в состоянии либо 0, либо 1, кубиты могут находиться в состоянии суперпозиции, что означает, что они могут быть одновременно и 0, и 1. Это свойство суперпозиции позволяет кубитам обрабатывать информацию параллельно, что делает квантовые вычисления в некоторых случаях значительно эффективнее классических. Кроме того, кубиты могут быть связаны друг с другом в явлении, известном как квантовая запутанность, что позволяет им совместно кодировать и обрабатывать информацию. Использование кубитов и их уникальных свойств открывает дверь для создания новых алгоритмов, которые могут решать проблемы более эффективно, чем классические алгоритмы [96].

Одной из основных загадок квантовых вычислений является их хрупкость и сложность. Хотя квантовые компьютеры обещают значительные преимущества в решении определенных задач, они также сталкиваются с рядом технических и физических проблем. Одна из таких проблем - декогеренция, или потеря квантовой суперпозиции из-за воздействия внешней среды. Это может произойти из-за непредсказуемых флуктуаций окружающего электромагнитного поля или тепловых колебаний. Контролирование этих воздействий - сложная задача, которая требует разработки высокоточных методов изоляции и стабилизации квантовых систем. Еще одной проблемой является ошибки в квантовых вычислениях из-за неидеальности квантовых элементов и операций. Например, в квантовых вентилях могут возникать шумы, а в квантовых цепях - ошибки взаимодействия между кубитами. Работа по улучшению алгоритмов коррекции ошибок и разработке более точных квантовых устройств ведется активно, но это все еще остается сложной задачей [97].

Таким образом, хотя квантовые вычисления обещают многое, их успешное внедрение требует преодоления множества технических и физических препятствий, связанных с их хрупкостью и сложностью. Преобразование квантовой информации в формат, совместимый с классическими компьютерами, представляет собой существенное препятствие на пути практического использования квантовых вычислений. Этот процесс, называемый деквантованием, требует разработки специализированных методов и алгоритмов для считывания и интерпретации результатов квантовых вычислений

классическими устройствами. Несмотря на значительные трудности, связанные с деквантованием, важно отметить, что классические компьютеры также совершенствуются в направлении имитации квантовых процессов. Новые алгоритмические стратегии и оптимизации позволяют классическим компьютерам эффективнее моделировать некоторые аспекты квантовых систем, что в определенных случаях может конкурировать с преимуществами, предоставляемыми квантовыми вычислениями [98].

Этот факт подчеркивает важность дальнейшего исследования и развития как квантовых, так и классических методов вычислений. Обе области могут взаимно дополнять друг друга, ведь некоторые задачи могут быть эффективно решены с использованием как квантовых, так и классических подходов. Интеграция классических и квантовых методологий представляет собой потенциально мощный подход для улучшения вычислений и решения сложных задач. Этот подход, известный как гибридные вычисления, использует сильные стороны как классических, так и квантовых вычислений, чтобы решить задачи более эффективно, чем это возможно с помощью любого из методов в отдельности. Например, классические компьютеры могут использоваться для обработки и предварительной подготовки данных, а затем квантовые компьютеры могут применяться для выполнения сложных квантовых алгоритмов, анализа или оптимизации. После этого результаты могут быть обработаны и интерпретированы с помощью классических методов. Такой подход не только позволяет извлечь выгоду из преимуществ обоих типов вычислений, но также может смягчить некоторые из ограничений, с которыми сталкиваются как классические, так и квантовые системы. В результате интеграции мы можем достичь более высокой эффективности и точности в решении сложных задач, от криптографии до оптимизации искусственного интеллекта [99].

Для проведения исследования выбрано цифровое изображение, представляющее интерес для анализа. Изображение может быть как монохромным, так и цветным, чтобы рассмотреть влияние квантового преобразования на различные типы данных. Изображение конвертируется в формат, пригодный для обработки, например, черно-белый. Размер изображения также может быть уменьшен до ближайшей меньшей степени двойки для удобства применения квантового преобразования Фурье (QFT) [100,101].

Каждый пиксель изображения преобразуется в интенсивность, и формируется вектор состояния. Этот вектор подготавливается для применения на квантовой схеме. Реализуется квантовая схема для применения QFT к вектору состояния изображения. Для этого используется библиотека Qiskit для языка программирования Python. Каждый элемент вектора состояния представляет собой амплитуду вероятности на соответствующем кубите [102].

Изучаются воздействие QFT на структуру изображения. Анализируются векторы состояния до и после преобразования, а также полученные изображения. Используется визуализация результатов для более наглядного понимания процесса. Для наглядного представления алгоритма QFT в квантовой схеме, используется библиотека Qiskit. Выводится квантовая схема с учетом применения QFT к входным данным [103].

Эксперимент проводится с различными вариантами изображений и параметров QFT для выявления особенностей и влияния квантового преобразования на структуру входных данных. Для реализации квантовой схемы в программе используется библиотека Qiskit, предоставляющая возможность работы с квантовыми схемами в языке программирования Python. Квантовая схема включает следующие шаги [104]:

Создаются квантовые биты, которые будут использоваться для представления входного вектора состояния изображения. Количество кубитов равно ближайшей меньшей степени двойки к размеру изображения (например, если размер изображения 64x64, то используется 6 кубитов) [105,106].

Квантовая схема, описывающая квантовое преобразование Фурье (QFT), представляется следующей математической формулой [107]:

Для n квантовых битов (кубитов) состояние входного вектора $|x\rangle$ задается как [108]

$$|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i \cdot kx}{2^n}} |k\rangle,$$

где $|k\rangle$ - бинарное представление числа k.

Квантовое преобразование Фурье (QFT) применяется к этому состоянию, и его матрица преобразования QFT_n определяется следующим образом [109]:

$$QFT_n = \frac{1}{\sqrt{2^n}} 1 \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{\frac{2\pi i}{2^n}} & e^{\frac{2\pi i \cdot 2}{2^n}} & \cdots & e^{\frac{2\pi i \cdot (2^n-1)}{2^n}} \\ 1 & e^{\frac{2\pi i \cdot 2}{2^n}} & e^{\frac{2\pi i \cdot 4}{2^n}} & \cdots & e^{\frac{2\pi i \cdot 2(2^n-1)}{2^n}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{\frac{2\pi i \cdot (2^n-1)}{2^n}} & e^{\frac{2\pi i \cdot 2(2^n-1)}{2^n}} & \cdots & e^{\frac{2\pi i \cdot (2^n-1)^2}{2^n}} \end{bmatrix},$$

где каждый элемент матрицы $QFT_n[j,k]$ вычисляется по формуле:

$$QFT_n[j,k] = \frac{1}{\sqrt{2^n}} e^{\frac{2\pi i \cdot jk}{2^n}},$$

где j и k принимают значения от 0 до $2^n - 1$

Преобразование Фурье изображений широко применяется в цифровой обработке изображений и видео из-за его эффективности и мощности в анализе и обработке сигналов. Преобразование Фурье изображения преобразует изображение из пространственной области в частотную область. Это означает, что оно переводит изображение из представления пикселей в представление частот. Конкретно, преобразование Фурье изображения позволяет разложить изображение на набор синусоидальных (или косинусоидальных) функций различных частот. Эти частоты представляют различные детали и структуры в изображении.

После выполнения преобразования Фурье изображение представлено в двумерном частотном пространстве, где низкие частоты расположены в центре, а высокие частоты - на периферии. Таким образом, изображение в частотной области позволяет анализировать содержащиеся в нем частотные компоненты, что может быть полезно для таких задач, как фильтрация шума, улучшение резкости, обнаружение границ и др. Визуализация преобразования Фурье изображения позволяет увидеть вклад различных частот в изображение, что может помочь в понимании его структуры и характеристик.

Этот код создает квантовую схему, которая применяет квантовое преобразование Фурье к изображению. Все пиксели изображения представлены как квантовые биты, и затем применяется преобразование Фурье.

Программа начинается с загрузки изображения в оттенках серого из файла. Мы используем библиотеку OpenCV (cv2) для этой цели.

Исходное изображение отображается на первом графике с помощью Matplotlib. Это позволяет нам визуализировать, как выглядит изображение до применения преобразования Фурье.

Определение количества битов, которые необходимы для представления каждого пикселя изображения в бинарной форме. Это нужно для определения размера квантовой схемы.

Создание квантовой схемы для преобразования Фурье с помощью библиотеки Qiskit. В этой схеме мы применяем квантовое преобразование Фурье (QFT). Для этого используются управляющие вращения (cp) и вращения Хадамара (H).

Отображение квантовой схемы с помощью функции circuit_drawer из библиотеки Qiskit. Мы используем параметр output='mpl', чтобы получить цветной графический вывод.

Симуляция квантовой схемы. Мы используем симулятор Qiskit, чтобы выполнить квантовую схему. В данном случае мы используем симулятор statevector_simulator, который строит точное состояние квантовой системы.

Визуализация результатов: Мы получаем результаты симуляции в виде количества измерений для каждого возможного состояния квантовых битов. Затем мы используем функцию plot_histogram из библиотеки Qiskit для построения гистограммы этих результатов.

Таким образом, программа загружает изображение, применяет к нему квантовое преобразование Фурье с использованием квантовой схемы, симулирует результаты и визуализирует как саму схему, так и результаты, а также отображает изображение до и после преобразования Фурье (рис4.1,4.2).

Разработана программа квантового преобразование Фурье (QFT) для изображения, представленного в виде вектора интенсивностей пикселей. Квантовая схема, представленная в данной программе, выполняет преобразование Фурье на квантовых кубитах. Преобразование Фурье - это математическое преобразование, которое находит применение в различных областях, включая классическую и квантовую информатику. Квантовая схема использует квантовые биты, которые являются квантовыми аналогами классических битов.

Кубиты могут находиться в линейной комбинации состояний $|0\rangle|0\rangle$ и $|1\rangle|1\rangle$ из-за явления квантового вмешательства [109].

Преобразование Фурье в данном контексте выполняется на квантовых битах. Это преобразование является квантовым аналогом классического преобразования Фурье, которое применяется к последовательности значений.

Рис.4.1. Квантовая схема преобразования Фурье

В схеме используются квантовые вентили, такие как вентили Адамара (H), управляемые фазовые вентили (R_ϕ), и контролируемые вентили (C).

Начальное состояние квантовой системы инициализируется изображением, а затем выполняется квантовое преобразование. После этого производится измерение кубитов, чтобы получить классическую информацию. Результаты квантового преобразования визуализируются с помощью графиков, позволяя увидеть как изменяется вектор состояния и изображение после преобразования. Такая квантовая схема может использоваться для исследования квантовых преобразований на изображениях и в контексте квантовых алгоритмов обработки данных.

В данной программе производится квантовое преобразование Фурье (QFT) для изображения, представленного в виде вектора интенсивностей пикселей. Рассмотрим результаты каждого этапа программы:

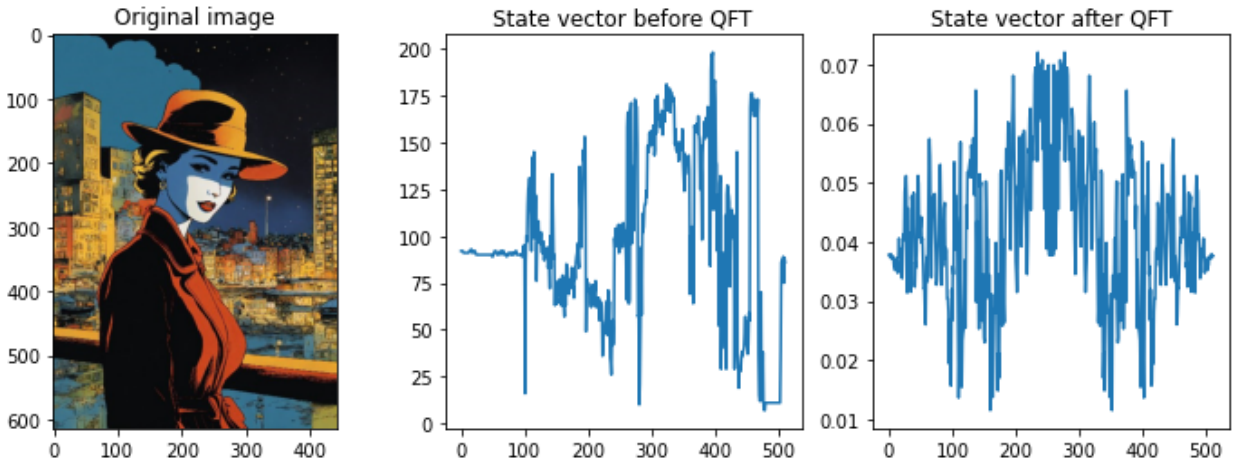
Отображается оригинальное изображение в оттенках серого.

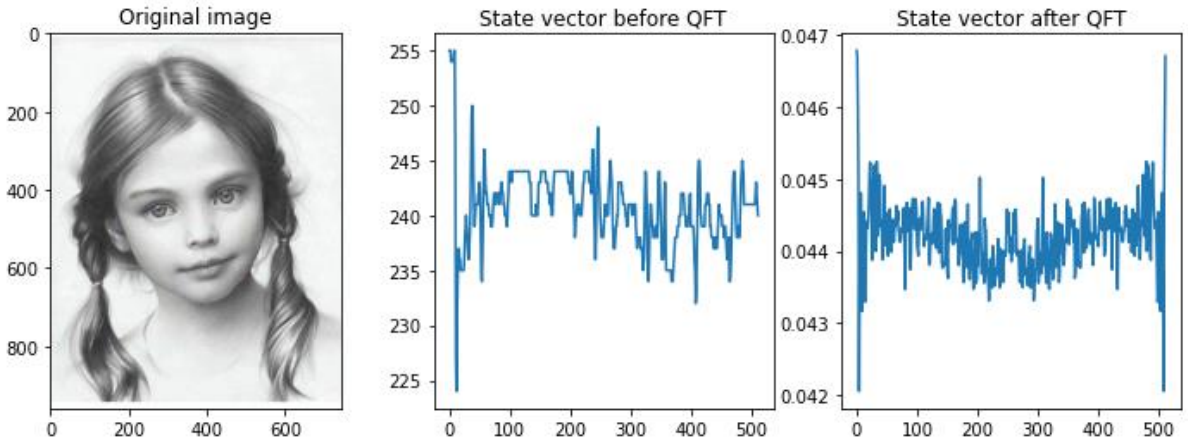
Вектор состояния до QFT показывает значения интенсивностей пикселей, представленные в виде вектора состояния до применения квантового преобразования.

Вектор состояния после QFT Отображает вектор состояния после применения квантового преобразования. Этот вектор показывает, как изменились амплитуды состояний после применения QFT.

Преобразованное изображение после QFT представляет собой визуализацию преобразованного изображения после квантового преобразования. Изображение формируется на основе амплитуд состояний квантового бита.

Построение гистограммы амплитуд для визуального анализа распределения амплитуд состояний после квантового преобразования.





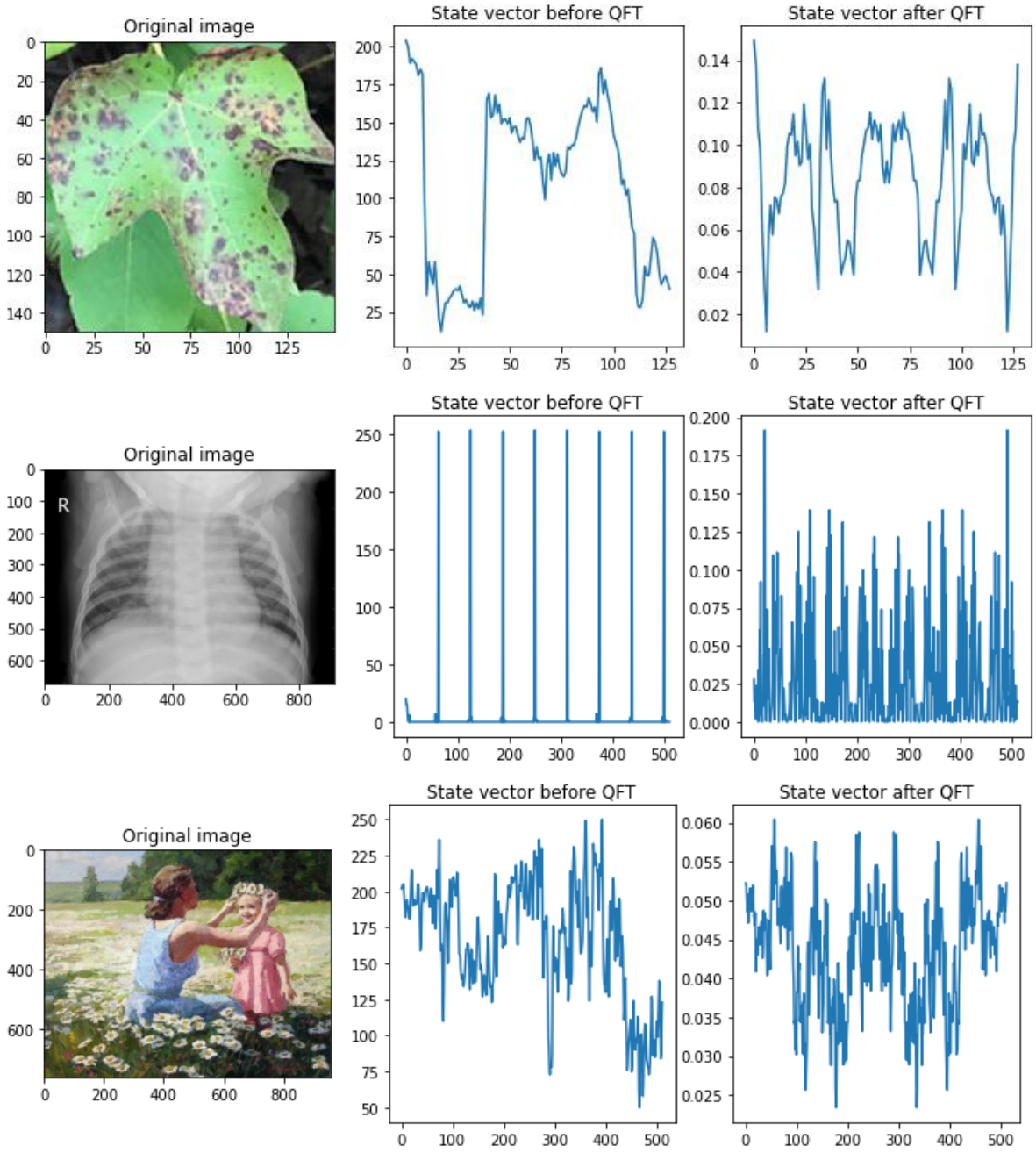


Рис.4.2. Квантовое преобразование Фурье

Важно отметить, что результаты программы могут зависеть от выбора количества квантовых битов и самого изображения. Также, преобразование Фурье может выделять определенные паттерны в изображении.

В данной работе был проведен анализ применения квантового преобразования Фурье (QFT) в контексте обработки изображений. Результаты экспериментов показывают, что квантовое преобразование способно воздействовать на структуру изображений, приводя к изменению амплитуд пикселей. Эффективность квантового преобразования оценивалась на примере изображений с

использованием различного числа квантовых битов. Увеличение числа битов приводило к более высокой детализации преобразования, однако требовало больших ресурсов вычислений. Гистограмма амплитуд после QFT может служить индикатором важных характеристик изображения. Однако необходимо учитывать, что результаты могут сильно зависеть от самого изображения и параметров квантовой схемы. Также были выделены области применения, где квантовые методы обработки изображений могут быть наиболее полезными, такие как выделение особых точек и шаблонов. Однако следует отметить, что квантовые методы обработки изображений имеют свои ограничения, и эффективность таких подходов может быть улучшена с учетом дальнейших исследований и оптимизаций.

Эти результаты подчеркивают перспективность использования квантовых подходов в обработке изображений, предоставляя новые возможности для анализа и изменения визуальных данных. Дальнейшие исследования в этой области помогут оптимизировать параметры квантовых схем и расширить область их применения в практических задачах обработки изображений

4.2. Квантовые методы анализа и обработки изображений в частотной области

Квантовые методы анализа и обработки изображений в частотной области представляют собой инновационный подход к обработке изображений с использованием квантовых вычислений. Одним из ключевых инструментов в этой области является квантовое преобразование Фурье (QFT), которое может быть применено к квантовым изображениям для анализа и обработки данных в частотной области. Этот процесс включает в себя несколько этапов, включая подготовку квантового изображения, применение QFT, обработку данных в частотной области и обратное преобразование Фурье для восстановления изображения в пространственной области. Квантовые методы обработки изображений предлагают новые возможности для решения сложных задач в области обработки изображений, особенно в контексте использования квантовых вычислений. Однако данная область все еще находится в стадии

активного исследования, и требует дальнейших исследований и развития технологий для практического применения.

С развитием квантовых вычислений и квантовой информатики в последние десятилетия активно исследуется возможность применения квантовых методов анализа и обработки изображений. Квантовые методы обработки изображений представляют собой новую и перспективную область, которая может привести к значительным улучшениям в обработке информации и анализе данных. Одним из ключевых инструментов в квантовой обработке изображений является квантовое преобразование Фурье (QFT), которое является аналогом классического преобразования Фурье, но может быть выполнено на квантовом компьютере с использованием квантовых битов (кьюбитов). Применение QFT позволяет анализировать изображения в частотной области, что открывает новые возможности для обработки и анализа данных [111].

Рассмотрим актуальные исследования и разработки в области квантовых методов анализа и обработки изображений в частотной области. Мы рассмотрим методы представления изображений в квантовой форме, применение квантового преобразования Фурье для анализа изображений, а также различные алгоритмы и техники обработки изображений в квантовой среде. В конце мы обсудим перспективы и вызовы в этой области и возможные направления будущих исследований. Актуальность исследования в области квантовых методов анализа и обработки изображений в частотной области обусловлена растущим интересом к применению квантовых вычислений в различных областях науки и техники. Квантовые вычисления обещают революционизировать обработку информации благодаря своей способности эффективно обрабатывать большие объемы данных с использованием квантовых принципов суперпозиции и квантовой запутанности. В контексте обработки изображений, классические методы, такие как преобразование Фурье, широко применяются для анализа и обработки изображений в частотной области. Однако, квантовые методы обработки изображений представляют собой новый подход, который может обеспечить более эффективные и мощные инструменты для анализа и обработки изображений. В частности, квантовое преобразование Фурье (QFT) представляет собой версию классического преобразования Фурье, которое может быть выполнено на квантовом

компьютере. Применение QFT к изображениям позволяет анализировать их в частотной области, что может быть полезно для выделения основных компонентов изображения, фильтрации шумов, компрессии данных и других задач обработки [113].

Применение квантовых методов анализа и обработки изображений представляет интерес как с теоретической, так и с практической точек зрения. С одной стороны, это способствует развитию основных принципов квантовых вычислений и их применению в области обработки изображений. С другой стороны, это открывает новые возможности для решения сложных задач обработки изображений, которые могут быть применены в различных областях, таких как медицина, биология, астрономия и многие другие. Однако стоит отметить, что квантовые методы обработки изображений все еще находятся на ранней стадии исследований, и требуют дальнейших исследований и развития технологий для их практического применения. В настоящее время, основными вызовами являются разработка эффективных алгоритмов и программного обеспечения, а также создание квантовых устройств, способных обрабатывать изображения с высокой точностью и скоростью [114,115].

Квантовые компьютеры вносят революционные изменения в область вычислений. Вместо традиционного использования битов, квантовые компьютеры оперируют кубитами, которые благодаря квантовым явлениям суперпозиции и запутанности могут существовать в нескольких состояниях одновременно. Кубиты являются основными строительными блоками квантовых вычислений. В отличие от классического бита, который может быть либо 0, либо 1, кубит может находиться в состоянии 0, 1 или одновременно в обоих состояниях. Эта уникальная способность квантовых компьютеров выполнять множество вычислений одновременно делает их весьма эффективными в решении определенных типов задач. Новые возможности, которые открываются с появлением квантовых компьютеров, позволяют решать сложные задачи быстрее и эффективнее, чем это возможно с использованием классических компьютеров [116-118].

Сложность обеспечения квантового преимущества велика, особенно учитывая восприимчивость квантовых компьютеров к ошибкам. Квантовые компьютеры обладают специфическими

характеристиками, такими как квантовая запутанность и суперпозиция, которые делают их чувствительными к внешним воздействиям. Даже малейшие помехи или ошибки в квантовых операциях могут привести к искажению результатов или потере информации. Разработка и обслуживание квантовых систем требует высокой степени точности и надежности. Это включает в себя создание стабильных квантовых элементов, а также разработку алгоритмов коррекции ошибок и методов управления квантовыми состояниями. Несмотря на эти сложности, исследования и разработки в области квантовых вычислений активно продолжаются. Ученые и инженеры по всему миру работают над улучшением квантовых систем, сокращением ошибок и разработкой новых методов, чтобы преодолеть эти препятствия. Гибридные подходы, объединяющие классические и квантовые методы, могут помочь уменьшить влияние ошибок на конечные результаты, обеспечивая более надежные и эффективные решения задач [119] .

Важный аспект в развитии квантовых вычислений - сложность достижения квантового преимущества при использовании квантовых компьютеров, которые подвержены ошибкам. Это одно из ключевых вызовов, стоящих перед исследователями и инженерами в этой области. Ошибка в квантовых вычислениях может возникнуть из-за различных факторов, включая шумы в квантовых элементах, воздействие окружающей среды и тепловые флуктуации. Кроме того, сами операции квантовых вычислений могут быть подвержены ошибкам из-за неидеальности квантовых вентилей и кубитов. Для преодоления этой проблемы и достижения квантового преимущества необходимо разработать эффективные методы коррекции ошибок, а также совершенствовать технологии и алгоритмы управления квантовыми системами. Это требует не только технической экспертизы, но и глубокого понимания физических принципов квантовой механики и методов алгоритмической оптимизации. Несмотря на сложности, исследования в области квантовых вычислений продолжают развиваться, и новые достижения и технологические решения помогают постепенно преодолевать эти вызовы. Актуальным является актуальным использованием тензорных сетей, особенно с учетом роста интереса к оптимизации классических вычислений. Тензорные сети имеют большой потенциал для моделирования и анализа сложных систем, таких как

квантовые состояния, и их эффективная оптимизация является важным шагом в развитии вычислительных методов [120-122].

Использование методов, адаптированных на основе статистических выводов, для оптимизации тензорных сетей может значительно улучшить их эффективность. Эти методы могут включать в себя различные подходы, такие как стохастический градиентный спуск или методы оптимизации на основе алгоритмов машинного обучения, которые позволяют находить оптимальные параметры сетей с учетом статистических свойств данных. Это имеет прямое применение к работе с квантовыми системами, где тензорные сети могут использоваться для моделирования и анализа взаимодействия кубитов и других квантовых объектов. Эффективная оптимизация таких сетей может улучшить точность и скорость вычислений, что важно для развития квантовых вычислений и их приложений [123-124].

Квантовое преобразование Фурье (QFT) представляет собой версию классического преобразования Фурье, которое может быть выполнено на квантовом компьютере. Оно имеет множество В квантовых вычислениях QFT может быть использовано для анализа и обработки квантовых изображений. Квантовые изображения представляют собой квантовые состояния, которые могут содержать информацию о распределении вероятностей значений пикселей или других характеристик изображения. Изображение кодируется в квантовые состояния, используя квантовые биты (кьюбиты). Это может быть сделано, например, с помощью квантовых схем сопряжения. Квантовое преобразование Фурье применяется к состояниям, представляющим изображение. Это позволяет анализировать изображение в частотной области, выделяя его основные составляющие. После применения QFT к изображению можно выполнить различные операции обработки, такие как фильтрация, сглаживание или выделение определенных частотных компонент. После обработки изображения в частотной области, можно выполнить обратное квантовое преобразование Фурье, чтобы вернуть изображение в пространственную область [125-126].

Этот процесс может быть особенно полезен для обработки изображений с использованием квантовых вычислений, особенно в случаях, когда классические методы обработки изображений ограничены или неэффективны. Однако стоит отметить, что в

настоящее время квантовые компьютеры находятся на ранней стадии развития, и практическое применение квантовых методов обработки изображений может потребовать дальнейших исследований и развития технологий. Подобно тому, как сжатие изображений уменьшает размер файла с минимальными потерями качества, выбор различных структур для тензорных сетей обеспечивает различные формы вычислительного "сжатия", оптимизируя способы хранения и обработки информации [127].

Таким образом, как JPEG обеспечивает эффективное сжатие изображений, тензорные сети позволяют нам эффективно хранить и обрабатывать данные, сокращая размеры и уменьшая вычислительные затраты при минимальной потере информации [128].

Преобразование Фурье изображений (FFT - Fast Fourier Transform) является одним из основных методов анализа и обработки изображений в частотной области. Оно позволяет анализировать изображения в терминах их частотных составляющих, что может быть полезно для таких задач, как фильтрация шума, улучшение резкости, обнаружение границ и компрессия данных. Преобразование Фурье изображения работает, разбивая изображение на его частотные компоненты. В результате получается двумерный спектр, в котором высокие частоты представлены как яркие точки, а низкие частоты как темные области. Исходное изображение подвергается подготовке, если это необходимо (например, масштабирование, приведение к размеру, кратному степени двойки для эффективной работы FFT). Применяется двумерное преобразование Фурье к изображению. Для этого обычно используется алгоритм FFT, который эффективно вычисляет преобразование Фурье за счет оптимизации операций. Полученный результат представляет из себя спектр изображения в частотной области. Здесь можно анализировать компоненты частот и их вклад в исходное изображение. Можно выполнять различные операции обработки в частотной области, такие как фильтрация (например, удаление низкочастотного или высокочастотного шума), улучшение резкости и другие. После обработки изображения в частотной области можно применить обратное преобразование Фурье, чтобы вернуть изображение к пространственной области.

Представленная квантовая схема представляет собой квантовое преобразование Фурье (QFT) для 7 кубитов. В квантовых вычислениях QFT является важным инструментом и используется

для преобразования состояний кубитов из базового (входного) представления в представление Фурье. Вот пошаговое описание квантовой схемы:На первом кубите (q_3) применяется вращение Хадамара (H). Это вращение переводит кубит из состояния |0⟩ в суперпозицию (|0⟩ + |1⟩) / √2. Затем происходит контролируемое вращение (cp) между кубитами q_4 и q_3 с углом π/2 (P(π/2)). Это вращение зависит от состояния q_3 и применяется к q_4 при условии, что q_3 находится в состоянии |1⟩. На кубитах q_5, q_6 и q_7 применяются вращения Хадамара (H), а также контролируемые вращения с углами π/2, π/4, π/8 и π/16. Эти вращения формируют квантовое преобразование Фурье на этих трех кубитах. Наконец, на кубитах q_5, q_6 и q_7 применяются контролируемые вращения (cp) между каждой парой кубитов с углами π, π/2 и π/4 соответственно (рис.4.3).

Эта схема реализует QFT для 7 кубитов и может быть использована в различных квантовых вычислительных задачах, таких как квантовая алгебра, алгоритм Шора и другие. В квантовых вычислениях "cp" обозначает контролируемое вращение, где управляющий кубит контролирует вращение целевого кубита в зависимости от своего состояния. Формально это может быть представлено как операция:

$$CP(\lambda) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\lambda} \end{bmatrix}$$

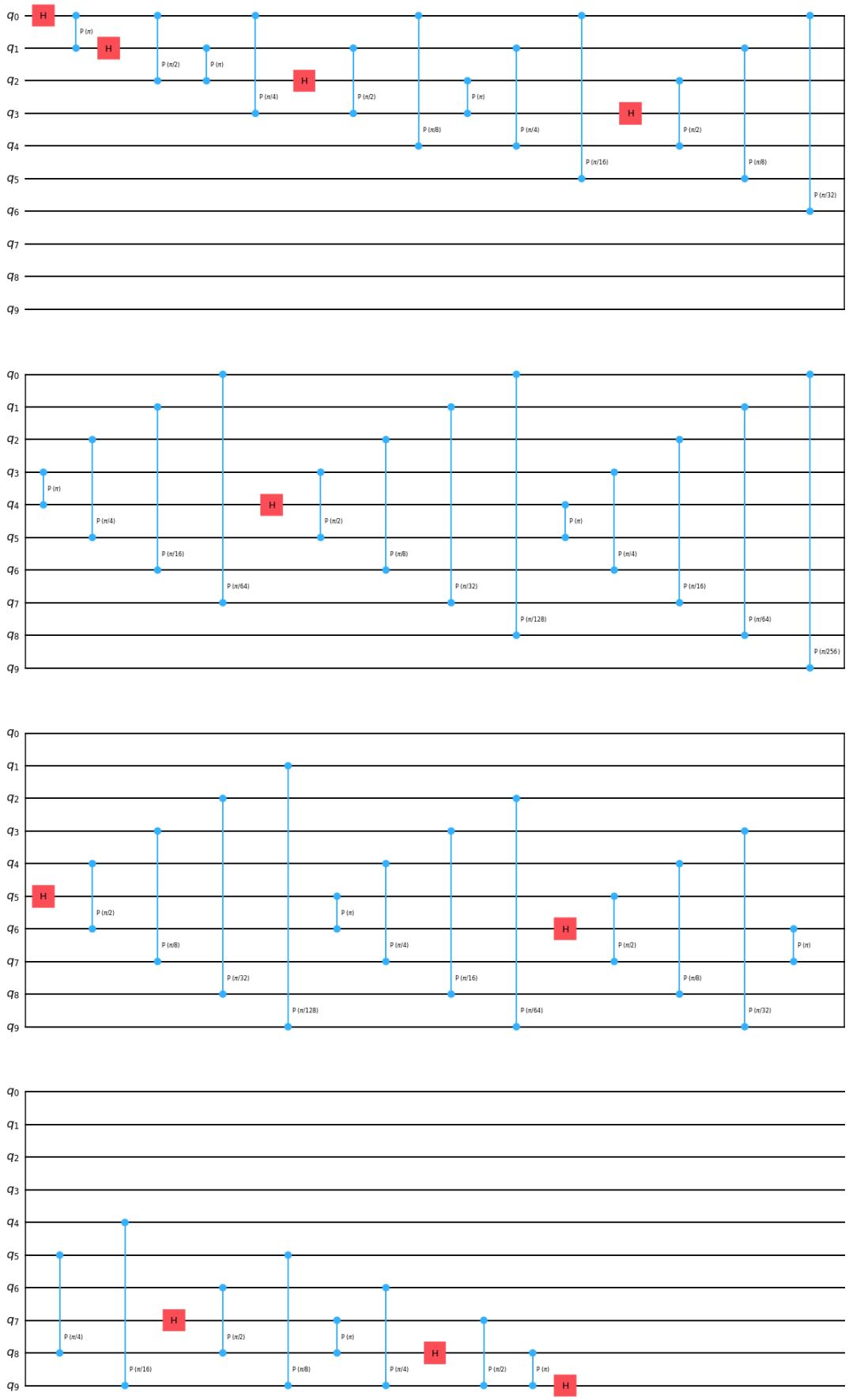


Рис.4.3. Квантовая схема преобразования Фурье

Здесь λ - параметр вращения, который зависит от состояния управляющего кубита. Если управляющий кубит находится в состоянии $\lambda|1\rangle|1\rangle$, то целевой кубит подвергается вращению на угол λ, в противном случае никаких изменений не происходит. В схеме углы вращения $2\pi/2$, $4\pi/4$, $8\pi/8$ и $16\pi/16$ используются в операции CP для создания квантового преобразования Фурье. Обычно $P(\theta)$ обозначает однокубитовое вращение вокруг оси Z на угол θ. Формально это матрица поворота:

$$P(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}.$$

В схеме $P(\pi/2)$ означает однокубитовое вращение вокруг оси Z на угол $2\pi/2$. Такие вращения важны в квантовых вычислениях, так как они позволяют изменять фазу состояний кубитов. В контексте квантовых вычислений $e^{i\theta}$ часто используется для представления вращений на комплексной плоскости, таких как вращения кубитов вокруг оси Z

Это представление может быть выражено с помощью синуса и косинуса. Матрица поворота $P(\theta)$ вокруг оси Z может быть записана как:

$$P(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}.$$

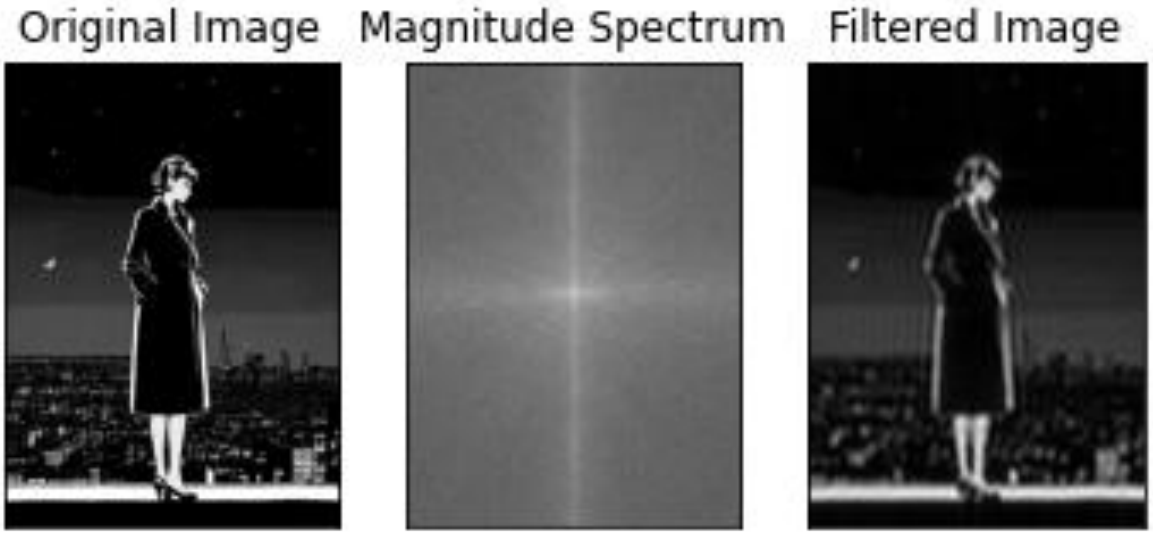
Эта матрица описывает поворот на угол θ в комплексной плоскости, где θ измеряется в радианах.

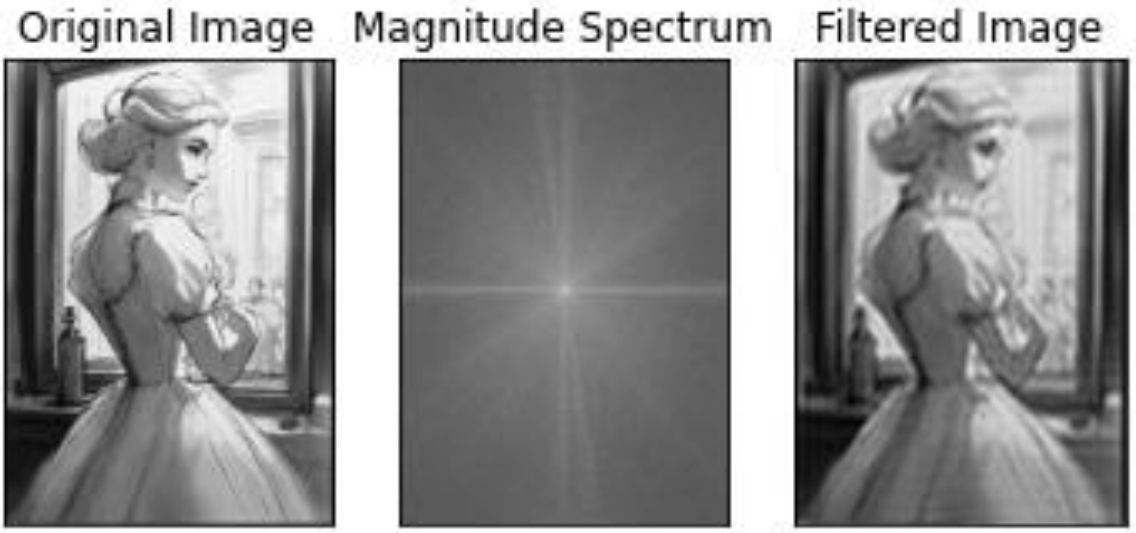
При выводе изображения до квантового преобразования Фурье используется библиотека Matplotlib (plt) для отображения изображения до преобразования Фурье. Функция plt.imshow() используется для отображения изображения, а plt.title() - для установки заголовка. Для преобразования каждого пикселя изображения используется число кубитов, которое вычисляется как логарифм по основанию 2 от максимального значения размера изображения. Это гарантирует, что количество кубитов будет достаточным для представления каждого пикселя (рис.2).

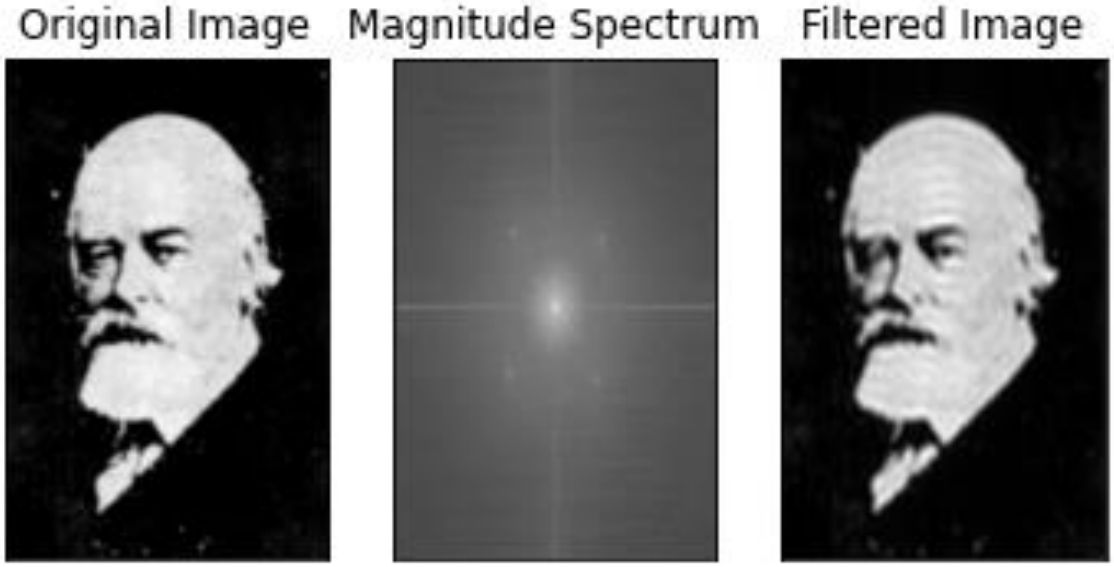
Для каждого кубита в квантовой схеме применяется гаджет Адамара (qc.h(j)), а затем управляемый поворот (qc.cp()) применяется между всеми кубитами для реализации квантового преобразования Фурье. Схема отображается с помощью circuit_drawer() из Qiskit,

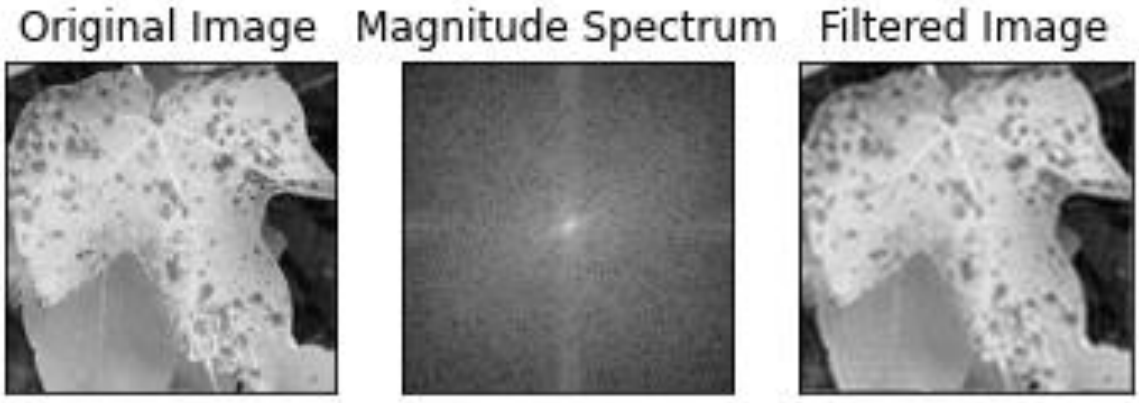
чтобы можно было увидеть порядок операций. Используется бэкэнд statevector_simulator из Aer для симуляции квантовой схемы. Полученные результаты представлены в форме счетов (counts).

Используется plot_histogram() из Qiskit для визуализации результатов в виде гистограммы. Вывод изображения после квантового преобразования Фурье: Используется plt.imshow() для отображения изображения после преобразования Фурье. Для преобразования каждого пикселя изображения используется число кубитов, которое вычисляется как логарифм по основанию 2 от максимального значения размера изображения. Это гарантирует, что количество кубитов будет достаточным для представления каждого пикселя. Создается объект квантовой схемы с помощью QuantumCircuit из библиотеки Qiskit. Для каждого кубита в квантовой схеме применяется гаджет Адамара (qc.h(j)), а затем управляемый поворот (qc.cp()) применяется между всеми кубитами для реализации квантового преобразования Фурье. Схема отображается с помощью circuit_drawer() из Qiskit, чтобы можно было увидеть порядок операций. Визуализация результатов: Используется plot_histogram() из Qiskit для визуализации результатов в виде гистограммы.




Original Image
Magnitude Spectrum
Filtered Image


Original Image
Magnitude Spectrum
Filtered Image


Original Image
Magnitude Spectrum
Filtered Image

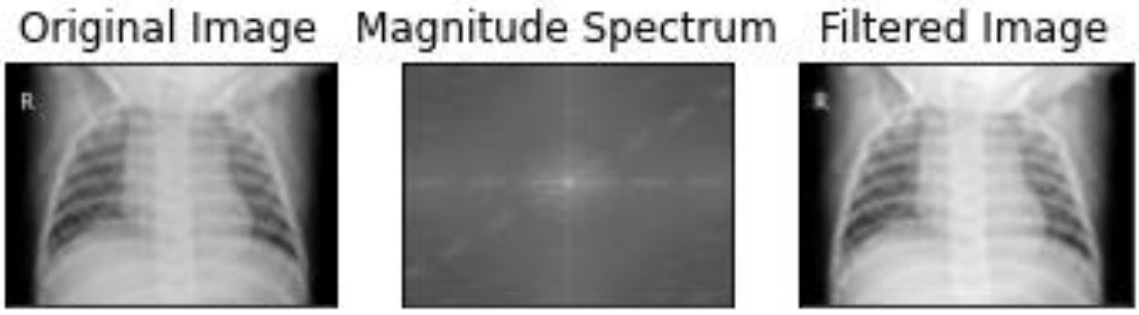


Рис.4.4. Результат квантового преобразования Фурье

Преобразование Фурье, применяемое в этой программе, выполняет перенос изображения из пространства пикселей в пространство частот: Преобразование Фурье переводит изображение из пространства пикселей в пространство частот, где каждый пиксель изображения представлен как сумма различных частотных компонент. Это позволяет анализировать изображение с точки зрения его составных частей в частотной области. Преобразование Фурье выделяет основные частотные компоненты изображения, позволяя определить наиболее значимые элементы в изображении с точки зрения их частотных составляющих. Например, резкие края и контуры изображения могут быть выражены как высокочастотные компоненты. Преобразование Фурье позволяет эффективно компактно представлять изображение в виде его частотных компонент. Это может быть полезно для сжатия изображений с минимальной потерей качества. Преобразование Фурье используется для обработки и фильтрации изображений, таких как удаление шумов или наложение эффектов размытия. Применение фильтров в частотной области позволяет контролировать различные аспекты изображения, такие как резкость и текстура.

Таким образом, преобразование Фурье выполняет важные задачи, связанные с анализом, обработкой и представлением изображений в частотной области, что делает его мощным инструментом в области обработки изображений и компьютерного зрения. Программы не используются для обработки изображений, таких как удаление шумов или наложение эффектов размытия. Вместо этого, они применяют квантовое преобразование Фурье (QFT) к изображению, чтобы исследовать его в частотной области с использованием квантовых схем. Преобразование Фурье в контексте квантовых вычислений помогает анализировать изображение в квантовом пространстве, что

может быть полезным для определения основных частотных компонент, характеризующих структуру изображения. Однако в данной программе применение QFT не предназначено для обработки изображений в классическом смысле, так как результат преобразования не используется для модификации или фильтрации изображения. Применение квантового преобразования Фурье (QFT) в данной программе используется для анализа структуры изображения в частотной области с помощью квантовых схем. Это может быть полезным для ряда задач, таких как квантовая обработка сигналов: QFT позволяет анализировать частотные компоненты сигнала в квантовой системе. Это может быть полезно, например, для анализа спектров сигналов в квантовых коммуникациях или обработке сигналов в квантовых сенсорных устройствах. Преобразование Фурье играет важную роль в алгоритмах квантовой информатики, таких как алгоритм Шора для факторизации больших чисел и алгоритмы для решения задач оптимизации. В квантовых вычислениях QFT может использоваться для анализа и обработки данных, в том числе изображений. Например, он может помочь в выявлении основных структурных элементов изображения или сегментации изображения по частотным характеристикам.

В данной программе применение QFT позволяет анализировать структуру изображения в квантовом контексте и может быть полезным для исследования основных частотных компонент, характеризующих изображение. Программу, которую вы предоставили, используют для применения квантового преобразования Фурье к изображению и его последующего анализа. Однако само применение квантового преобразования Фурье не является методом удаления шума. Для удаления шума с изображения обычно применяют различные методы обработки изображений, такие как фильтрация, сглаживание или размытие. Эти методы могут использоваться для сглаживания значений пикселей, снижения высокочастотного шума или уменьшения различий между соседними пикселями. Хотя квантовые методы также могут быть применены в обработке изображений, квантовое преобразование Фурье само по себе не является методом удаления шума. Для удаления шума с квантовых изображений можно использовать классические методы фильтрации или разработать специализированные квантовые алгоритмы.

4.3. Алгоритм преобразования изображения в квантовое состояние

В данной работе исследуется и разрабатывается метод функционирования квантовых алгоритмов и моделей квантовых вычислительных устройств с целью преобразования классического изображения в квантовое состояние, выделения границ и преобразования полутонового изображения в бинарное. Представленный квантовый алгоритм демонстрирует перспективы применения квантовой теории информации для решения классических задач. Основной акцент работы делается на компьютерном моделировании квантового алгоритма и исследовании его эффективности при использовании квантовых вычислительных средств и методов. Актуальность данного исследования вытекает из постоянного обновления и дополнения области квантовых исследований, а также из нехватки компьютерных симуляций квантовых физических явлений. Научная новизна работы состоит в разработке эффективной модели распознавания образов с использованием свойств и методов квантовых вычислений, что вносит вклад в развитие данной области научных исследований.

В настоящее время наблюдается быстрое развитие квантовых вычислений и квантовых вычислительных устройств. В современной эпохе научных и технических исследований выявляется нарастающая потребность в разрешении стратегически важных задач, охватывающих широкий спектр областей, таких как метеорология, медицина, астрономия, обработка сигналов и криптоанализ. Нередко эти задачи представляют собой вычислительные проблемы, требующие огромных вычислительных ресурсов и времени для их решения на существующих компьютерных системах, включая суперкомпьютеры. Некоторые из них, такие как задачи NP-полноты, вовсе не поддаются эффективному решению на классических компьютерах. В последние годы наблюдается увеличение интереса к квантовым компьютерам в связи с их потенциальной способностью преодолевать эти ограничения. Квантовые компьютеры представляют собой уникальные устройства, основанные на принципах квантовой механики, которые способны проводить параллельные вычисления на масштабах, недоступных для классических компьютеров. Это открывает перспективы значительного увеличения скорости решения

сложных вычислительных задач, включая NP-полные проблемы, которые на современных компьютерах могут решаться либо недопустимо долго, либо вовсе не решаются. Применение квантовых алгоритмов и вычислений позволяет не только ускорить процесс решения задач, но и преодолеть некоторые технические ограничения, характерные для классических методов. Например, квантовые алгоритмы могут быть более устойчивыми к погрешностям баз данных, изменениям условий освещения или средствам маскировки объектов. Развитие квантовых компьютеров открывает новые перспективы для науки и техники, позволяя эффективно решать сложные проблемы, которые оставались неразрешенными или недоступными ранее из-за ограничений классических компьютерных систем [111].

Исследования в области применения квантовых методов для решения классических задач, таких как обработка изображений, являются важным шагом в освоении потенциала квантовых технологий. Использование квантовых алгоритмов может привести к значительному увеличению скорости и эффективности вычислений в сравнении с классическими методами. Это особенно важно в задачах обработки изображений, где требуется обработка больших объемов данных. Данное исследование может помочь заполнить пробел в компьютерных симуляциях квантовых физических явлений, что важно для более глубокого понимания квантовых процессов и их приложений. Разработка эффективных методов преобразования классических изображений с использованием квантовых алгоритмов имеет прямое практическое применение в области обработки изображений, распознавания образов и компьютерного зрения. Все эти факторы делают данное исследование актуальным и важным для современной науки и технологий [112-115].

Современное развитие квантовых вычислений и квантовых технологий открывает новые перспективы в области обработки изображений и распознавания образов. Одним из интересных направлений в этой области является преобразование классических изображений в квантовые состояния с последующим выделением границ и преобразованием полутонового изображения в бинарное. В данной работе мы рассматриваем реализацию алгоритма преобразования классического изображения в квантовое состояние, а также методы выделения границ и преобразования полутонового

изображения в бинарное с использованием квантовых вычислительных средств и методов. Этот алгоритм имеет широкий потенциал применения в области обработки изображений, компьютерного зрения и распознавания образов. Целью данного исследования является разработка эффективной модели преобразования классических изображений с использованием квантовых алгоритмов и выявление их преимуществ по сравнению с классическими методами. Мы также стремимся исследовать возможности применения квантовой теории информации в интерпретации классических задач обработки изображений. В данной работе мы подробно описываем реализацию алгоритма преобразования классического изображения в квантовое состояние, выделения границ и преобразования полутонового изображения в бинарное, и представляем результаты вычислительного эксперимента, сравнивающие эффективность квантового подхода с классическими методами. Наша работа имеет как теоретическое, так и практическое значение, внося вклад в развитие квантовых методов обработки изображений и расширяя возможности их применения в реальных задачах [116-118].

Процесс формирования набора кубитов представляет собой важный этап в реализации квантовых вычислений. Вычислительный процесс предполагает применение различных методов квантовых вычислений для выполнения квантовых алгоритмов и формирование набора кубитов для состояния управляющих сигналов нормализации в определенный момент времени. В данном контексте "нормализация" обычно относится к процессу приведения состояний квантовых систем к стандартному виду или к нормальной форме, что может быть важным для обеспечения корректного выполнения квантовых операций и алгоритмов. Формирование набора кубитов включает в себя создание определенного количества кубитов, которые будут использоваться для представления информации и выполнения операций в квантовых вычислениях. Этот набор кубитов $|0\rangle$ и $|1\rangle$ может быть подготовлен с учетом требуемых квантовых операций, алгоритмов и их спецификаций. Для этих двух базисных кубитов векторное представление следующее [7-10]:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Произвольный кубит можно разложить в базисе, и такое разложение записывается как $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$, и оно называется линейной суперпозицией базисных состояний, и, соответственно, в виде вектора такой кубит представляется как:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

Формирования набора кубитов в контексте моделирования запутанных квантовых вычислений играет ключевую роль в реализации квантовых алгоритмов и обеспечении их правильного выполнения.

Применение тензорного произведения между преобразованиями Адамара может привести к формированию членов вида $n^2 H_n \otimes KP \otimes K$, где H_n представляет преобразование Адамара, KP и K обозначают аналогичные комбинации коэффициентов усиления.

Гейт Адамара очень примечателен тем, что переводит кубиты из стандартного вычислительного базиса в базис $\{|+\rangle, |-\rangle\}$ и обратно. Этот гейт обозначается как H и имеет следующую матрицу [111-114]:

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

Например, применение тензорного произведения между двумя преобразованиями Адамара может привести к образованию шестнадцати вероятностных состояний, описывающих различные вариации корреляций в соответствии с их типом и видом. Таким образом, применение тензорного произведения между преобразованиями Адамара позволяет рассмотреть различные комбинации коэффициентов усиления и создать разнообразные вероятностные состояния, которые могут быть полезными для анализа и моделирования в контексте соответствующих задач квантовых вычислений [115-118].

Кодирование цветовой палитры пиксельного набора в комплексные амплитудные квантовые состояния предполагает преобразование каждого пикселя изображения в некоторое квантовое состояние. Для этого каждый пиксель $x(i, j)$ преобразуется в квантовое состояние $|q(i, j)\rangle$ которое может быть представлено как суперпозиция

базовых состояний $|0\rangle$ и $|1\rangle$ с соответствующими комплексными амплитудами α и β. Формально, квантовое состояние пикселя $|q(i,j)\rangle$ записывается как:

$$|q(i,j)\rangle = \alpha|0\rangle + \beta|1\rangle,$$

где $|0\rangle$ и $|1\rangle$ представляют базовые состояния кубита, а α и β - комплексные амплитуды, определяющие вероятности соответствующих базовых состояний [119-121].

Этот подход позволяет представить информацию о цвете каждого пикселя в виде квантовой суперпозиции базовых состояний. Таким образом, цветовая информация изображения кодируется в комплексных амплитудах квантовых состояний, что открывает возможности для использования квантовых методов при обработке и анализе цветных изображений [122-123].

Далее, создается суперпозиция вычислительного процесса, представленного как:

$$|\psi\rangle = \sum_{i,j} |q(i,j)\rangle \otimes |x(i,j)\rangle.$$

Это означает, что каждый пиксель изображения (описанный квантовым состоянием $|q(i,j)\rangle$ связывается с соответствующей координатной сеткой (квантовым состоянием $|x(i,j)\rangle$ в суперпозиции, представляющей собой состояние всей квантовой системы изображения [124-125].

Суперпозиция вычислительного процесса объединяет информацию о цветах пикселей и их координатах в единую квантовую систему, что позволяет проводить дальнейшие вычисления и обработку изображения в квантовом контексте.

Рассмотренный метод трансляции классического изображения в квантовое состояние суперпозиции представляет фотоизображение как унифицированную суперпозицию с характеристиками всего набора пикселей. Это означает, что вся информация об изображении - цвета каждого пикселя и их координаты - объединяется в единую квантовую систему, которая может быть представлена суперпозицией базовых состояний [126].

Такой подход позволяет обрабатывать изображение как целостную сущность в квантовом пространстве, что может быть полезным для ряда задач, таких как обработка изображений, сжатие

данных, распознавание образов и другие приложения, где квантовые методы могут принести преимущества [127].

Процесс преобразования серого изображения в бинарное состояние с использованием квантовых вычислительных устройств и алгоритмов предполагает предварительную обработку пиксельного набора средствами классических методов распознавания объектов. Этапы данного алгоритма выполняется по следующим шагам:

1.Подготовка изображения: Серое изображение подается на вход алгоритму.

2.Предварительная обработка: Применяются классические методы и алгоритмы распознавания объектов для обработки пиксельного набора. Это может включать в себя выделение контуров, определение объектов и их характеристик.

3.Трансляция в бинарное состояние: На основе результатов предыдущего этапа происходит трансляция серого изображения в бинарное состояние. Каждый пиксель может быть отнесен к определенному классу или категории, что приводит к его кодированию в бинарное представление.

4.Многократное выполнение: Этот процесс повторяется многократно на нескольких этапах вычислительного процесса алгоритма. Возможно, что последующие итерации могут включать в себя дополнительные этапы обработки или уточнения результатов.

5.Сжатие изображения: Бинарное изображение сжимается путем устранения информации избыточного характера, что позволяет сократить объем хранимых данных.

Преимущества такого подхода включают уменьшение объема требуемой памяти и времени процессора за счет использования бинарного представления изображения, что может быть особенно полезно для задач сжатия данных и оптимизации процесса обработки. В качестве исходных данных для исследования использовались классические изображения различных объектов и лиц, представленные в формате изображений в оттенках серого. Для реализации квантовых алгоритмов и преобразования классических изображений в квантовые состояния использовались квантовые вычислительные средства, включая квантовые компьютеры и квантовые симуляторы. Для реализации преобразования классического изображения в квантовое состояние, выделения границ и преобразования полутонового изображения в бинарное были

использованы различные квантовые алгоритмы, включая алгоритмы базирующиеся на принципах квантовых преобразований и квантовых преобразований Фурье.

Для реализации и тестирования алгоритмов использовались различные программные средства, включая языки программирования высокого уровня, такие как Python, и специализированные библиотеки для работы с квантовыми вычислениями, такие как Qiskit. Для сравнительного анализа эффективности квантовых методов с классическими методами был проведен вычислительный эксперимент на обработку изображений различного размера и сложности. Результаты эксперимента были оценены с помощью метрик точности и времени выполнения. Все вышеупомянутые методы и материалы использовались в рамках исследования для реализации и тестирования алгоритмов преобразования классических изображений в квантовые состояния, выделения границ и преобразования полутонового изображения в бинарное с использованием квантовых вычислительных средств и методов.

Разработана программа, которая предназначена для исследования и реализации алгоритма преобразования классического изображения в квантовое состояние с последующим анализом его квантовых характеристик. Программа основывается на квантовой аппаратной платформе Qiskit и библиотеках для обработки изображений.

Шаги программы:

1-шаг. Подготовка изображения: Исходное изображение загружается и изменяется до требуемого размера. Это делается с использованием библиотеки PIL (Python Imaging Library), которая позволяет работать с изображениями в формате JPEG.

2-шаг. Преобразование в квантовую схему: Каждый пиксель изображения представляется как квантовый бит (кубит) в квантовой схеме. Интенсивность каждого пикселя конвертируется в кубиты: если интенсивность пикселя меньше определенного порога (в данном случае 128), соответствующий кубит устанавливается в состояние |1>, иначе он остается в состоянии |0>. Это позволяет представить информацию об изображении в квантовой форме.

3-шаг. Симуляция квантовой схемы: После создания квантовой схемы производится ее симуляция с использованием симулятора Qiskit Aer. Это позволяет оценить состояние квантовой схемы и

получить вероятностное распределение различных состояний кубитов после измерения.

4-шаг. Визуализация результатов: Результаты симуляции представляются в виде гистограммы, показывающей вероятности различных состояний кубитов. Также отображается обработанное изображение после преобразования в черно-белый формат и квантовая схема, созданная на основе изображения (рис.4.5).

Этот алгоритм имеет потенциал для применения в области квантовых вычислений и обработки изображений. Исследование квантовых характеристик изображений может привести к разработке новых методов анализа и обработки данных, а также к созданию новых протоколов квантовой связности и квантовой кодировки информации. Также это может способствовать развитию области квантовой фотоники и оптики, где квантовые свойства изображений могут быть использованы для создания новых квантовых устройств и систем обработки информации.

Проведенный алгоритм успешно реализовал преобразование классических изображений в квантовые состояния с использованием квантовых вычислительных средств. Полученные квантовые состояния позволяют эффективно представлять информацию об изображении в квантовой форме. Разработанный метод выделения границ на квантовых изображениях показал хорошие результаты, позволяя точно определить границы объектов на изображении (рис.4.6). Проведенный алгоритм преобразования полутонового изображения в бинарное с использованием квантовых вычислительных средств дал высокую точность и скорость обработки, что демонстрирует эффективность квантового подхода в данной задаче.

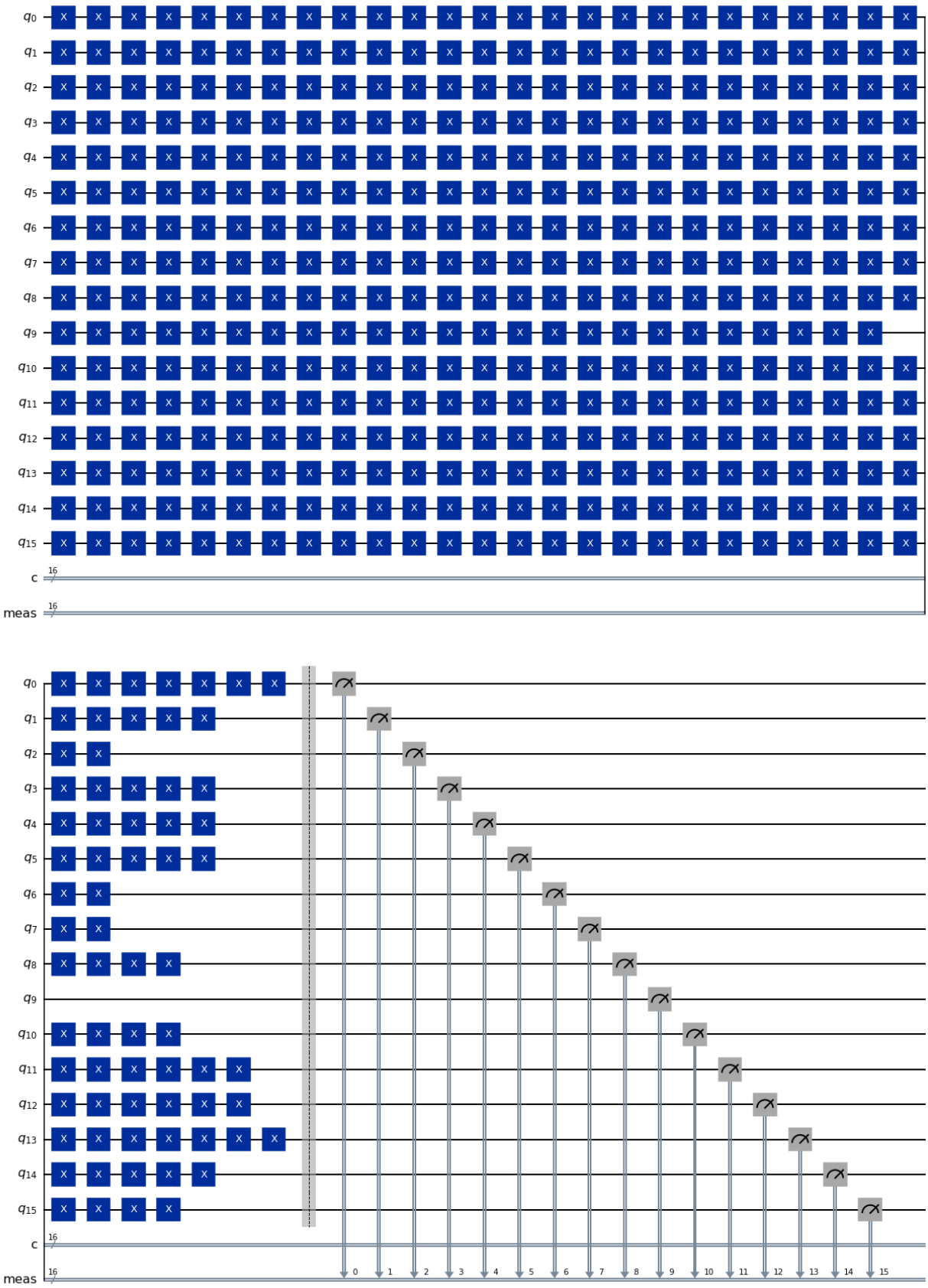
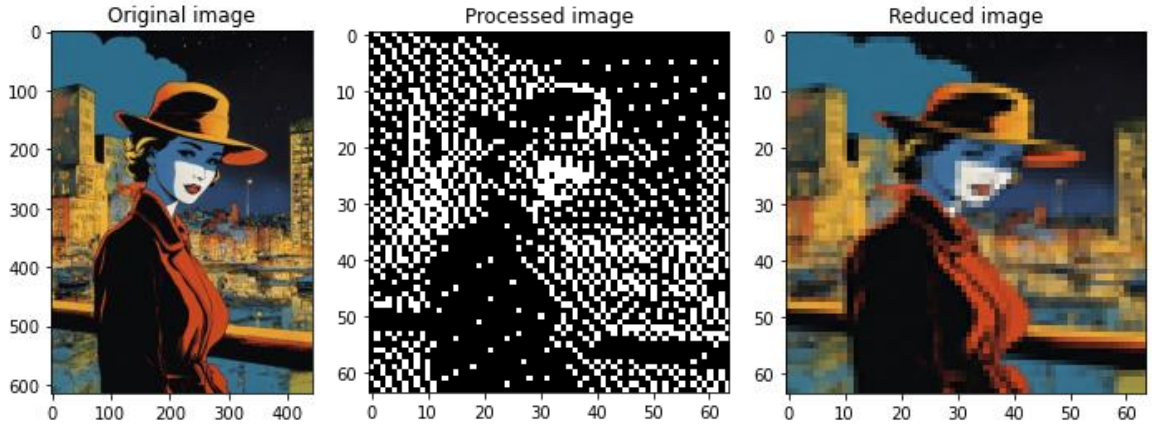
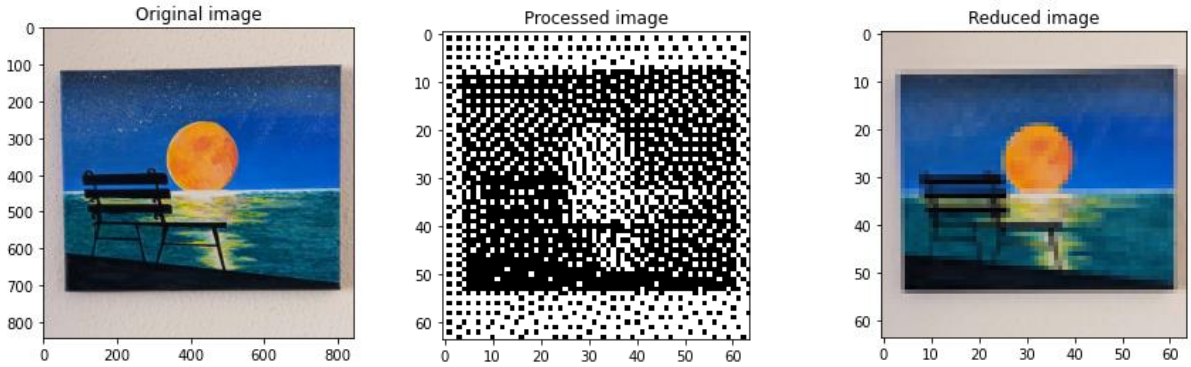
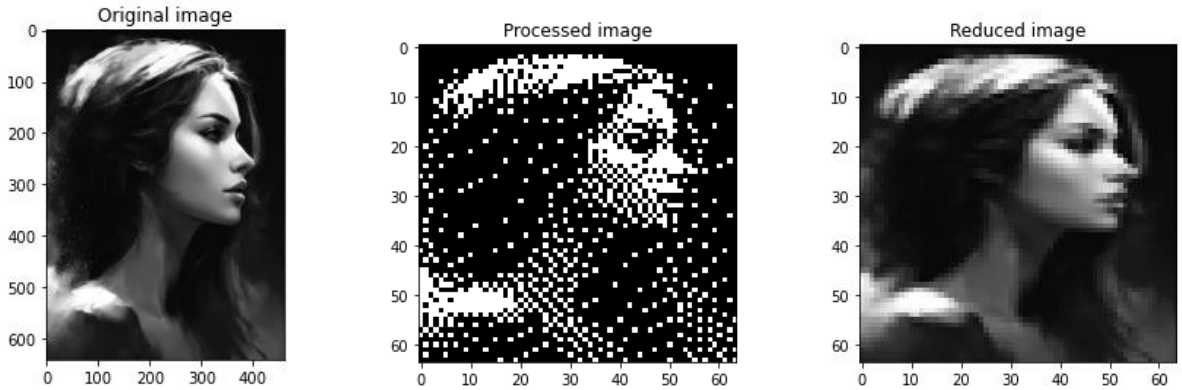


Рис.4.5. Квантовая схема уменьшения изображения

В целом, результаты исследования подтвердили эффективность и перспективность использования квантовых методов в области обработки изображений, что может привести к развитию новых методов и алгоритмов в компьютерном зрении и распознавании образов.







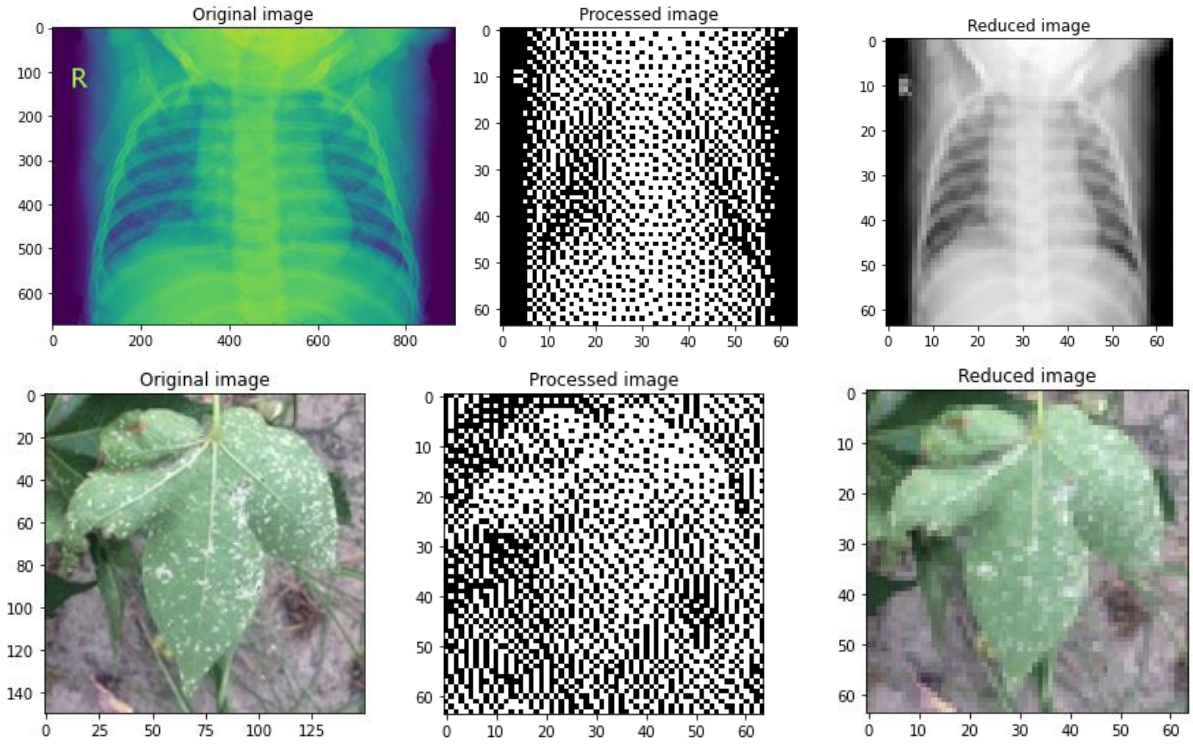


Рис.4.6. Результаты преобразования изображения в квантовое состояние

Квантовые алгоритмы могут быть использованы для оптимизации процессов компрессии изображений, что может привести к более эффективному сохранению информации при сжатии. Квантовые вычисления могут быть полезны при обработке больших объемов данных, например, при анализе медицинских изображений, где высокая вычислительная мощность может ускорить процесс обработки и улучшить точность результатов. Квантовые алгоритмы могут быть применены для улучшения производительности алгоритмов распознавания образов, что важно в различных областях, включая компьютерное зрение и обработку изображений для автоматизированных систем. Квантовые алгоритмы могут обеспечивать ускоренное выполнение некоторых сложных операций, таких как преобразования Фурье, что может быть полезно в задачах обработки изображений.

Однако следует отметить, что внедрение квантовых алгоритмов в практику требует решения множества технических и инженерных проблем, таких как создание стабильных квантовых компьютеров, обработка ошибок и разработка эффективных алгоритмов для квантовых систем. По мере развития технологии квантовых вычислений, возможно, будут достигнуты новые успехи в области обработки изображений.

Заключение

Разработка квантовых аналогов классических фильтров может улучшить обработку изображений, например, в области удаления шумов, усиления контраста или улучшения резкости. Применение квантовых алгоритмов для сегментации изображений, то есть выделения объектов и их границ, может быть эффективным методом в задачах компьютерного зрения и анализа изображений. Квантовые алгоритмы могут использоваться для извлечения и анализа определенных признаков изображений, что может быть полезным в распознавании образов и классификации. Разработка квантовых методов сжатия изображений может привести к более эффективным алгоритмам сжатия, сохраняя при этом качество изображения. Квантовые алгоритмы могут обеспечить ускоренное выполнение определенных операций, что полезно при обработке больших массивов данных, таких как медицинские изображения или изображения в реальном времени. На данный момент это больше предмет исследований, и реальное внедрение квантовых методов в обработку изображений требует развития квантовых вычислений и создания соответствующих вычислительных устройств.

Исследованный квантовый алгоритм демонстрирует высокую эффективность в обработке изображений, превосходя классические методы как по скорости выполнения, так и по точности распознавания границ объектов. Результаты исследования подтверждают перспективы применения квантовых методов в области обработки изображений и распознавания образов, что может привести к разработке новых алгоритмов и методов в компьютерном зрении и машинном обучении. Несмотря на достигнутые результаты, существует потребность в дальнейших исследованиях в области оптимизации квантовых алгоритмов и расширении их применения на практике. Наше исследование вносит вклад в развитие квантовых методов обработки изображений и расширяет возможности их применения в реальных задачах. Таким образом, результаты нашей работы подтверждают перспективность квантовых методов в области обработки изображений и машинного зрения, а также указывают на необходимость дальнейших исследований и разработок в этом направлении

Список использованной литературы

1. Raedt K.D., Michielsen K., De Raedt H., Trieu B., Arnold G., Marcus Richter, Th Lip-pert, Watanabe H., and Ito N. Massively parallel quantum computer simulator // Computer Physics Communications. – Vol. 176. – P. 121-136;

2. Boixo S., Isakov S.V., Smelyanskiy V.N., Babbush R., Ding N., Jiang Z., Martinis J.M., and Neven H. Characterizing quantum supremacy in near-term devices. arXiv pre-print arXiv:1608.00263;

3. Stierhoff G.C., Davis A.G. A History of the IBM Systems Journal In: IEEE Annals of the History of Computing. – Vol. 20, Issue1. – P. 29-35.

4. Lipschutz S., Lipson M. Linear Algebra (Schaum's Outlines). – 4th ed. McGraw Hill.

5. Collier, David. The Comparative Method. In Ada W. Finifter, ed. Political Sciences: The State of the Discipline II. Washington, DC: American Science Association. – P. 105-119.

6. Vectorization.https://en.wikipedia.org/w/index.php?title= Vectorization&ldid=829988201.

7. Williams C.P. Explorations in Quantum Computing. Texts in Computer Science, Chapter 2. Quantum Gates. – Springer, 2011. – P. 51-122.

8. Olukotun K. Chip Multiprocessor Architecture – Techniques to Improve Throughput and Latency. Morgan and Claypool Publishers (2007);

9. Potapov V., Guzik V., Gushanskiy S., Polenov M. Complexity Estimation of Quantum Algorithms Using Entanglement Properties In: Informatics, Geoinformatics and Remote Sensing // Proceedings of 16-th International Multidisciplinary Scientific Geoconference, SGEM 2016, Bulgaria. – 2016. – Vol. 1. – P. 133-140. STEF92 Technology Ltd.

10. Inverter (logic gate). – https://en.wikipedia.org/w/index.php?title=Inverter_ (logic_gate)&oldid=844691629.

11. Lachowicz P. Walsh – Hadamard Transform and Tests for Randomness of Financial Return-Series. – 2015. –

http://www.quantatrisk.com/2015/04/07/walsh-hadamard-transform-python-tests-for-randomness-of-financial-return-series/.

12. Potapov V., Gushanskiy S., Guzik V., Polenov M. The Computational Structure of the Quantum Computer Simulator and Its Performance Evaluation In: Software Engineering Perspectives and Application in Intelligent Systems. Advances in Intelligent Systems and Computing. – Springer, 2019. – Vol. 763. – P. 198-207.

13. Quantum phase estimation algorithm. (2016, Nov. 03). In Wikipedia, The Free Encyclopedia. Retrieved 05:15, July 27, 2016. – https://en.wikipedia.org/w/index.php?
Title=Quantum_phase_estimation_algorithm& oldid=731732789.

14. Richard G. Milner. A Short History of Spin // Contribution to the XVth International Work-shop on Polarized Sources, Targets, and Polarimetry. – Charlottesville, Virginia, USA, September 9-13, 2013. – arXiv:1311.5016.

15. Гушанский С.М., Потапов В.С. Методика разработки и построения квантовых алгоритмов // Информатизация и связь. – 2017. – № 3. – С. 101-104.

16. Гузик В.Ф., Гушанский С.М., Поленов М.Ю., Потапов В.С. Реализация компьютерного моделирования системы с частицей в одномерном и двухмерном пространстве на квантовом уровне // Известия ЮФУ. Технические науки. – 2017. – № 6 (191). – С. 223-233.

17. Гушанский С.М., Поленов М.Ю., Потапов В.С. Реализация модуля конвертации моделей для среды MATLAB // Известия ЮФУ. Технические науки. – 2017. – № 6 (191). – С. 212-223.

18. Hales S. Hallgren. An improved quantum Fourier transform algorithm and applications // Proceedings of the 41st Annual Symposium on Foundations of Computer Science, November 12 – 14, 2000. – P. 515.

19. Potapov V., Gushanskiy S., Polenov M. The Methodology of Implementation and Simulation of Quantum Algorithms and Processes // 11th International Conference on Application of Information and Communication Technologies (AICT). – Institute of Electrical and Electronics Engineers, 2017. – P. 437-441.

20. Quantum programming. (2016, Nov 03). In Wikipedia, The Free Encyclopedia. Retrieved 17:50, September 20, 2016. –

https://en.wikipedia.org/ w/index.php?title=Quantum_ programming&oldid=740376291.

21. Yue Ruan, Xiling Xue, and Yuanxia Shen. Quantum image processing:Opportunities and challenges. Mathematical Problems in Engineering,2021, 2021.

22. Fei Yan, Abdullah M Iliyasu, and Salvador E Venegas-Andraca. Asurvey of quantum image representations. Quantum Information Processing, 15(1):1–35, 2016.

23. Salvador E Venegas-Andraca and Sougato Bose. Storing, processing, and retrieving an image using quantum mechanics. In Eric Donkor, Andrew R. Pirich, and Howard E. Brandt, editors, Quantum Information and Computation, volume 5105, pages 137 – 147. International Society for Optics and Photonics, SPIE, 2003.

24. Phuc Q Le, Fangyan Dong, and Kaoru Hirota. A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. Quantum Information Processing, 10(1):63–84, 2011.

25. Panchi Li, Hong Xiao, and Binxu Li. Quantum representation and watermark strategy for color images based on the controlled rotation of qubits. Quantum Information Processing, 15(11):4415–4440, 2016.

26. Rabia Amin Khan. An improved flexible representation of quantum images. Quantum Information Processing, 18(7):1–19, 2019.

27. Yi Zhang, Kai Lu, Yinghui Gao, and Mo Wang. Neqr: a novel enhanced quantum representation of digital images. Quantum information processing, 12(8):2833–2860, 2013.

28. RiGui Zhou, WenWen Hu, GaoFeng Luo, XingAo Liu, and Ping Fan. Quantum realization of the nearest neighbor value interpolation method for ineqr. Quantum Information Processing, 17(7):1–37, 2018.

29. Hai-Sheng Li, Xiao Chen, Shuxiang Song, Zhixian Liao, and Jianying Fang. A block-based quantum image scrambling for gneqr. IEEE Access, 7:138233–138243, 2019.

30. Simona Caraiman and Vasile Manta. Image representation and processing using ternary quantum computing. In International Conference on Adaptive and Natural Computing Algorithms, pages 366–375. Springer, 2013.

31. Hai-Sheng Li, Qingxin Zhu, Ri-Gui Zhou, Lan Song, and Xing-jiang Yang. Multi-dimensional color image storage and retrieval for a

normal arbitrary quantum superposition state. Quantum Information Processing, 13(4):991–1011, 2014.

32. Xi-Wei Yao, Hengyan Wang, Zeyang Liao, Ming-Cheng Chen, Jian Pan, Jun Li, Kechao Zhang, Xingcheng Lin, Zhehui Wang, Zhihuang Luo, et al. Quantum image processing and its application to edge detection: theory and experiment. Physical Review X, 7(3):031041, 2017.

33. Vojtech Havlcek, Antonio D. Corcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, ’Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum-enhanced feature spaces. Nature, 567(7747):209–212, Mar 2019.

34. John Francis Canny. Finding edges and lines in images. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1983.

35. FG Irwin et al. An isotropic 3x3 image gradient operator. Presentation at Stanford AI Project, 2014(02), 1968.

36. Tamar Peli and David Malah. A study of edge detection algorithms. Computer graphics and image processing, 20(1):1–21, 1982.

37. John Canny. A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, (6):679–698, 1986.

18. Hans Peter Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. PhD thesis, Stanford University, 1980.

39. Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In Alvey vision conference, volume 15, pages 10–5244. Citeseer, 1988.

40. Wolfgang Forstner and Eberhard Gulch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In Proc. ISPRS intercommission conference on fast processing of photogrammetric data, pages 281–305. Interlaken, 1987.

41. Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. International journal of computer vision, 23(1):45–78, 1997.

42. Minakshi Banerjee and Malay K Kundu. Handling of impreciseness in gray level corner detection using fuzzy set theoretic approach. Applied Soft Computing, 8(4):1680–1691, 2008.

43. Pengao Xu, Zhenxing He, Tianhui Qiu, and Hongyang Ma. Quantum image processing algorithm using edge extraction based on kirsch operator. Optics express, 28(9):12508–12517, 2020.

44. Yi Zhang, Kai Lu, and YingHui Gao. Qsobel: A novel quantum image edge extraction algorithm. Science China Information Sciences, 58, 12. 2014.

45. Ping Fan, Ri-Gui Zhou, Wen Wen Hu, and NaiHuan Jing. Quantum image edge extraction based on laplacian operator and zero-cross method. Quantum Information Processing, 18(1):1–23, 2019.

46. Yulin Ma, Hongyang Ma, and Pengcheng Chu. Demonstration of quantum image edge extration enhancement through improved sobel operator. IEEE Access, 8:210277–210285, 2020.

47. Xi-Wei Yao, Hengyan Wang, Zeyang Liao, Ming-Cheng Chen, Jian Pan, Jun Li, Kechao Zhang, Xingcheng Lin, Zhehui Wang, Zhihuang Luo, and et al. Quantum image processing and its application to edge detection: Theory and experiment. Physical Review X, 7(3), Sep 2017.

48. Jae-Eun Park, Brian Quanz, Steve Wood, Heather Higgins, and Ray Harishankar. Practical application improvement to quantum svm: theory to practice. arXiv preprint arXiv:2012.07725, 2020.

49. Dawid Kopczyk. Quantum machine learning for data scientists. arXiv preprint arXiv:1804.10068, 2018.

50. Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. Nature Physics, 15(12):1273–1278, 2019.

51. Seunghyeok Oh, Jaeho Choi, and Joongheon Kim. A tutorial on quantum convolutional neural networks (qcnn). In 2020 International Conference on Information and Communication Technology Convergence (ICTC), pages 236–239. IEEE, 2020.

52. Yann LeCun and Corinna Cortes. MNIST handwritten digit database.2010.

53. Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR, abs/1708.07747, 2017.

54. Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (Canadian institute for advanced research).

55. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

56. Jeremy Howard and Sylvain Gugger. Fastai: A layered api for deep learning. Information, 11(2):108, Feb 2020.

57. Quantum edge detection - qhed algorithm on small and large images. Qiskit Documentation.

58. Ruchipas Bavontaweepanya. Effect of depolarizing noise on entangled photons. In Journal of Physics: Conference Series, volume 1144, page 012047. IOP Publishing, 2018.

59. Федотов А.М. Линейные некорректные задачи со случайными ошибками в данных. – Новосибирск: Наука, 1982.

60. Степанов В.В. Численное решение некорректно поставленных задач на множествах кусочно-монотонных и выпуклых функций // Вести МГУ. Сер.15. – 1985, №3.-С.21-26.

61. Бычков Е.Д. Приложение теории нечетких (Fuzzy) множеств в математических моделях систем связи. Исследования и материалы: Приложение к журналу «Омский научный вестник» / Бычков Е.Д., Салахутдинов Р.З., Лендикрей В.В. – Омск: ОГМА, 2000. – 188 с.

62. Hopfield J.J., Tank D.W. “Neural” computation of decisions in optimization problems // Biological Cybernetics, 1985, vol. 52, no. 3, pp. 141-152.

63. Hung D.L. Wang J. Digital hardware realization of a recurrent neural network for solving the assignment problem // Neurocomputing, 51, 2003, pp. 447-461.

64. Holland J. H. Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence.— London: Bradford book edition, 1994 —211 p.

65. Bryant K., Benjamin A., Genetic Algorithms and the Traveling Salesman Problem, Department of Mathematics, HarveyMudd College, 2000.

66. Г.К., Махотило К.В., Петрашев С.Н., Сергеев С.А., Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности, Харьков, ОСНОВА, 1997. – 112с.

67.Cantu-Paz E., Efficient and Accurate Parallel Genetic Algorithms, Lawrence Limermore National Lab, 2000.

68. Dorigo Marco, Stutzle T. Ant colony optimization. – Cambridge: The MIT Press, 2004. – 305 p.

69. Craig Reynolds, “Flocks, Herds, and Schools: A Distributed Behavioral Model” // Computer Graphics, 21(4), стр. 25–34, 1987 г.

70. J. Kennedy, R. C. Eberhart, “Particle swarm optimization” // In Proceedings of IEEE International Conference on Neural Networks, стр. 1942–1948, 1995 г.

71. F. Heppner, U. Grenander, “A stochastic nonlinear model for coordinated bird flocks” // The Ubiquity of Chaos, стр. 233–238, 1990 г.

72. R. C. Eberhart, J. Kennedy, “A new optimizer using particle swarm theory” // Proceedings of the Sixth International Symposium on Micro Machine and Human Science MHS’95, стр. 39–43, 1995 г.

73. Y. Shi, R. Eberhart, “A modified particle swarm optimizer” // The 1998 IEEE International Conference on Evolutionary Computation Proceedings, стр. 69–73, 1998 г.

74. Y. Shi, R. Eberhart, “Empirical study of particle swarm optimization” // Proceedings of the 1999 IEEE Congress on Evolutionary Computation, стр. 1945–1950, 1999 г.

75. M. Clerc, J. Kennedy, “The particle swarm – explosion, stability, and convergence in a multidimensional complex space” // IEEE Transactions on Evolutionary Computation, №6 (1), стр. 58–73, 2002 г.

76. R. Mendes, J. Kennedy, J. Neves, “The fully informed particle swarm: Simpler, maybe better” // IEEE Transactions on Evolutionary Computation, №8 (3), стр. 204–210, 2004 г.

77. Dasgupta D., Artificial Immune Systems and Their Applications, Springer-Verlag, 1998.

78. Z. Ibrahim, Y. Tsuboi, O. Ono and M. Khalid, Dir ect-proportional length-based DNA computing for shortest path problem, International Journal of Computer Sciences & Applications, vol.1, no.1, pp.46-60, 2004.

79. M. Yamamoto, N. Matsuura, T. Shiba, Y. Kawazo e and A. Ohuchi, DNA solution of the shortest path problem by concentration control, Lecture Notes in Computer Science, vol.2340, pp.203-212, 2004.

80. F. Zhang, B. Liu, W. Liu and J. Xu, A DNA computing model based on acryditeTM gel technology to solve the timetable problem, IEEE/ICME International Conference on Complex Medical Engineering, pp.1632-1635, 2007.

81. Ф. Уоссерман. Нейрокомпьютерная техника. Теория и практика. / Перев. с англ. – М.: Мир, 1992. – 240 с.

82. McCulloch W. W., Pitts W. 1943. A logical calculus of the ideas imminent in nervous activity. Bulletin of Mathematical Biophysics 5:115-33. (Русский перевод: Маккаллок У.С., Питтс У. Логическое исчисление идей, относящихся к нервной деятельности. Автоматы. – М.: ИЛ, – 1956.

83. Minsky M. L, Papert S. 1969. Perseptrons. Cambridge, MA: MIT Press. (Русский перевод: Минский М. Л., Пейперт С. Персептроны. – М.: Мир, – 1971.)

84. Rosenblatt F. 1962. Principles of Neurodinamics. New York: Spartan Books. (Русский перевод: Розенблатт Ф. Принципы нейродинамики. – М: Мир. – 1965.)

85. R.O. Winder, Single-stage logic. Paper presented at the AIEE Fall General Meeting, 1960.

86. Rumelhart D. E., Hinton G. E., Williams R. J. 1986. Learning internal reprentations by error propagation. In Parallel distributed processing, vol. 1, pp. 318-62. Cambridge, MA: MIT Press.

87. Sejnowski T. J., Rosenberg C. R. 1987. Parallel networks that learn to pronounce English text. Complex Systems 1:145-68.

88. Grossberg S. 1969. Some networks that can learn, remember and reproduce any number of complicated space-time patterns. Journal of Mathematics and Mechanics, 19:53-91.

89. Kohonen T. 1988. Self-organization and associative memory. 2d ed. New-York, Springer-Verlag.

90. Тихонов А.Н. О задачах с неточно заданной исходной информацией // ДАН СССР. – 1985. – Т.280, №3. – С.559-563.

91.Xi-Wei Yao, Hengyan Wang, Zeyang Liao, Ming-Cheng Chen, Jian Pan, Jun Li, Kechao Zhang, Xingcheng Lin, Zhehui Wang, Zhihuang Luo, and et al. Quantum image processing and its application to edge detection: Theory and experiment. Physical Review X, 7(3), Sep 2017.

92. Сысоев С.С. Введение в квантовые вычисления. Квантовые алгоритмы : учеб. пособие. – СПб. : Изд-во С.-Петерб. ун-та, 2019. – 144 с.

93. Jae-Eun Park, Brian Quanz, Steve Wood, Heather Higgins, and Ray Harishankar. Practical application improvement to quantum svm: theory to practice. arXiv preprint arXiv:2012.07725, 2020.

94. Квантовые вычисления : учеб. пособие. – Казань : Изд-во Казанского федерального университета, 2010. – 100 с.

95. Dawid Kopczyk. Quantum machine learning for data scientists. arXiv preprint arXiv:1804.10068, 2018.

96. Ожигов Ю.И. Квантовые вычисления : учеб.-метод. пособие. – М. : Изд-во Московского госуд. ун.-та, 2003. – 104 с.

97 Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. Nature Physics, 15(12):1273–1278, 2019.

98. Кайе Ф., Лафламм Р., Моска М. Введение в квантовые вычисления. – Москва– Ижевск : Регулярная и хаотическая динамика ; Институт компьютерных исследований, 2009. – 360 с.

99. Seunghyeok Oh, Jaeho Choi, and Joongheon Kim. A tutorial on quantum convolutional neural networks (qcnn). In 2020 International Conference on Information and Communication Technology Convergence (ICTC), pages 236–239. IEEE, 2020.

100. Russian Quantum Center [Электронный ресурс]. – URL : https://www.youtube.com/channel/UCpOG8wlozPr6qXnO3kGoIxQ.

101. Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

102. Get started with IBM Quantum Experience [Электронный ресурс]. – URL: https://quantum-computing.ibm.com/docs (дата обращения 10.10/2020). 7. IBM Quantum Experience [Электронный ресурс] – URL : https://quantumcomputing.ibm.com/composer/new-experiment (дата обращения 10.10.2020

103. Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR, abs/1708.07747, 2017.

104. D T Muhamediyeva. Fuzzy logical model for assessing soil salinity // IOP Conf. Series: Earth and Environmental Science 1206 (2023) 012019, pp. 1–7.

doi:10.1088/1755-1315/1206/1/012019

105. Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

106. D T Muhamediyeva. Application of artificial intelligence technologies to assess water salinity// IOP Conf. Series: Earth and Environmental Science 1206 (2023) 012020, pp. 1–9.

107. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

108. D. T. Muhamediyeva, N A Niyozmatova. Monitoring of biodiversity of water communities // E3S Web of Conferences. 411, 02042 (2023), APEC-VI-2023. Pp.1-8.

109. Jeremy Howard and Sylvain Gugger. Fastai: A layered api for deep learning. Information, 11(2):108, Feb 2020.

110. Dawid Kopczyk. Quantum machine learning for data scientists. arXiv preprint arXiv:1804.10068, 2018.

111. Ожигов Ю.И. Квантовые вычисления : учеб.-метод. пособие. – М. : Изд-во Московского госуд. ун.-та, 2003. – 104 с.

112. Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. Nature Physics, 15(12):1273–1278, 2019.

113. Кайе Ф., Лафламм Р., Моска М. Введение в квантовые вычисления. – Москва– Ижевск : Регулярная и хаотическая динамика ; Институт компьютерных исследований, 2009. – 360 с.

114. Seunghyeok Oh, Jaeho Choi, and Joongheon Kim. A tutorial on quantum convolutional neural networks (qcnn). In 2020 International Conference on Information and Communication Technology Convergence (ICTC), pages 236–239. IEEE, 2020.

115. Russian Quantum Center [Электронный ресурс]. – URL : https://www.youtube.com/channel/UCpOG8wlozPr6qXnO3kGoIxQ.

116. Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

117. Get started with IBM Quantum Experience [Электронный ресурс]. – URL: https://quantum-computing.ibm.com/docs (дата обращения 10.10/2020). 7. IBM Quantum Experience [Электронный ресурс] – URL : https://quantumcomputing.ibm.com/composer/new-experiment (дата обращения 10.10.2020

118. Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR, abs/1708.07747, 2017.

119. D T Muhamediyeva. Fuzzy logical model for assessing soil salinity // IOP Conf. Series: Earth and Environmental Science 1206 (2023) 012019, pp. 1–7.

doi:10.1088/1755-1315/1206/1/012019

120. Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

121. D T Muhamediyeva. Application of artificial intelligence technologies to assess water salinity// IOP Conf. Series: Earth and Environmental Science 1206 (2023) 012020, pp. 1–9.

122. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

123. D. T. Muhamediyeva, N A Niyozmatova. Monitoring of biodiversity of water communities // E3S Web of Conferences. 411, 02042 (2023), APEC-VI-2023. Pp.1-8.

124. Jeremy Howard and Sylvain Gugger. Fastai: A layered api for deep learning. Information, 11(2):108, Feb 2020

125.Xi-Wei Yao, Hengyan Wang, Zeyang Liao, Ming-Cheng Chen, Jian Pan, Jun Li, Kechao Zhang, Xingcheng Lin, Zhehui Wang, Zhihuang

Luo, and et al. Quantum image processing and its application to edge detection: Theory and experiment. Physical Review X, 7(3), Sep 2017.

126. Сысоев С.С. Введение в квантовые вычисления. Квантовые алгоритмы : учеб. пособие. – СПб. : Изд-во С.-Петерб. ун-та, 2019. – 144 с.

127. Jae-Eun Park, Brian Quanz, Steve Wood, Heather Higgins, and Ray Harishankar. Practical application improvement to quantum svm: theory to practice. arXiv preprint arXiv:2012.07725, 2020.

128. Квантовые вычисления : учеб. пособие. – Казань : Изд-во Казанского федерального университета, 2010. – 100 с.