Московский Государственный институт Электроники и Математики
(технический университет)

кафедра МОСОИиУ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломной работе

по специальности 230201 «Информационные системы и технологии»

на тему: «Разработка и анализ эффективности детектора контуров в графических изображениях на основе нейросетевой технологии»

Студент:	Милевски А.Е.		
Руководитель:	Истратов А.Ю.	доцент, к.т.н.	...
Консультант:	Путилин Д.С.	инженер	...

допущен к защите 10 июня 2008 года

Зав. кафедрой _____

Москва

Аннотация

Оглавление

Введение

Определение контуров в графических изображениях играет важную роль в обработке изображений и компьютерном зрении. Особенное значение эта задача принамает в рамках выявления и выделения объектов в изображениях, где особенно востребованы алгоритмы, определяющие точки, явно выделяющиеся по яркости, или же, говоря более формально, выделяющиеся по интересующим свойствам в обрабатываемом изображении.

Цели

Цель выявления значительных различий в яркости или освещённости в изображениях состоит в том, чтобы зафиксировать важные явления и изменения свойств в представленной на изображении среде. В модели представления изображения разрывы монотонности яркости чаще всего могут соответствовать

- разнице в глубине
- разнице в угле поворота поверхностей
- различиям в фактуре и материале поверхностей
- разнице в освещённости среды

В идеальном случае, обработка изображения контурным детектором даёт набор кривых, представляющих границы объектов, очертания поверхностей или же кривые, соответствующие пересечениям, преломлениям или изменениям ориентации поверхностей. Применение контурных детекторов также уменьшает объём информации, предназначенной для обработки, отсеивая менее значительную информацию и сохраняя наиболее важные структурные особенности изображения. Успешное выделение контуров может существенно упростить следующие за ним операции распознавания и интерпретации. К сожалению, не всегда возможно получить отвечающую потребностям карту контуров даже из средней сложности изображений реальной среды. Контуры, полученные из нетривиальных изображений, зачастую нарушены ложной фрагментацией, состоящей в разрывах кривых контуров, пропусках участков контуров, а также ошибочным выделением контуров объектов, не касающихся поставленных задач, что может усложнить последующие действия по интерпретации содержащейся в изображении информации.

Свойства границ

Контуры, выделенные на двухмерном изображении трехмерной среды, можно разделить на зависящие и не зависящие от точки зрения. Не зависящие от точки зрения контуры обычно представляют различия в собственных свойствах трёхмерных объектов, таких как форма или поверхность. Контуры, зависящие от точки обзора, могут изменяться вместе с точкой зрения и обычно представляют свойства среды и взаимное расположение объектов.

Контур, к примеру, может представлять границу между областями разного цвета. В этом усматривается его отличие от кривой или линии, которые, будучи выделенной детектором

кривых, являются ограниченным набором пикселов цвета отличающегося от более или менее постоянного цвета фона. Таким образом, кривая будет очерчена контурным детектором со всех сторон, где будет зарегистрировано различие в цвете. Контуры играют особенно важную роль в обработке изображений во многих приложениях, в частности в системах машинного зрения, которые анализируют поведение объектов при контролируемыех условиях освещения. Ввиду сложности этой задачи, в последние годы ведётся разработка в направлении методов функционирования машинного зрения, не основанных исключительно на выявлении контуров как предварительной стадии обработки изображения.

Обобщённая модель границы

Хотя этапу выявления контуров посвящено множество разработок, карты контуров, полученные из изображений естественной среды, зачастую являются далеко не идеально подходящими для дальнейшей обработки. Напротив, часто они искажены следующими явлениями:

- оптическое размытие по причине ограниченных фокусного расстояния и угла обзора
- размытие полутеней от источников света с ненулевым радиусом распространения лучей
- затенение на сглаженных границах формы объектов
- локальные рефлексы или взаимные отражения соседних объектов в районе их границ

Хотя данная модель не отражает всего многообразия контуров в реальной среде, функция ошибок erf применяется множеством исследователей в прикладных задачах для учёта размытия контуров, как простейшее приближение ([12],[13]). Одномерное изображение f с одиночным контуром в точке $x = 0$ моделируется следующей функцией:

$$f(x) = \frac{I_r - I_l}{2}\left(\operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + 1\right) + I_l \tag{0.1}$$

Слева от контура яркость равна $I_l = \lim\limits_{x\to-\infty} f(x)$, а справа от контура $I_r = \lim\limits_{x\to\infty} f(x)$. Коэффициент σ называется коэффициентом размытия контура.

Нетривиальность задачи выявления границ

Чтобы наглядно показать, почему задача выявления контура является нетривиальной, рассмотрим задачу выявления контура в векторе уровней яркости $[5, 7, 6, 4, 152, 148, 149]$. Интуитивно можно сказать, что контур находится между 4-м и 5-м пикселом. Точное определение порога различия яркости между соседними пикселами, предполагающего наличие контура между ними, не всегда является простой задачей. В сущности, это одна из причин, по которым выявление контуров является нетривиальной задачей в случае, если рассматриваемая среда не примитивна и нет возможности контролировать условия освещения.

Глава 1.

Обзор существующих детекторов контуров в графических изображениях

1.1. Традиционные подходы к выявлению границ

Существует множество подходов к выявлению контуров, но большинство из них можно разделить на две категории: поисковые и методы нулевых пересечений. Поисковые методы основаны на вычислении величины контура при помощи дифференциальных уравнений первого порядка, таких как операторы вычисления векторов градиентов, которые позволяют оценить направление контура. Методы нулевых пересечений осуществляют поиск нулевых прерсечений во второй производной функции сигнала изображения, обычно используются оператор Лапласа или нелинейные дифференциальные уравнения. Почти во всех случаях, перед применением этих методов изображение для подавления шумов подвергается размытию, обычно по Гауссу.

Большинство известных методов различаются именно применяемыми фильтрами размытия и способами оценки силы контура. Так как многие методы основаны на вычислении векторов градиентов, они могут отличаться типами фильтров, применяемых для вычисления градиентов горизонтальных и вертикальных контуров.

1.2. Операторы на основе первой производной

1.2.1. Оператор Робертса

Оператор Робертса является примером нелинейного фильтра. В обработке участвуют четыре пикселя, связанные попарно по диагоналям.

$$\begin{pmatrix} (x,y) & (x+1,y) \\ (x,y+1) & (x+1,y+1) \end{pmatrix} \tag{1.1}$$

Оператор можно записать следующим образом:

$$\mathrm{R}(x,y) = \sqrt{(f(x,y) - f(x+1,y+1))^2 + (f(x+1,y) - f(x,y+1))^2} \tag{1.2}$$

Такой оператор позволяет измерить разницу интенсивностей по двум диагоналям. Оператор Робертса еще называют пространственным дифференцированием. В области с однотонной интенсивностью оператор вернет нулевое значение, в ином случае будут выделены перепады интенсивности и границы.

Таким образом производится вычисление значений компонент вектора-градиента для каждой точки изображения путём свёртки локальной окрестности точки наложением малоразмерных масок

$$M_1 = \begin{vmatrix} 1 & 0 \\ 0 & -1 \end{vmatrix}, M_2 = \begin{vmatrix} 0 & -1 \\ 1 & 0 \end{vmatrix} \tag{1.3}$$

Иногда вместо оператора Робертса используют другую операцию пространственного дифференцирования, которая дает почти аналогичный результат:

$$\mathrm{R_m}(x, y) = |(f(x, y) - f(x + 1, y + 1))| + |(f(x + 1, y) - f(x, y + 1))^2| \tag{1.4}$$

Эксперименты показывают, что оператор Робертса не является в достаточной мере помехозащищённым. Один из самых ранних методов выделения контуров, оператор Робертса продолжает использоваться по причине его простоты и скорости обработки.

1.2.2. Оператор Собела

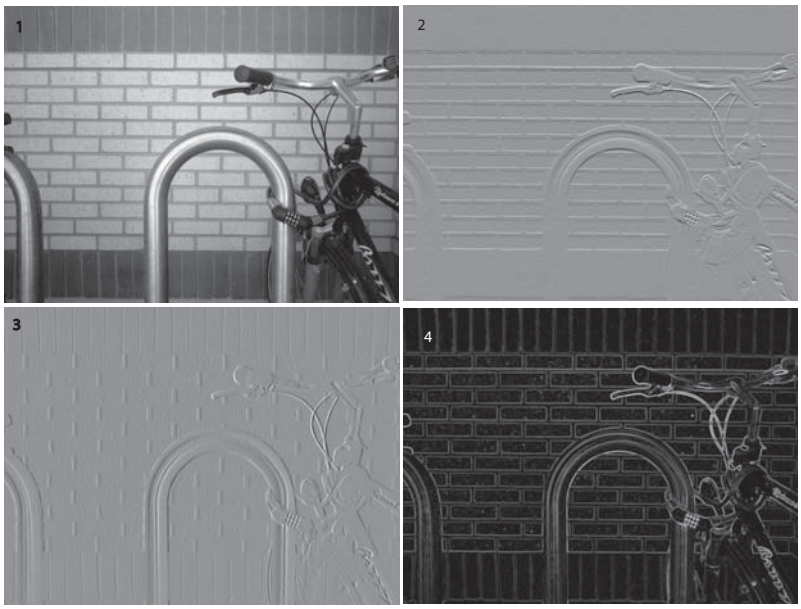


Рис. 1.1. Оператор Собела: 1 - исходное изображение, 2 - x-градиент, 3 - y-градиент, 4 - результирующий вектор-градиент

Оператор Собела вычисляет векторы-градиенты интенсивности в участках изображения, определяя направления наибольшего увеличения яркости и величину её изменения. Таким образом, результат его применения показывает непрерывность изменения яркости, что является критерием наличия контура, и даже даёт возможное направление контура. Практически, показатели направления контура в результатах работы оператора Собела сложнее интерпретировать и использовать, нежели показатели изменения яркости, также они являются менее точными.

Математический смысл оператора Собела состоит в вычислении вектора-градиента функции яркости точки, то есть вектора, компонентами которого являются производные функции яркости по вертикальному и горизонатальному направлениям. Для каждой точки вектор-градиент направлен в сторону максимального возможного увеличения яркости, а его длина

показывает величину этого изменения. На участке с постоянным уровнем яркости компоненты этого вектора нулевые, на участке, содержащем контур, вектор направлен по нормали к контуру, в сторону увеличения яркости. Применение оператора Собела осуществляется свёрткой окрестности точки A наложением масок

$$M_1 = \begin{vmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{vmatrix}, M_2 = \begin{vmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix} \tag{1.5}$$

Результирующий вектор-градиент, инвариантный к поворотам, получается из выражения $G = \sqrt{G_x^2 + G_y^2}$, где $G_x = A * M_1, G_y = A * M_2$, а угол θ его направления вычисляется так: $\theta = \arctan(\frac{G_y}{G_x})$. G не зависит от поворота, а величина θ будет меняться с поворотом изображения на угол $d\theta$ на тот же угол. Точность углового разрешения для масок 3×3 равна примерно $4°$.

1.2.3. Оператор Прюитт

Оператор Прюитт аналогичен опреатору Собела, с тем лишь отличием, что он не включает в себя дополнительного усиления контура в направлении вектора-градиента. Аналогично оператору Собела, свёртка по горизонтальной и вертикальной маскам

$$M_1 = \begin{vmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{vmatrix}, M_2 = \begin{vmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{vmatrix} \tag{1.6}$$

даёт составляюще градиента G_x и G_y, результирующий вектор-градиент и угол направления контура вычисляются по тем же формулам, что и в детекторе Собела.

1.3. Операторы на основе второй производной

Эти операторы основаны на вычислении симметричных круговых производных.

1.3.1. Лапласиан

Простейшим оператором такого класса является оператор Лапласа(лапласиан). Лапласиан 3×3имеет маску

$$\begin{vmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{vmatrix} \tag{1.7}$$

Такую маску можно интерпретировать как сумму разностей центрального элемента с каждым из восьми его непосредственных соседей. Таким образом, в равной степени учитываются возможные перепады яркости во всех направлениях.

1.3.2. Оператор Марра

Оператор Марра выделения краёв "ступенчатого" типа основан на поиске пересечений нуля второй пространственной производной $f(x, y)$. Для этого используется оператор Лапласа ∇^2, где ∇ - Гамильтониан $(\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$, применённый после сглаживания изображения гауссовским

линейным фильтром с симметричной маской $G(\sigma, x, y)$, или непосредственно осуществляется свёртка с маской $C^2 * G(\sigma, x, y)$. Этот фильтр также известен как РГР-фильтр(разности гауссовских распределений), так как форма маски $\nabla^2 G(\sigma, x, y)$ хорошо аппроксимируется разностью гауссовских масок $G(\sigma_1, x, y) - G(\sigma_2, x, y)$ с соотношением $\frac{\sigma_1}{\sigma_2} = 1.7$.

Оператор Марра является инвариантным к повороту, если носителем его маски является круговая область. Этот оператор не вычисляет в явном виде направления нормали к контуру. В то же время для определения множества контурных точек нет необходимости вводить искусственный порог(по модулю градиента), как в градиентных методах, так как он определяется непосредственно как точка пересечения нулевого уровня на отфильтрованном изображении. Ещё одним преимуществом оператора Марра является то, что получаемые с его помощью контуры не имеют разрывов. Возможна также масштабная настройка алгоритма путём выбора значения параметра σ.

1.4. Комплексные детекторы краев

1.4.1. Детектор Кэнни

Кэнни (1986) занялся математической проблемой получения оптимального фильтра размытия для обнаружения, выделеления и приведения контуров. Он показал, что оптимальный фильтр для этих целей может быть представлен суммой четырёх экспоненциальных критериев. Также он доказал, что этот фильтр может быть успешно аппроксимирован производной первого порядка фильтра Гаусса. Кэнни сформулировал понятие подавления немаксимальных компонент, означающее, что для работы фильтров предварительного размытия контурные точки - это те, в которых величина градиента достигает локального максимума в его направлении.

Требования, поставленные им к детектору контуров, состоят в следующем:

- Оптимальное обнаружение - алгоритм должен обнаруживать максимальное количество существующих контуров в изображении
- Оптимальная локализация - алгоритм должен выделять контур максимально близко к краю, обнаруженному в изображении
- Единственность отклика на перепад - алгоритм должен отметить контур лишь однажды и, по возможности, подавить ложные контуры, порождённые шумами

Работа детектора Кэнни состоит из следующих стадий:

1.4.1..1 Подавление шума

Поскольку детектор краёв Кэнни использует первую производную гауссовского фильтра, он восприимчив к шуму на изображении, поэтому перед обработкой производится сглаживание по Гауссу, чтобы отдельные шумовые пикселы в последствии не оказали влияния на детекцию. Вот пример гауссовского фильтра 5×5, применяющегося в детекторе Кэнни с $\sigma = 0.4$:

$$B = \frac{1}{159} \begin{vmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{vmatrix} \tag{1.8}$$

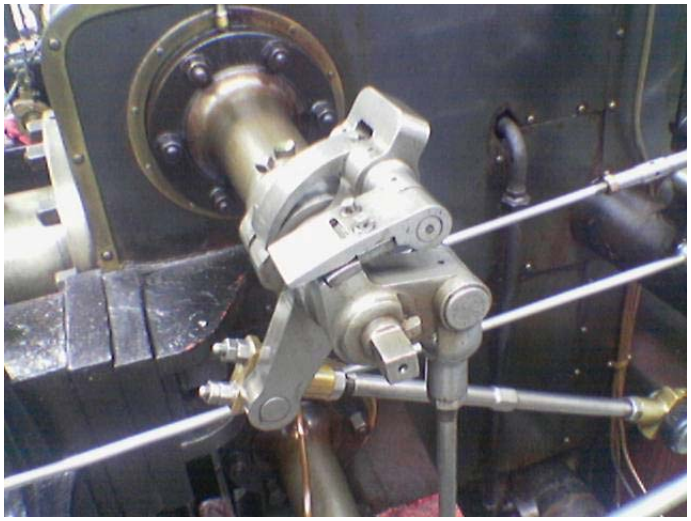


Рис. 1.2. Исходное изображение

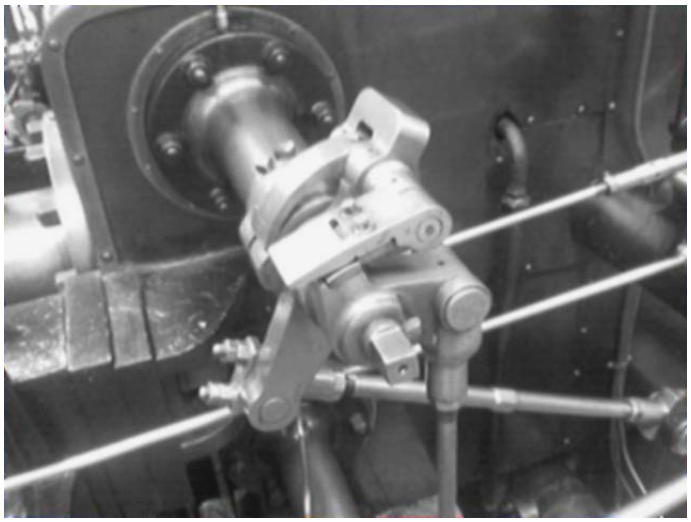


Рис. 1.3. После применения фильтра Гаусса

1.4.1..2 Нахождение градиента яркости изображения

Контур в изображении может быть ориентированным по множеству направлений, поэтому детектор Кэнни использует четыре фильтра для горизонтальных, вертикальных и диагональных контуров, соответственно. Детекторы первой производной - Прюитта, Робертса или Собела - вычисляют первую производную функции яркости по вертикальному и горизонтальному направлениям. Их методами можно получить значение компонент и угла направления вектора-градиента. После этого угол направления округляется до одного из четырёх значений - $0°, 45°, 90°, 135°$

1.4.1..3 Подавление немаксимальных значений компонент градиента

После получения оценок градиента по первой производной, детектор определяет, является ли величина градиента максимальной по его направлению. Таким образом, если округлённый угол градиента равен $0°$, то значения яркости сверху и снизу от него должны убывать. Если

условие не выполняется, контур отсеивается.

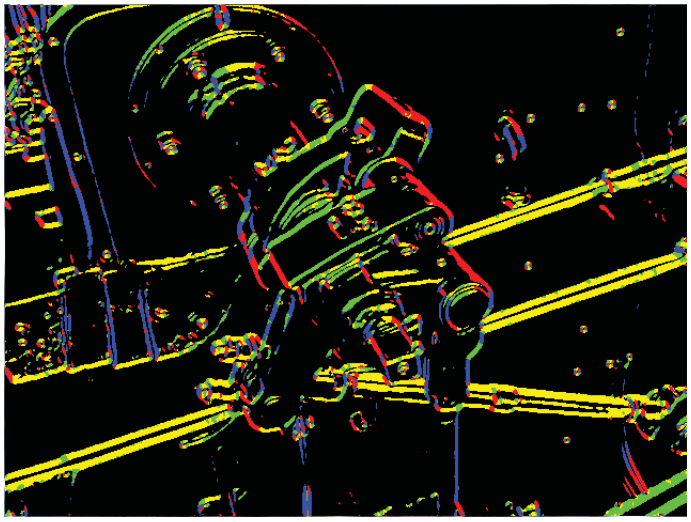


Рис. 1.4. До подавления

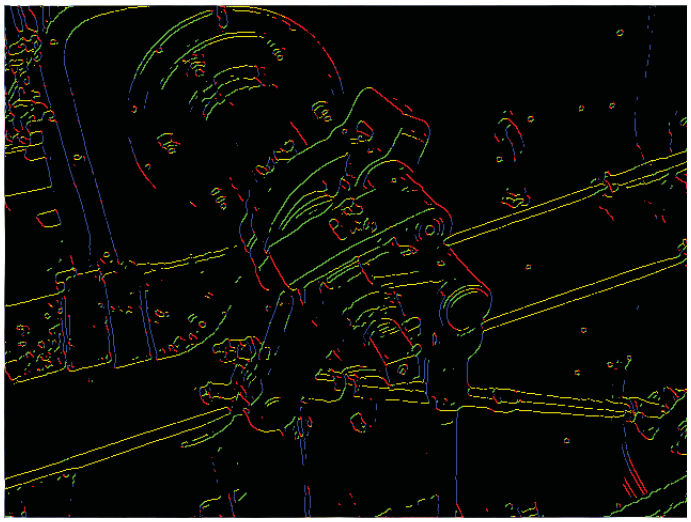


Рис. 1.5. После подавления немаксимальных значений градиента

1.4.1..4 Отслеживание контура и пороговый гистерезис

Величина градиента напрямую связана с наличием в представляемом им участке контура, но в большинстве случаев, невозможно установить порог, при значении градиента ниже которого в участке не существует контура. По этой причине детектор Кэнни использует пороговый гистерезис.

Определение порогов с гистерезисом требует двух порогов - верхнего и нижнего. Предположение, что важные контуры являются частью более или менее продолжительных кривых, даёт нам возможность следовать по линии контура при уменьшении яркости и отбрасывать участки, дающие большую величину градиента, но не имеющие продолжения в линии контура. В последнем состоит причина использования верхнего порога.

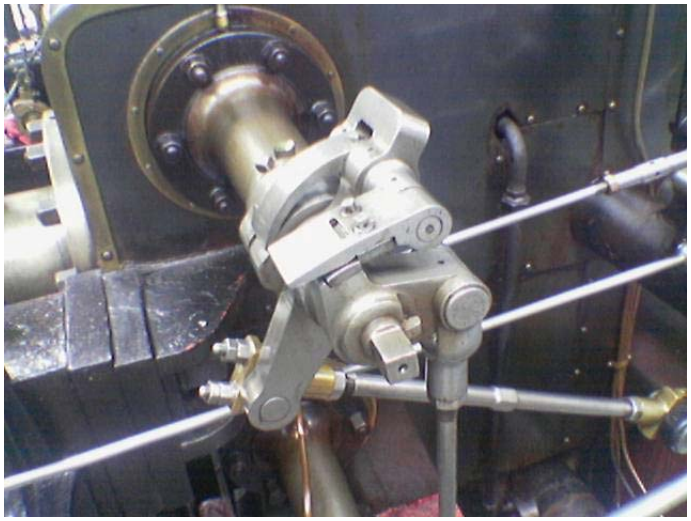


Рис. 1.6. Результат работы фильтра Кэнни

1.4.1..5 Параметры детектора Кэнни

Время работы и эффективность детектора Кэнни зависит от начальных параметров, таких как коэффициент сглаживания и пороговые значения. Коэффициент сглаживания вляет на чувствительность детектора, сохраняя или подавляя небольшие различия в яркости. Пороговые параметры требуют тонкой настройки для избежания присоединения шумовых точек к контуру или разрывов сплошных контуров.

Глава 2.

Введение в теорию нейронных сетей

После двух десятилетий почти полного забвения в начале 90-х годов XX века интерес к искусственным нейронным сетям быстро возрос. С появлением новых возможностей, специалисты из таких разных областей, как техническое конструирование, философия, физиология и психология, заинтересовались этой технологией, и ищут приложения ей для своих задач. Это возрождение интереса было вызвано как теоретическими, так и прикладными достижениями. Неожиданно открылись возможности использования вычислений в сферах, до этого относящихся лишь к области человеческого интеллекта, возможности создания машин, способность которых учиться и запоминать удивительным образом напоминает мыслительные процессы человека, и наполнения новым значительным содержанием критиковавшегося термина "искусственный интеллект".

2.1. Свойства нейронных сетей

Идея искусственных нейронных сетей порождена биологией, так как они состоят из элементов, функциональные возможности которых аналогичны большинству элементарных функций биологического нейрона. Эти элементы затем организуются по способу, который возможно подобен анатомии мозга. Несмотря на такое поверхностное сходство, искусственные нейронные сети демонстрируют удивительное число свойств присущих человеческому мозгу. Например, они способны обучатся на основе опыта, обобщать предыдущие прецеденты для анализа новых случаев и извлекать существенные свойства из поступающей информации, содержащей избыточные данные. Несмотря на такое функциональное сходство, было бы, безусловно, необоснованно предполагать, что в скором будущем искусственные нейронные сети будут способны дублировать функции человеческого мозга. Однако равным образом было бы неверным игнорировать удивительное сходство в функционировании некоторых нейронных сетей с человеческим мозгом. Эти возможности, как бы они ни были ограничены сегодня, могут быть эффективно использованы во многих областях, где решение нетривиальных задач традиционными способами невозможно или чересчур трудоёмко. Нейронные сети во множестве случаев являются столь необходимым альтернативным подходом.

2.1.1. Обучение

Искусственные нейронные сети могут менять свое поведение в зависимости от внешней среды. Этому фактору в большей степени, чем любому другому, нейронные сети обязаны тем интересом, который они вызывают. После предъявления входных сигналов (возможно, вместе с требуемыми выходами) они самонастраиваются, чтобы обеспечивать требуемую реакцию.

Было разработано множество обучающих алгоритмов, каждый со своими сильными и слабыми сторонами. Обучение является самым сложным этапом в построении нейронных сетей.

2.1.2. Обобщение

Отклик сети после обучения может быть до некоторой степени нечувствителен к небольшим изменениям входных сигналов. Эта внутренне присущая робастность, способность видеть интересующий образ сквозь шум и искажения жизненно важна для распознавания образов в реальном мире. Она позволяет преодолеть требование строгой точности вычислений, предъявляемое обычным компьютером, и открывает подход к системам, которые могут обрабатывать информацию от реального мира, во многом далёкого от любых математических моделей. Важно отметить, что искусственная нейронная сеть делает обобщения автоматически исходя из своей структуры, без помощи созданных человеческим интеллектом сценариев.

2.1.3. Абстрагирование

Некоторые из искусственных нейронных сетей обладают способностью извлекать сущность из входных сигналов. Например, сеть может быть обучена на последовательности искаженных версий образа. После соответствующего обучения предъявление иного искаженного образа приведет к тому, что сеть породит образ совершенной формы, породив из неполных образов полный. В некотором смысле она научится порождать то, что никогда не видела. Эта способность извлекать идеальное из несовершенных входов ставит интересные философские вопросы

2.1.4. Применимость

Искусственные нейронные сети не являются панацеей. Область их применения на данный момент ограничивается следующими классами задач:

Распознавание образов и классификация: в качестве образов могут выступать различные по своей природе объекты: символы текста, изображения, образцы звуков и т. д. При обучении сети предлагаются различные образцы образов с указанием того, к какому классу они относятся. Образец, как правило, представляется как вектор из его признаков. При этом совокупность всех признаков должна однозначно определять класс, к которому относится образец. В случае, если признаков недостаточно, сеть может соотнести один и тот же образец с несколькими классами, что неверно. По окончании обучения сети можно предъявлять неизвестные ей ранее образы и получать от нее ответ о принадлежности к определенному классу. Топология такой сети характеризуется тем, что количество нейронов в выходном слое, как правило, равно количеству определяемых классов. При этом устанавливается соответствие между выходом нейронной сети и классом, который он представляет. Когда сети предъявляется некий образ, на одном из ее выходов должен появится признак того, что образ принадлежит этому классу. В то же время на других выходах должен быть признак того, что образ данному классу не принадлежит. Если на двух или более выходах есть признак принадлежности к классу, считается что сеть "не уверена"в своем ответе.

Принятие решений и управление: эта задача близка к задаче классификации. Классификации подлежат ситуации, характеристики которых поступают на вход нейронной сети. На выходе сети при этом должен появится признак решения, которое она приняла. При

этом в качестве входных сигналов используются различные критерии описания состояния управляемой системы.

Кластеризация: под кластеризацией понимается разбиение множества входных сигналов на классы, при том, что ни количество, ни признаки классов заранее неизвестны. После обучения такая сеть способна определять, к какому классу относится входной сигнал. Сеть также может сигнализировать о том, что входной сигнал не относится ни к одному из выделенных классов - это является признаком новых, отсутствующих в обучающей выборке, данных. Таким образом, подобная сеть может выявлять новые, неизвестные ранее классы сигналов. Соответствие между классами, выделенными сетью, и классами, существующими в предметной области, устанавливается человеком.

Прогнозирование и аппроксимация: способности нейронной сети к прогнозированию напрямую следуют из ее способности к обобщению и выделению скрытых зависимостей между входными и выходными данными. После обучения сеть способна предсказать будущее значение некой последовательности на основе нескольких предыдущих значений и/или каких-то существующих в настоящий момент факторов. Следует отметить, что прогнозирование возможно только тогда, когда предыдущие изменения действительно в какой-то степени предопределяют будущие. Например, прогнозирование котировок акций на основе котировок за прошлую неделю может оказаться успешным (а может и не оказаться), тогда как прогнозирование результатов завтрашней лотереи на основе данных за последние 50 лет почти наверняка не даст никаких результатов.

Сжатие данных и ассоциативная память: способность нейросетей к выявлению взаимосвязей между различными параметрами дает возможность выразить данные большой размерности более компактно, если данные тесно взаимосвязаны друг с другом. Обратный процесс - восстановление исходного набора данных из части информации - называется (авто)ассоциативной памятью. Ассоциативная память позволяет также восстанавливать исходный сигнал и образ из зашумленных и поврежденных входных данных.

2.2. История изучения

Людей всегда интересовало их собственное мышление. Самоанализ, размышление мозга о себе самом является, возможно, одной из важнейших отличительных черт человеческого мышления. Имеется множество теорий о природе мышления, от духовных до анатомических. Ни одна из них не смогла дать полного ответа на поставленные вопросы, так как сам предмет весьма труден для изучения, а подходы конфликтуют между собой. Самоанализ и размышление приводят к выводам, не отвечающим уровню строгости физических наук. Наблюдение и эксперимент же говорят только о том, что мозг труден для изучения и ставит в тупик своей организацией. Короче говоря, мощные методы научного исследования, изменившие наш взгляд на физическую и нервную активность человека, оказались бессильными в понимании самого человека.

Между тем, нейробиология и нейроанатомия достигли значительного прогресса. Изучая структуру и функции нервной системы человека, они получили представление об электрической природе распространения сигналов в мозге, но мало узнали, о том, как именной он функционирует. В процессе накопления знаний выяснилось, что мозг чрезвычайно сложен. Огромное количество нейронов, каждый из которых соединен с множеством других, образует систему, превосходящую по сложности все представления об устройстве суперкомпьютера, который мог бы имитировать работу мозга. Но, тем не менее, наука узнаёт всё больше о

человеческом мозге. Лучшее понимание функционирования нейрона и картины его связей позволило исследователям создать математические модели для проверки своих теорий. Эксперименты теперь могут проводиться на цифровых компьютерах без привлечения опытов на живых существах, что решает многие практические и морально-этические проблемы. В первых же работах выяснилось, что эти модели не только повторяют функции мозга, но и способны выполнять функции, имеющие свою собственную ценность. Поэтому возникли и остаются в настоящее время две взаимно обогащающие друг друга цели нейронного моделирования: первая - понять функционирование нервной системы человека на уровне физиологии и психологии и вторая - создать вычислительные системы (искусственные нейронные сети), выполняющие функции, сходные с функциями мозга.

Параллельно с прогрессом в нейроанатомии и нейрофизиологии психологами были созданы модели человеческого обучения. Одной из таких моделей, оказавшейся наиболее плодотворной, была модель Хэбба, который в 1949 году предложил закон обучения, явившийся стартовой точкой для алгоритмов обучения искусственных нейронных сетей. Дополненный сегодня множеством других методов он продемонстрировал ученым того времени, как сеть нейронов может обучаться. В пятидесятые и шестидесятые годы группа исследователей, объединив эти биологические и физиологические подходы, создала первые искусственные нейронные сети. Выполненные первоначально как электронные сети, они были позднее перенесены в более гибкую среду компьютерного моделирования, сохранившуюся и в настоящее время. Первые успехи вызвали взрыв активности и оптимизма. Минский, Розенблатт, Уидроу и другие разработали сети, состоящие из одного слоя искусственных нейронов. Часто называемые перцептронами, они были использованы для такого широкого класса задач, как предсказание погоды, анализ электрокардиограмм и искусственное зрение. В течение некоторого времени казалось, что ключ к интеллекту найден и воспроизведение человеческого мозга является лишь вопросом конструирования достаточно большой сети. Но эта иллюзия скоро рассеялась. Сети не могли решать задачи, внешне весьма сходные с теми, которые они успешно решали. С этих необъяснимых неудач начался период интенсивного анализа. Минский, используя точные математические методы, строго доказал ряд теорем, относящихся к функционированию сетей.

Исследования Минского описано в его книге, в которой он доказал, что используемые в то время однослойные сети теоретически неспособны решить многие простые задачи, в том числе реализовать функцию "Исключающее ИЛИ". Минский также не был оптимистичен относительно потенциально возможного здесь прогресса:

> «Перцептрон показал себя заслуживающим изучения, несмотря на жесткие ограничения (и даже благодаря им). У него много привлекательных свойств: линейность, занимательная теорема об обучении, простота модели параллельных вычислений. Нет оснований полагать, что эти достоинства сохраняться при переходе к многослойным системам. Тем не менее мы считаем важной задачей для исследования подкрепление (или опровержение) нашего интуитивного убеждения, что такой переход бесплоден.
>
> Возможно, будет открыта какая-то мощная теорема о сходимости или найдена глубокая причина неудач дать интересную "теорему обучения"для многослойных машин»[4]

Неоспоримая аргументация Минского, а также его репутация вызвали огромное доверие к книге - ее выводы были приняты. Разочарованные исследователи оставили поле исследований ради более многообещающих областей, а правительства перераспределили свои субсидии, и

искусственные нейронные сети были забыты почти на два десятилетия. Тем не менее несколько наиболее настойчивых ученых, таких как Кохонен, Гроссберг, Андерсон продолжили исследования. Наряду с плохим финансированием и недостаточной оценкой ряд исследователей испытывал затруднения с публикациями. Поэтому исследования, опубликованные в семидесятые и начале восьмидесятых годов, разбросаны в массе различных журналов, некоторые из которых малоизвестны. Постепенно появился теоретический фундамент, на основе которого сегодня конструируются наиболее мощные многослойные сети. Оценка Минского оказалась излишне пессимистичной, многие из поставленных в его книге задач решаются сейчас сетями с помощью стандартных процедур. Возможно, что шок, вызванный его книгой «Перцептроны», обеспечил необходимый для созревания этой научной области период.

За последние несколько лет теория стала применяться в прикладных областях и появились новые корпорации, занимающиеся коммерческим использованием этой технологии. Нарастание научной активности носило взрывной характер. В 1987 г. было проведено четыре крупных совещания по искусственным нейронным сетям и опубликовано свыше 500 научных сообщений.

2.3. Основные виды нейронных сетей

Несмотря на существенные различия, отдельные типы нейронных сетей обладают несколькими общими чертами. Во-первых, основу каждой нейронной сети составляют относительно простые, в большинстве случаев - однотипные, элементы (ячейки), имитирующие работу нейронов мозга. Далее под нейроном будет подразумеваться искусственный нейрон, то есть ячейка нейронной сети. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены или заторможены. Он обладает группой синапсов - однонаправленных входных связей, соединенных с выходами других нейронов, а также имеет аксон - выходную связь данного нейрона, с которой сигнал (возбуждения или торможения) поступает на синапсы следующих нейронов. Общий вид нейрона приведен на рис.2.1. Каждый синапс характеризуется величиной синаптической связи или ее весом w_i, который по физическому смыслу эквивалентен электрической проводимости.

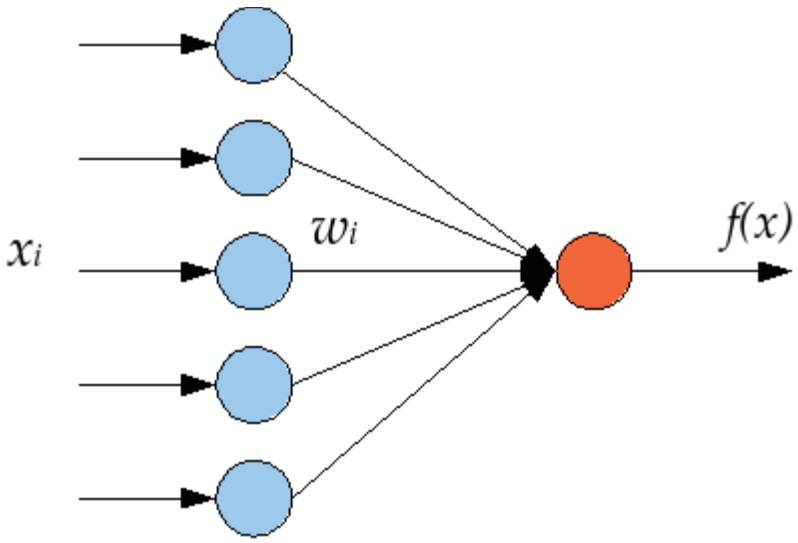


Рис. 2.1. Однослойный перцептрон

Текущее состояние нейрона определяется, как взвешенная сумма его входов:

$$s = \sum_{i=1}^{n} x_i \cdot w_i \tag{2.1}$$

Выход нейрона есть функция его состояния:

$$y = f(s) \tag{2.2}$$

Нелинейная функция f называется активационной и может иметь различный вид, как показано на рисунке 2. Одной из наиболее распространеных является нелинейная функция с насыщением, так называемая логистическая функция или сигмоид (т.е. функция S-образного вида):

$$f(x) = \frac{1}{1 + e^{-\alpha x}} \tag{2.3}$$

При уменьшении α сигмоид становится более пологим, в пределе при $\alpha = 0$ вырождаясь в горизонтальную линию на уровне 0.5, при увеличении α сигмоид приближается по внешнему виду к функции единичного скачка с порогом T в точке x=0. Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне $[0, 1]$. Одно из ценных свойств сигмоидной функции - простое выражение для ее производной, применение которого будет рассмотрено в дальнейшем.

$$f'(x) = \alpha \cdot f(x)(1 - f(x)) \tag{2.4}$$

Следует отметить, что сигмоидная функция дифференцируема на всей оси абсцисс, что используется в некоторых алгоритмах обучения. Кроме того она обладает свойством усиливать слабые сигналы лучше, чем большие, и предотвращает насыщение от больших сигналов, так как они соответствуют областям аргументов, где сигмоид имеет пологий наклон. Возвращаясь к общим чертам, присущим всем нейронным сетям, отметим, принцип параллельной обработки сигналов, который достигается путем объединения большого числа нейронов в так называемые слои и соединения определенным образом нейронов различных слоев, а также, в некоторых конфигурациях, и нейронов одного слоя между собой, причем обработка взаимодействия всех нейронов ведется послойно.

2.3.1. Нейронные сети прямого распространения

Нейронные сети прямого распространения - это сети, топология которых может быть представлена ациклическим графом, то есть не включает в себя ни одной обратной связи. Такие сети были первыми и, возможно, простейшими из предложенных искусственных нейронных сетей. Входные данные продвигаются только в одном направлении, от входных узлов через внутренние слои, если таковые имеются, к выходным. В сетях прямого распространения не может быть петель и циклов.

2.3.1..1 Однослойный перцептрон

Самым первым примером нейронной сети был однослойный перцептрон, состоящий из единственного слоя выходных узлов. Входные данные в такой сети подавались непосредственно на синапсы выходных узлов. В каждом узле вычисляется сумма взвешенных входных сигналов, и если она превышает некоторый предел(обычно 0), происходит активация нейрона,

его весовой параметр становится обычно равным 1. Иначе нейрон остаётся неактивированным, сохраняя значение -1. Такая активационная функция называется функцией единичного скачка. В литературе термин "перцептрон"обычно относится к сетям, описанным МакКаллоком и Питтсом в 1940-х годах, состоящим из одного нейрона.

Значения состояний активации и деактивации нейрона могут быть любыми числами, ограничивающими отрезок, внутри которого находится активационный порог. Большинство перцептронов пользуются значениями $-1, 1$, так как это ускоряет процесс обучения. Перцептрон обучается с применением дельта-правила, заключающего в себе оба правила Хэбба. Оно рассчитывает разницу между желаемыми и полученнымы выходными данными для корректировки весов, уменьшая ошибку по принципу градиентного спуска.

Однонейрнонные перцептроны способны различать только линейно разделимые классы. В упомянутой выше работе Минского и Пайперта это послужило основой для неверного предположения, что и многослойный перцептрон неспособен реализовать операцию XOR. Несмотря на то, что однопороговый узел весьма ограничен в своих вычислительных возможностях, сеть из параллельных перцептронов может аппрокимировать любую функцию на ограниченном участке в отрезок $[-1, 1]$. Новейшие результаты этих исследований появились в 2001-2003 годах в работах Ауэра, Бергштайнера, Маасса. Однослойный перцептрон может использовать также непрерывную активационную функцию, к примеру, сигмоид:

$$y = \frac{1}{1 + e^{-x}} \tag{2.5}$$

Эта функция непрерывно дифференцируема, что позволяет использовать для обучения алгоритм обратного распространения ошибки, и производная её легко вычислима.

2.3.1..2 Многослойный перцептрон

Двухслойный перцептрон уже способен вычислять функцию XOR. Вообще, в многослойном перцептроне весовые числа нейронов являются их отдельными порогами, и если взвешенная сумма входов не достигает порога, выходом будет 0, а не -1. Выходы нейронов одного слоя связаны со входами нейронов следующего слоя, а в качестве активационной функции обычно используется сигмоид.Универсальная теорема аппроксимации для нейронных сетей утверждает, что любая непрерывная функция, проецирующаяя интервал действительных чисел в некоторый ограниченный интервал выходных значений, может быть аппроксимирована нейронной сетью с одним внутренним слоем. Это справедливо при использовании ограниченного класса активационных функций - сигмоидальных.

Многослойная нейронная сеть может использовать различные алгоритмы обучения, чаще всего это алгоритм обратного распространения ошибки. По этому алгоритму вычисляется функция ошибок выходов нейронов относительно верных ответов, после чего коррекция весов проводится по сети в направлении, обратном прохождению входных сигналов. Последовательно повторяя этот шаг, можно добиться уменьшения ошибки до установленного предела. Чтобы правильно настроить веса, применяется метод градиентного спуска, по которому согласно значениям производной функции ошибок веса корректируются в сторону её минимизации. Этот алгоритм может применяться только в сетях с дифференцируемыми активационными функциями.

Согласно теории машинного обучения, требуется эвристический подход к проблеме ограничения обучающего набора для сети, чтобы не произошло перенасыщение сети, при котором сеть утратит способность выявлять настоящие закономерности, становясь зависимой от конкретного обучающего набора.

2.3.2. Сети с обратными связями(рекуррентные)

У сетей, рассмотренных до сих пор, не было обратных связей, т. е. соединений, идущих от выходов некоторого слоя к входам предшествующих слоев. Этот специальный класс сетей, называемых сетями без обратных связей или сетями прямого распространения, представляет интерес и широко используется. Сети более общего вида, имеющие соединения от выходов к входам, называются сетями с обратными связями. У сетей без обратных связей нет памяти, их выход полностью определяется текущими входами и значениями весов. В некоторых конфигурациях сетей с обратными связями предыдущие значения выходов возвращаются на входы; выход, следовательно, определяется как текущим входом, так и предыдущими выходами. По этой причине сети с обратными связями могут обладать свойствами, сходными с кратковременной человеческой памятью, сетевые выходы частично зависят от предыдущих входов.

2.3.2..1 Простые рекуррентные сети

Простые рекуррентные сети, называемые также сетями Элмана, являются разновидностью многомлойного перцептрона. В них используется трёхслойная сеть с добавлением "контекстных"узлов во входной слой. Скрытый слой связан с контекстными узлами связями с константными единичными весами. На каждом этапе входные сигналы распространияются в прямом направлении, а затем применяется обучающий алгоритм, таким образом, контекстные узлы всегда содержат копии значений внутреннего слоя на предыдущем этапе. Так сеть может поддерживать состояние в котором она может выполнять задачу предсказания последовательности, что невозможно для обычного многослойного перцептрона.

В полной рекуррентной сети каждый нейрон получает входные сигналы других нейронов сети. Такая сеть неразделима на слои, и только ограниченное количество нейронов получает входные сигналы извне, и другое ограниченное множество нейронов выводит выходные сигналы за пределы сети, также отправляя их другим нейронам внутри сети. Эти множества выполняют роль входного и выходного слоёв соответственно, сохраняя также обратную связь с остальными нейронами сети.

2.3.2..2 Сети Хопфилда

Сети Хопфилда - это рекуррентные сети, в которых все связи симметричны. Они были изобретены Джоном Хопфилдом в 1982 году и их сходимость доказана. Если обучить сеть Хопфилда по правилам Хэбба, она сможет выполнять функции робастной ассоциативной памяти, устойчивой к изменениям связей.

2.3.3. Стохастические нейронные сети

Особенность стохастических нейронных сетей заключается в применении случайных значений для весов. В вероятностном подходе к нейронным сетям такие сети рассматриваются как реализации метода статистического семплирования, такого как метод Монте-Карло.

2.3.3..1 Машина Больцманна

Изобретённая Хинтоном и Сейновским в 1985 году, машина Больцманна является сетью Хопфилда с шумом. Она была первой сетью, продемонстрировавшей обучение внутренних

слоёв. Обучение машины Больцманна в своей основе похоже на обучение Хэбба и очень сложно моделируемо без наложения ограничений на связи между нейронами. Машина Больцманна может быть применена для решения различных комбинаторных задач. В последние годы авторами машины Больцманна были предложены более эффективные методы обучения.

2.3.4. Самоорганизующиеся карты Кохонена

Самоорганизующиеся карты, предложенные Т.Кохоненом , являются соревновательными нейронными сетями, использующими алгоритм обучения без учителя. Они осуществляют проекцию многомерного пространства в пространство с более низкой размерностью (чаще всего, двумерное) и применяются для решения задач моделирования, прогнозирования и прочих.

2.3.4..1 Структура сети

Самоорганизующаяся карта состоит из компонент, называемых узлами или нейронами. Их количество задаётся аналитиком. Каждый из узлов описывается двумя векторами. Первый - так называемый вектор веса m, имеющий такую же размерность, что и входные данные. Второй - координаты узла на карте, далее вектор r. Обычно узлы располагают в вершинах регулярной решётки с квадратными или шестиугольными ячейками. Изначально известна размерность входных данных, по ней некоторым образом строится первоначальный вариант карты. В процессе обучения векторы веса узлов приближаются к входным данным. Для каждого наблюдения выбирается наиболее похожий по вектору веса узел, и значение его вектора веса приближается к наблюдению. Также к наблюдению приближаются векторы веса нескольких узлов, расположенных рядом, таким образом, если в множестве входных данных два наблюдения были схожи, на карте им будут соответствовать близкие узлы. Циклический процесс обучения, перебирающий входные данные, заканчивается по достижении картой допустимой (заранее заданной аналитиком) погрешности, или по совершении заданного количества итераций.

2.3.4..2 Работа сети Кохонена

Первый этап - инициализация карты, то есть первоначальное задание векторов веса для узлов. Производится либо инициализация случайными значениями, либо случайным набором значений из входных данных, либо значениями из линейного пространства, образованного главными компонентами пространства входных данных. Затем сеть начинает итерации по наборам входных данных, на каждом шаге выбирая из них случайный набор. Здесь сеть вычисляет ближайший по весу узел $M_c(t)$ - победитель согласно условию

$$\|x(t) - m_c(t)\| \leq \|x(t) - m_i(t)\|, \tag{2.6}$$

где $m_i(t)$ - весовой вектор узла $M_i(t)$. Если по условию прошло несколько узлов, победитель из них выбирается случайным образом. Далее с помощью функции соседства h определяется корректировка весов соседних узлов сети. Эта функция определяет меру соседства и степень изменения весов узлов, постепенно обрабатывая всё меньшее количество узлов, и слабее изменяя их параметры. Часто в качестве функции соседства используется гауссовская функция:

$$h_{ci}(t) = \alpha(t) \cdot \exp\left(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}\right), \tag{2.7}$$

где $0 < \alpha(t) < 1$ - обучающий множитель, монотонно убывающий с каждой итерацией t, r_i, r_c - координаты узлов $M_i(t)$ и $M_c(t)$ на карте, а $\sigma(t)$ - сомножитель, постепенно ограничивающий количество соседей выбранного узла, веса которых подвергаются корректировке. Параметры α, σ задаются аналитиком. Весовые векторы изменяются по закону

$$m_i(t) = m_i(t-1) + h_{ci}(t) \cdot (x(t) - m_i(t-1)), \tag{2.8}$$

таким образом, вектора веса всех узлов, являющихся соседями победителя, приближаются к рассматриваемому наблюдению. Последней стадией является вычисление ошибки карты, например, как среднее арифметическое расстояний между наблюдениями и векторами веса соответствующих им победителей:

$$\delta = \frac{1}{N} \sum_{i=1}^{N} \|x_i - m_c\|, \tag{2.9}$$

где N - количество элементов набора входных данных.

2.4. Алгоритмы обучения

Искусственные нейронные сети обучаются самыми разнообразными методами. К счастью, большинство методов обучения исходят из общих предпосылок и имеет много идентичных характеристик. Целью данного приложения является обзор некоторых фундаментальных алгоритмов, как с точки зрения их текущей применимости, так и с точки зрения их исторической важности. После ознакомления с этими фундаментальными алгоритмами другие, основанные на них, алгоритмы будут достаточно легки для понимания и новые разработки также могут быть лучше поняты и развиты.

2.4.1. Обучение с учителем и без учителя

Обучающие алгоритмы могут быть классифицированы как алгоритмы обучения с учителем и без учителя. В первом случае существует учитель, который предъявляет входные образы сети, сравнивает результирующие выходы с требуемыми, а затем настраивает веса сети таким образом, чтобы уменьшить различия. Трудно представить такой обучающий механизм в биологических системах; следовательно, хотя данный подход привел к большим успехам при решении прикладных задач, он отвергается исследователями, полагающими, что искусственные нейронные сети обязательно должны использовать те же механизмы, что и человеческий мозг. Во втором случае обучение проводится без учителя, при предъявлении входных образов сеть самоорганизуется посредством настройки своих весов согласно определенному алгоритму. Вследствие отсутствия указания требуемого выхода в процессе обучения результаты непредсказуемы с точки зрения определения возбуждающих образов для конкретных нейронов. При этом, однако, сеть организуется в форме, отражающей существенные характеристики обучающего набора. Например, входные образы могут быть классифицированы согласно степени их сходства так, что образы одного класса активизируют один и тот же выходной нейрон.

В процессе обучения сеть в определенном порядке просматривает обучающую выборку. Порядок просмотра может быть последовательным, случайным. Сети, обучающиеся без учителя, просматривают выборку только один раз. При обучении с учителем сеть просматривает выборку множество раз, при этом один полный проход по выборке называется эпохой обучения. Обычно набор исходных данных делят на две части - собственно обучающую выборку и

тестовые данные; принцип разделения может быть произвольным. Обучающие данные подаются сети для обучения, а проверочные используются для расчета ошибки сети (проверочные данные никогда для обучения сети не применяются). Таким образом, если на проверочных данных ошибка уменьшается, то сеть действительно выполняет обобщение. Если ошибка на обучающих данных продолжает уменьшаться, а ошибка на тестовых данных увеличивается, значит, сеть перестала выполнять обобщение и просто "запоминает"обучающие данные. Это явление и называется переобучением сети или оверфиттингом. В таких случаях обучение обычно прекращают. В процессе обучения могут проявиться другие проблемы, такие как паралич или попадание сети в локальный минимум поверхности ошибок. Невозможно заранее предсказать проявление той или иной проблемы, равно как и дать однозначные рекомендации к их разрешению.

Глава 3.

Применение нейросетевой технологии для выделения контуров

3.1. Введение

Выявление контуров, помимо описанных фильтров, можно осуществлять с помощью нейросетевой технологии. Используя подборку карт контуров в качестве базы данных для обучения сети специально спроектированной архитектуры, можно получить сеть, выполняющую задачу выделения контуров и способную абстрагироваться от изображений обучающего набора и выполнять поставленную задачу на изображениях, не входивших в базу данных.

В общем случае, задача выделения контуров состоит из двух стадий: выявление участков изображения, существенно отличающихся по яркости от соседних с ними, и сравнение выявленных отличий с неким предустановленным пороговым значением, определяющим действительную достаточность этих различий в участках для объявления их участками контура. На первой стадии требуется аппарат выявления отличий в яркости, в роли которого может выступать, к примеру, оператор Прюитта, Собеля или Робертса. На второй же стадии, детектору обычно требуется задать пороговые значения для оптимальной обработки изображения.

Выделение контуров грубыми способами является интерактивным процессом, в котором пользователь должен корректировать глобальные пороговые параметры, так как результат редко сразу бывает удовлетворительным по той причине, что не настроенный под конкретное изображение детектор может ошибочно выделять зашумлённые участки изображения, как содержащие контурные точки. Более сложные методы, такие как детектор Кенни и детектор Шена-Кастана, используют так называемую операцию порогового гистерезиса, при которой за значимый контур принимается последовательность пикселов, в которой отличие в яркости от окружения по меньшей мере одного пиксела превосходит верхнее пороговое значение, тогда как остальные пикселы последовательности превышают нижний порог. К тому же, для выявления характеристик контура на каждом участке шириной в один пиксель требуется использование преобразования Лапласа от фильтра Гаусса для выделения нулевых пересечений, в связи с чем требуется задание дополнительных параметров. Добавление этих параметров к уже имеющимся пороговым увеличивает количество вариаций наборов параметров, каждый из которых даст свою контурную карту изображения.

Преимущество применения нейросетевой технологии в данной задаче заключается в том, что оно позволяет сократить количество параметров детектора.

Учитывая, что сложная операция выявления контуров тем не менее потребует определения достаточно большого количества параметров, логично будет представить их в виде весов связей в нейронной сети. За основу следует взять иерархическую архитектуру, предложен-

ную Кунгом и Тауром и изменить её для решения задачи выявления контуров таким образом, что веса связей будут играть двойную роль - моделировать разновидности окружения контуров на начальной стадии поиска контуров и выступать в качестве пороговых параметров на финальной стадии принятия решения.

3.2. Нейронная сеть для определения характеристик границ

Обычно иерархическая нейронная сеть состоит из набора взвешенных, или параметризованных по весу нейронов-моделей в качестве вычислительных элементов, а выход отдельной подсети определяется линейной комбинацией локальных выходов входящих в неё нейронов. В случае задачи поиска контуров эта схема нуждается в модификации: параметризованные по весу нейроны будут заменены альтернативной моделью - так называемыми взвешенными по входу нейронами, а выход подсети будет определяться в ходе соревновательного процесса по принципу "победитель получает всё".

3.2.1. Взвешенный по входу нейрон-модель

Вместо отображения вложенного вектора весов $z \in R^M$ меньшей размерности в весовой вектор высокой размерности $p \in R^N$ (как это делается в случае параметризованного по весу нейрона), в схеме с взвешенным по входу нейроном: вектор входов высокой размерности $x \in R^N$ отображается в вектор низкой размерности $x^P \in R^M$. Это делается из соображений, что вектор может исчерпывающе описать наиболее важные характеристики своего прообраза более высокой размерности x. Если такое отображение существует для множества векторов x, то векторы весов можно использовать в их форме z меньшей размерности в сети вместо исходной формы вектора высокой размерности p.

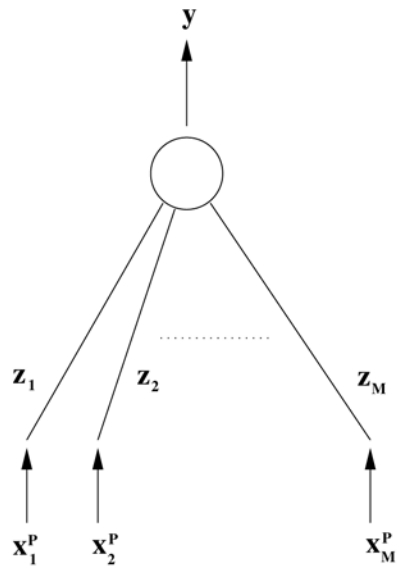


Рис. 3.1. Взвешенный по входу нейрон

Предположим, что существует отображение $\mathcal{P}$: $R^N \longrightarrow R^M$ такое, что $x^p = \rho(x) \in R^M$, где $M < N$. Операцию такого нейрона-модели можно определить следующим образом:

$$y_s = f_s(x, p) = f_s(x^P, z) \tag{3.1}$$

Схема параметризованного по входу нейрона показана на Рис.3.1.

Оба описанные выше типа нейронов позволяют проводить подбор оптимальных весов для увеличения эффективности работы в пространстве меньшей размерности. В выборе вида нейрона следует исходить из существования отображения $\mathcal{M}$ для весовых коэффициентов или $\mathcal{P}$ для входов: если пространство параметров ограничено подпространством меньшей размерности в пространстве высокой размерности, следует применять параметризованный по входу нейрон. Его использование также имеет смысл даже в случае, когда такое представление вектора весовых коэффициентов заранее не известно, а из распределения входного вектора или после применения метода главных компонент обнаруживаются признаки существования подходящего оператора с минимальной потерей информации, который производит отображение входного вектора в пространство меньшей размерности. Именно так дело обстоит с нейронной сетью для поиска контуров, которая производит отображение участка пикселей в приконтурном участке в векторы размерности 2, отражающие два доминирующих уровня яркости вокруг контура.

3.2.2. Расчёт выхода подсети

В выбранном подходе для расчёта выхода подсети вместо линейной комбинации выходов входящих в неё нейронов используется соревновательный процесс, победитель в котором определяется следующим образом:

$$p_{win} = \underset{s}{\operatorname{argmax}}(x, p_s), \text{ внутри } r\text{-й подсети} \tag{3.2}$$

где p_{win} - индекс нейрона-победителя.

Выход подсети $\phi(x, p_r)$ заменяется выходом нейрона-победителя.

$$\phi(x, p_r) = \phi(x, p_{win}) \tag{3.3}$$

С учётом этого соревновательного процесса, для вычисления выхода отдельного нейрона естественно использовать Эвклидово расстояние

$$\phi(x, p_r) = \phi(x, p_{win}) = \|x - p_{win}\| \tag{3.4}$$

Такой тип сетей хорошо подходит для задач классификации образов без наблюдателя, при которой каждый класс образов состоит из нескольких разобщённых наборов слабо отличающихся характеристик. Тогда каждый главный, первичный класс можно связать с подсетью, каждый производный внутри него - с нейроном этой подсети. Полученная топология сети и структура отдельно взятой подсети показана на Рис.3.2

В взвешенном по входу нейроне не производится включения вектора z_s в пространство более высокой размерности, а происходит отображение входного вектора $x \in R^N$ в подпространство меньшей размерности R^M с помощью оператора $\mathcal{P}$. Таким образом:

$$\phi(x, p_s) \equiv_r (x^P, z_s) \tag{3.5}$$

$$= \phi(\mathcal{P}(x), z_s) \tag{3.6}$$

где $x^P = \mathcal{P}(x)$ - входной вектор пониженной размерности, то есть образ входного вектора x в R^N.

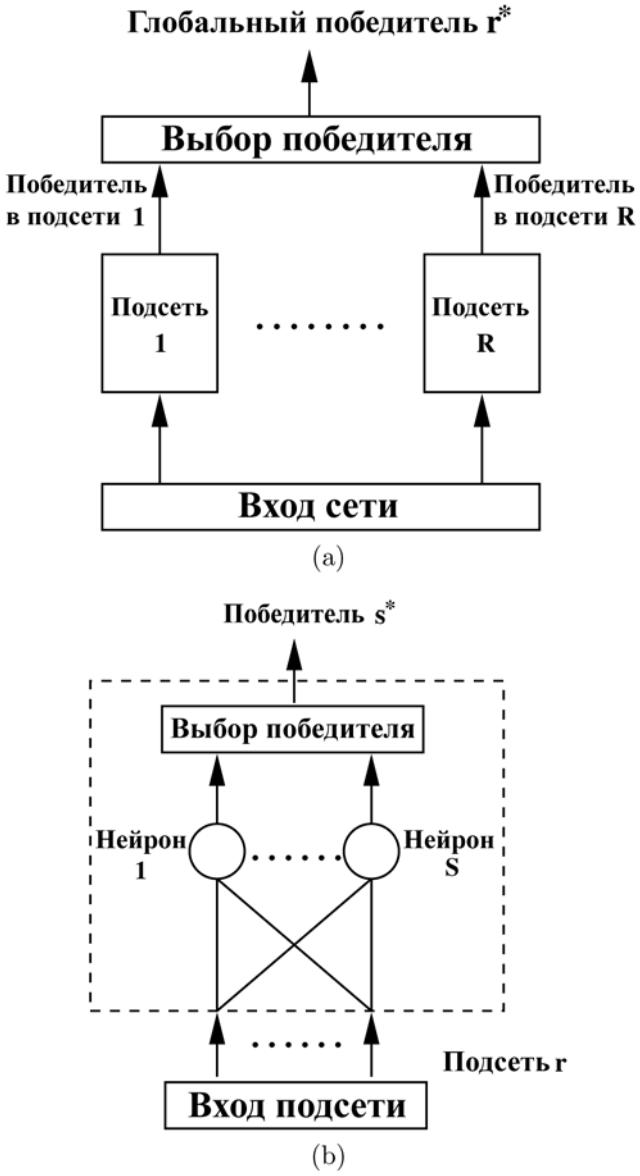


Рис. 3.2. Топология нейронной сети для выделения контуров: (a) архитектура сети, (b) архитектура подсети

3.2.3. Описание и выявление контуров

Выбор данной топологии сети основан на наблюдении различных предпочтений человека в отношении определённой величины различия между уровнями яркости в качестве границы при разных условиях освещённости. Чтобы включить этот критерий в процесс выделения контуров, целесообразнее использовать предложенную иерархическую топологию сети, в которой для представления различных уровней освещённости предназначены подсети, а нейроны внутри них - выполняют роль представления различных прототипов контуров при заданном подсетью уровне освещённости. Под прототипом контура понимается вектор размерности 2 $w \in R^2$ отражающий два доминирующих уровня яркости по обе стороны контура.

Так как прототипы создаются в процессе обучения на основе отобранных человеком образцов, следует применить схему состязательного обучения без наблюдателя, а каждый прототип представить в сети при помощи вектора весовых коэффициентов нейрона. Соревнование по правилу "победитель получает всё"также располагает к использованию иерархической топо-

логии с подклассами, внутри которых во время обучения только нейрон-победитель может обновлять свой вектор весов. Необходимо добавить, что выход нейрона внутри r-той подсети вычисляется как Эвклидово расстояние между прототипом и образцом участка контура.

$$_r(x^P, z_{s_r}) = \left\| x^P - z_{s_r} \right\| \tag{3.7}$$

где $x^P \in (R)^2$ текущий обрабатываемый участок, а z_{s_r} - s_r-тый прототип границы в рамках r-той подсети.

С учётом того факта, что отдельный участок контура обычно представлен набором значений уровня яркости $x \in (R)^N$, где $N \gg 2$, необходимо суммировать эти значения для выделения двух преобладающих уровней яркости. Иными словами, необходимо получить отображение $\mathcal{P}$: $R^N \to R^2$ такое, что

$$x^P = \mathcal{P}(x) \tag{3.8}$$

что будет соответствовать операции взвешенного по входу нейрона-модели.

3.3. Топология сети

Предложенная топология нейронной сети состоит из некоторого количества подсетей, каждый нейрон которых ответственен за усвоение определённого набора образцов из базы данных обучения сети. На начальной стадии обучения образцы обучающего набора адаптивно разделяются на подмножества в ходе не управляемого извне соревновательного процесса между подсетями. В последующей стадии - распознавании - пикселы контуров выявляются путём сравнения конфигурации текущего пиксела с прототипами, связанными с разными подсетями.

Применение иерархической топологии основано на наблюдении, что при выделении контуров более эффективнее использовать множество наборов пороговых параметров для принятия решения при различных условиях окружения, чем использовать один набор параметров для всего изображения, как это делается в других методах.

В предложенной модели представления каждая подсеть связана с шаблоном контура при определённом уровне фоновой освещённости, а каждый нейрон подсети соответствует одной из возможных вариаций контура при данной освещённости. Топология этой классифицирующей сети-детектора показана на Рис 3.3, а иерархическая схема представления прототипов описана в следующих разделах.

3.3.1. Представление информации о контурах

Выделение контуров проводится на $N \times N$-окрестности текущего пиксела. При конкатенации соответствующих значений уровней яркости в вектор $x = [x_1 ... x_{N^2}]^T \in R^N$ средний уровень яркости рассчитывается следующим образом:

$$\overline{x} = \frac{1}{N^2} \sum_{n=1}^{N^2} x_n \tag{3.9}$$

После задания этого среднего значения информация об уровнях яркости обрабатываемого участка дальше может быть представлена в виде вектора $m = [m_1 m_2]^T \in R^2$, компоненты

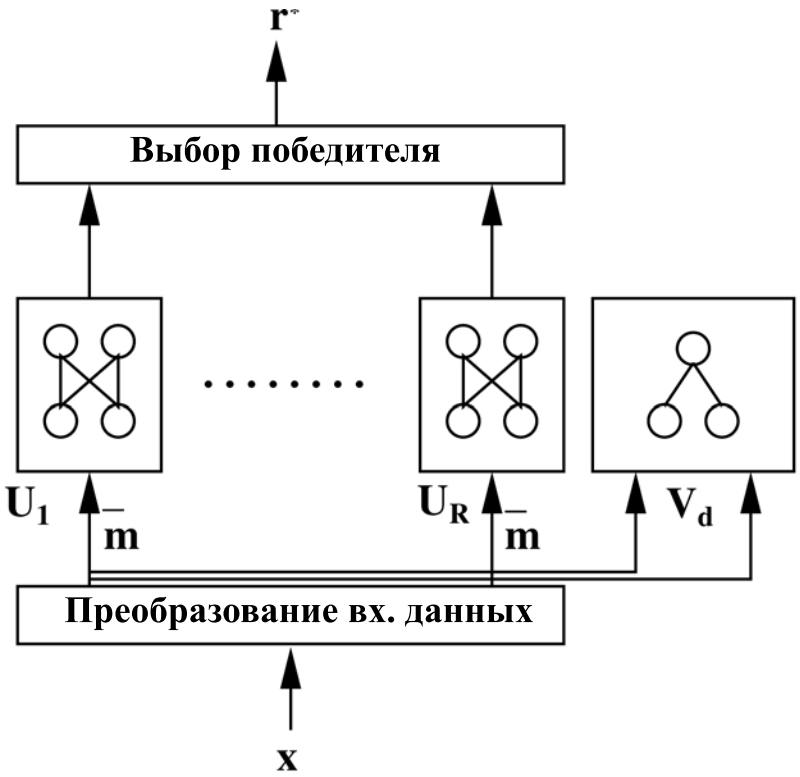


Рис. 3.3. Нейронная сеть для выделения контуров

которого соответствуют двум преобладающим уровням яркости внутри участка и определяются так:

$$m_1 = \frac{\sum_{i=1}^{N^2} I(x_i < \overline{x}) x_i}{\sum_{i=1}^{N^2} I(x_i < \overline{x})} \tag{3.10}$$

$$m_2 = \frac{\sum_{i=1}^{N^2} I(x_i \geq \overline{x}) x_i}{\sum_{i=1}^{N^2} I(x_i \geq \overline{x})} \tag{3.11}$$

$$\overline{m} = \frac{m_1 + m_2}{2} \tag{3.12}$$

где функция $I(\bullet)$ является функцией истинности, которая обращается в единицу, если её аргумент несёт логически истинное значение, и в ноль в противном случае.

3.3.2. Определение подсети-победителя

Как было описано ранее, каждая подсеть $U_r, r = 1, ..., R$ связана с прототипом значения уровня яркости фоновой освещённости p_r. Локальные участки $N \times N$ в изображении со средними значениями уровня яркости, ближайшими к p_r отправляются для дальнейшей обработки в подсеть U_r. Если говорить подробнее, участок пикселов $\mathcal{W}$ в изображении с его средними показателями яркости $\overline{m}, m_1, m_2$, полученными из уравнений (3.10) - (3.12), присваивается подсети $U_{r_{win}}$, если выполняются следующие условия:

$$p_{r_{win}} \in [m_1, m_2] \tag{3.13}$$

$$|\overline{m} - p_{r_{win}}| < |\overline{m} - p_r| \quad r = 1, ..., R, r \neq r_{win} \tag{3.14}$$

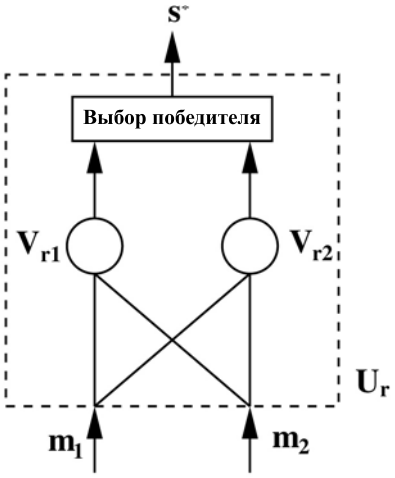


Рис. 3.4. Схема подсети u_R

где $[m_1, m_2]$ - закрытый интервал с границами m_1, m_2. Множество выделенных участков $N \times N$, таким образом, разбивается на подмножества, элементы которых представляют собой различные уровни освещённости. Для удобства обозначим два условия (3.13) и (3.14) вместе как $x \to U_{r_{win}}$. Архитектура подсети показана на Рис.3.4.

3.3.3. Определение нейрона-победителя в подсети

Каждая подсеть U_r состоит из S нейронов $V_{rs}, s = 1, ..., S$, которые являются шаблонами контуров, представляя возможные вариации контуров при среднем общем уровне освещённости p_r. Каждый нейрон связан с весовым вектором $w_{rs} = [w_{rs,1} w_{rs,2}]^T \in R^2$, который объединяет два преобладающих уровня яркости в каждом участке $N \times N$ пикселов $\mathcal{W}$ в форме вектора-прототипа m. Участок с принадлежащим ему вектором m присваивается нейрону $V_{r_{win} s_{win}}$, если выполняется следующее условие:

$$\|m - w_{r_{win} s_{win}}\| < \|m - w_{rs}\| \quad s = 1, ..., S, s \neq s_{win} \tag{3.15}$$

При S, равном 2, каждая подсеть состоит из двух нейронов, из которых один представляет собой прототип слабо выраженного контура, другой - прототип явно выраженного контура. Соответственно, один из весовых векторов $w_{r_{win} s}, s = 1, 2$ становится прототипом слабого контура $w^l_{r_{win}}$, а второй - прототипом сильного контура $w^u_{r_{win}}$. К какому виду относится нейрон, определяется согласно следующему критерию:

$$w^l_{r_{win}} = w_{rs'}, \text{где} s' = \text{argmin}_s(w_{r_{win} s,2} - w_{r_{win} s,1})$$
$$w^u_{r_{win}} = w_{rs''}, \text{где} s'' = \text{argmax}_s(w_{r_{win} s,2} - w_{r_{win} s,1})$$

После назначения прототипу слабого контура весового вектора $w^l_{r_{win}} = [w^l_{r_{win},1} w^l_{r_{win},2}]^T$, мера $(w^l_{r_{win},2} - w^l_{r_{win},1})$ играет роль порогового параметра в общепринятых алгоритмах выделения контура в определении нижней границы видимости контура и полезна для нахождения потенциальных начальных точек контура в изображении для его отслеживания. Структура нейрона показана на Рис.3.5.

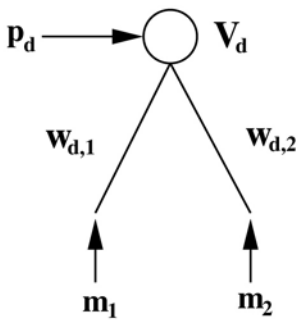


Рис. 3.5. Нейрон динамического отслеживания контура

3.3.4. Нейрон динамического отслеживания

В дополнение к подсетям U_r и локальным нейронам V_{rs} в топологию сети вводится нейрон динамического отслеживания контура V_d. Другими словами, этот нейрон находится вне подсетей, в глобальном пространстве сети.

Динамический нейрон является гибридом подсети и отдельного нейрона, который содержит и динамический весовой вектор $w_d = [w_{d,1} w_{d,2}]^T \in R^2$, соответствующий весовому вектору отдельного нейрона, и скалярный параметр p_d, аналогичный индикатору уровня освещённости подсети. Структура динамического нейрона отслеживания показана на Рис.(3.5).

Задачей этого нейрона является слежение за изменением уровня яркости фона во время отслеживания контура. На стадии обучения нейрон неактивен. На стадии распознавания и выделения явных точек контура динамический весовой вектор $w_d = [w_{d,1} w_{d,2}]^T \in R^2$ и индикатор уровня освещения фона p_d постоянно изменяются, чтобы отследить менее явные точки контура, связанные с начальной его точкой, то есть динамический нейрон непосредственно проходит стадию обучения в процессе обработки участка контура в изображении.

3.3.5. Двоичное представление контура

Предположим, что вектор m текущего участка $\mathcal{W}$ привязан к нейрону $V_{r_{win} s_{win}}$ с весовым вектором $w_{r_{win} s_{win}}$. Для выделения контуров целочисленный вектор $x \in R^{N^2}$ представляет значения уровней яркости пикселов в текущем участке и отображается в двоичный вектор $b \in B^{N^2}$, где $B = \{0, 1\}$. Чтобы добиться этого, определим операцию отображения $\mathcal{Q}$: $R^{N^2} \times R^2 \to B^{N^2}$ следующим образом:

$$b = \mathcal{Q}(x, w_{r_{win} s_{win}}) = [q(x_1, w_{r_{win} s_{win}}) \quad \dots \quad q(x_{N^2}, w_{r_{win} s_{win}})]^T \in B^{N^2} \tag{3.16}$$

где входящие в него отображения q: $R \times R^2 \to B$ заданы как

$$q(x_n, w_{r_{win} s_{win}}) = \begin{cases} 0, & \text{если } |x_n - w_{r_{win} s_{win},1}| < |x_n - w_{r_{win} s_{win},2}| \\ 1, & \text{если } |x_n - w_{r_{win} s_{win},1}| \geq |x_n - w_{r_{win} s_{win},2}| \end{cases} \tag{3.17}$$

Для верных конфигураций контура бинарные векторы b принимают фиксированные формы, показанные на Рис.(3.6) для $N = 3$. На начальной стадии обучения конфигурация контура, связанная с каждым образом контура, записанная в двоичной форме, хранится в наборе конфигураций контуров C, который является частью набора параметров самой сети. Множество C в дальнейшем расширяется, потому что при $N = 3$ оно замкнуто относительно операции $R_{\frac{\pi}{4}}$. В частности, эта операция является такой перестановкой над вектором $b \in B^9$,

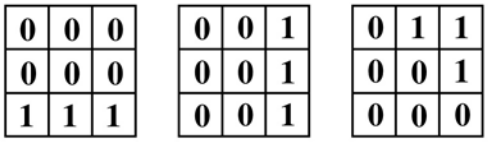


Рис. 3.6. Примеры конфигураций краёв

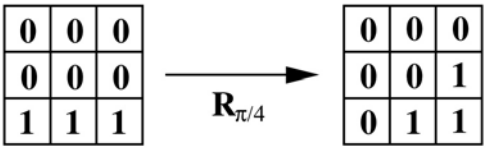


Рис. 3.7. Операция поворота образца контура на $\frac{\pi}{4}$

что, будучи представленной на матрице размером 3×3, является поворотом на $45°$ по часовой стрелке b. Это показано на иллюстрации Рис.3.7. Таким образом, выделение центральных поворотов конфигураций контуров, которые не были представлены на стадии обучения, облегчено.

3.3.6. Сравнение с топологией классической нейронной сети

Представленная нейронная сеть, выделяющая контуры, находится в прямом соответствии с обобщённой топологией нейронной сети прямого распространения, которое выражается в следующем: подсети $U_r, r = 1, \ldots, R$ тождественны подсетям в классической модели, а нейроны $V_{rs_r}, s_r = 1, \ldots, S_r$ соответствуют нейронам, входящим в подсети классической архитектуры, в данном случае, числом $S_r = 2$ на каждую подсеть.

Вместо включения весового вектора малой размерности в пространство высокой размерности, как делается в случае с параметризованным по весу нейроном, вектор входных данных x относительно высокой размерности отображается в вектор меньшей размерности m, содержащий информацию о двух доминирующих уровнях яркости в рассматриваемом участке. Другими словами, использовался взвешенный по входным данным нейрон-модель, описанный раньше, с функцией отображения $\mathcal{P}$: $R^{N^2} \to R^2$, заданной таким образом, что

$$m = \mathcal{P}(x) \tag{3.18}$$

Наряду с изложенными выше соответствиями, есть ряд небольших различий между исходной топологией классической нейронной сети с иерархической архитектурой с подразделами и представленной сетью, выделяющей контуры. В исходной архитектуре выходные данные подсети полностью заменяются выходными данными принадлежащего ей нейрона-победителя следующим образом:

$$\phi(x, w_r) \equiv_r (x, w_{s_{win}}) \tag{3.19}$$

где s - индекс локального победителя, а $_r(x, w_{s_{win}})$ его выходные данные. В данной сети соревновательный процесс на уровне подсетей независим от соревновательного процесса на уровне отдельных нейронов и имеет совсем другую природу: соревнование между подсетями происходит согласно соответствию в показателях текущего локального уровня яркости освещённости фона и уровня освещённости фона прототипа, что обозначает сравнение скалярных

значений. Соревнование же на уровне нейронов состоит в сравнении векторов размерности 2 - прототипов контуров при конкретном уровне яркости фона. В результате, выходные данные подсетей и нейронов независимы и иерархически находятся на двух разных уровнях. На уровне подсетей выход формулируется через показатели яркости фона:

$$\phi(\overline{m}, p_r) = |\overline{m} - p_r| \tag{3.20}$$

На уровне нейронов локальный выход формулируются через векторы-прототипы контуров:

$$_r(m, w_{rs}) = \|\overline{m} - w_{rs}\| \tag{3.21}$$

3.4. Обучение

Обучение состоит из трёх стадий: на первой стадии в соревновательном процессе задаются прототипы средней фоновой освещённости $p_r, r = 1, \ldots, R$ для каждой подсети U_r. На второй стадии каждый участок $\mathcal{W}$ приписывается своей подсети, и локальному нейрону этой подсети на основании его параметров p_r и m. Весовой вектор w_{rs} нейрона-победителя настраивается в соревновательном процессе обучения. На третьей стадии на основании отображения $\mathcal{Q}$ формируется соответствующая конфигурация двоичного образа b как функция весового вектора победителя w_{rs}. Далее к b применяется операция $R_{\frac{\pi}{4}}$ к b, чтобы получить восемь центральных поворотов образа, которые сохраняются в набор конфигураций C. Размер стороны обрабатываемого участка N принимается 3,

3.4.1. Обучение подсети-победителя

Допустим, что текущий обрабатываемый участок с его средним показателем p_r приписан подсети $U_{r_{win}}$ на основании условий (3.13) и (3.14). Затем значение $p_{r_{win}}$ модифицируется в соревновательном обучении следующим образом:

$$p_{r_{win}}(t+1) = p_{r_{win}}(t) + \eta(t)(\overline{m} - p_{r_{win}}(t)) \tag{3.22}$$

Корректирующий множитель $\eta(t)$ последовательно уменьшается по следующему правилу:

$$\eta(t+1) = \eta(0)\left(1 - \frac{t}{t_f}\right) \tag{3.23}$$

где t_f - общее количество итераций обучения.

3.4.2. Изменение весового вектора нейрона-победителя

Допустим, что настоящий участок изображения с его вектором параметров m приписан локальному нейрону $V_{r_{win}s_{win}}$ подсети $U_{r_{win}}$, его весовой вектор $w_{r_{win}s_{win}}$ нейрона снова корректируется в соревновательном обучении следующим образом:

$$w_{r_{win}s_{win}}(t+1) = w_{r_{win}s_{win}}(t) + \eta(t)(m - w_{r_{win}s_{win}}(t)) \tag{3.24}$$

причём множитель $\eta(t)$ последовательно уменьшается согласно уравнению (3.23).

3.4.3. Получение подходящих конфигураций контуров

По окончании предыдущих стадий, индикаторы фонового уровня освещённости p_r подсетей и весовые векторы w_{rs} для нейронов определены. В результате, все участки изображения размером $N \times N$ могут быть распределены по подходящим подсетям и соответствующим нейронам подсетей согласно их параметрам $\overline{m}$ и m. Если текущий участок приписан нейрону $V_{r_{win}s_{win}}$ внутри подсети $U_{r_{win}}$, вектор уровней яркости x этого участка $\mathcal{W}$ может быть переведён в бинарный вектор b конфигурации контура как функция $w_{r_{win}s_{win}}$ согласно уравнению (3.17):

$$b = \mathcal{Q}(x, w_{r_{win}s_{win}}) \tag{3.25}$$

Двоичный вектор b добавляется во множество допустимых конфигураций контура C. При работе с размером стороны участка $N = 3$, условие замкнутости множества C по операции $R_{\frac{\pi}{4}}$ может быть выполнено, если сгенерировать восемь конфигураций контура $b_j, j = 0, \ldots, 7$, используя операцию поворота $R_{\frac{\pi}{4}}$ следующим образом:

$$b_0 = b \tag{3.26}$$

$$b_{j+1} = R_{\frac{\pi}{4}}(b_j) \quad j = 0, \ldots, 6 \tag{3.27}$$

и сохраняя их во множестве конфигураций C.

3.5. Выделение контуров

На этой стадии все $N \times N$ участки тестового изображения проверяются на наличие характеристик контура. Эта стадия распознавания состоит из двух этапов. На первом этапе все пикселы с высокой степенью сходства с усвоенными сетью прототипами контуров объявляются наиболее вероятными точками контуров. На втором этапе эти наиболее вероятные контурные точки используются, как начальные точки для операции отслеживания контура, которая рекурсивно проверяет менее явные контурные точки, определённые как вторичные.

3.5.1. Выявление первичных потенциальных контурных точек

На этом этапе все участки $N \times N$ тестового изображения $\mathcal{W}$ подвергаются проверке. Все участки, параметры которых $\overline{m}, m_1, m_2$ удовлетворяют следующим условиям, объявляются первичными контурными точками:

(A1).	$x \to U_{r_{win}}$	выполнение условий (3.13) и (3.14)
(A2).	$m_2 - m_1 \geq w^l_{r_{win},2} - w^l_{r_{win},1}$,	где $w^l_{r_{win}}$ - вектор-прототип слабо видимого контура $U_{r_{win}}$
(A3).	$b = \mathcal{Q}(x, w_{r_{win}s_{win}}) \in C$,	где $w_{r_{win}s_{win}}$ - весовой вектор, связанный с нейроном $V_{r_{win}s_{win}}$

Условие (A1) означает, что средний уровень яркости текущего участка должен быть близок к показателю одной из запроектированных подсетей. Условие (A2) подтверждает, что значительность границы, выраженная разницей между m_2 и m_1, больше, чем разница между компонентами прототипа слабо видимого контура. Условие (A3) проверяет что двоичный образ контура данного участка - одна из допустимых конфигураций множества C.

3.5.2. Выявление вторичных контурных точек

На втором этапе активируется нейрон динамического отслеживания контура V_d, чтобы отследить вторичные контурные точки, связанные с текущей первичной точкой. Индикатор уровня яркости p_d и весовой вектор w_d нейрона инициализируются при помощи параметров обнаруженной первичной контурной точки $\overline{m}^p$ и $m^p = [m_1^p \quad m_2^p]^T$ следующим образом:

$$p_d(0) = \overline{m}^p \tag{3.28}$$

$$w_d(0) = m^p \tag{3.29}$$

После инициализации параметров динамического нейрона начинает работу рекурсивный алгоритм отслеживания контура, чтобы найти менее явно выраженные пикселы контура(вторичные точки), связанные с первичными точками контура, применяя следующий набор условий к каждому из 8 соседних участков выбранной первичной точки.

(B1). $b = \mathcal{Q}(x, w_d) \in C$

(B2). $p_d \in [m_1, m_2]$

Условие (B1) похоже на условие (A3) для распознавания первостепенных контурных точек, в то время, как условие (B2) является модификацией условия (3.13), составляющего часть набора необходимых условий для $x \to U_r$, подтверждающих, что средний уровень яркости потенциальной вторичной точки контура, представленный параметром p_d, близок к таковому предыдущей обработанной точки. В добавление к этому, никаких больше условий на силу контура в текущем пикселе не налагается, чтобы позволить включить в окончательный контур слабо видимые участки, если они связаны с более явными точками контура.

Для каждой новой вторичной точки контура с соответствующими параметрами $\overline{m}^s$ и m^s, удовлетворяющей условиям, описанным выше, индикатор локального уровня яркости p_d динамического нейрона корректируется следующим образом:

$$p_d(t+1) = p_d(t) + \eta(t)(\overline{m}^s - p_d(t)) \tag{3.30}$$

В добавление к этому, если точка контура удовлетворяет условию (A2) детекции первичных точек контура, показывающему, что её значимость сопоставима с значимостью явной первичной точки контура, весовой вектор w_d динамического нейрона изменяется для регистрации характеристик текущей точки:

$$w_d(t+1) = w_d(t) + \eta(t)(m^s - w_d(t)) \tag{3.31}$$

3.6. Схема работы контурного фильтра

Работу контурного фильтра на основе разработанной нейронной сети можно алгоритмически описать в виде автомата, на вход которого поступают графические изображения, а результатом его работы являются карты контуров объектов на этих изображениях. Обучение нейронной сети представляет собой этап начальной калибровки автомата при помощи базы специально отобранных образцов эталонов, которая производится однократно и не требует повторения перед каждым циклом работы.

3.6.1. Входные данные

Алгоритм работы фильтра не зависит от формата, цветовой палитры и размеров предлагаемых для обработки изображений, так как фильтр обрабатывает изображения дискретно,

разбивая их на равные участки и используя только яркостную составляющую палитры изображений. Параметризация входных данных состоит в сокращении информации об участке от матрицы уровней яркости входящих в него пикселов до набора трех характеристических значений - среднего уровня яркости участка $\overline{m}$ и средних верхних и нижних значений яркости пикселов участка m_1 и m_2, получаемых из формул (3.10)-(3.12). Далее соревновательные процессы в сети будут зависеть только от функций этих значений.

3.6.2. Процесс выявления контуров

После разбиения обработанного входного изображения на участки, стадия выявления первичных участков контура состоит в том, что каждый участок с его рассчитанными характеристиками проходит полный путь по графу сети последовательно через слой подсетей, где выбор соответствующей характеристикам участка подсети представляет собой классификацию участка по среднему уровню освещённости, к слою нейронов выбранной подсети, где выбор победителя равнозначен принятию решения о значимости обнаруженных внутри участка различий в уровнях яркости. Этап соревнования подсетей моделирует поведение человека в решении задачи выявления границ объектов - при различных средних уровнях освещённости оценка значимости контура происходит с разными параметрами.

Как было описано выше, после выбора соответствующей участку подсети соревновательный процесс для принятия решения о контуре переносится на уровень пары нейронов подсети-победителя, чтобы в дальнейшем выход нейрона-победителя заменил выход подсети, к которой он принадлежит, в соответствии со схемой "победитель получает всё". Нейроны подсети представляют собой модели контуров при данном освещении, причём один моделирует явные различия в яркости внутри участка, то есть контуры объектов, нахождение которых и является сутью поставленной задачи, а другой - слабые различия, то есть те участки, на которых на этой стадии контур выявлен не будет(см. условие (A2)).

3.6.2..1 Выявление первичных участков контура

Таким образом выходом нейрона-победителя, а, следовательно, и подсети, к которой он принадлежит, становится либо карта контура полученная преобразованием $\mathcal{Q}$ над значениями яркости пикселов участка, либо пустая карта, в случае, если победителем оказался нейрон, моделирующий слабые различия в яркости в участке. На выход нейрона-победителя, если таковым оказался нейрон, моделирующий явные различия при данном уровне освещённости, налагается ограничение соответствия полученной карты контура одной из конфигураций контура, содержащейся в наборе возможных конфигураций контуров нейронной сети, сформированном в процессе обучения. Если это условие выполняется, контурная карта участка заносится в карту контуров изображения, частью которого он является.

3.6.2..2 Выявление вторичных участков контура

Как было показано в предыдущем разделе, участки изображения, внутри которых различия в яркости пикселов при их среднем уровне яркости принимаются за незначительные, не дают карты контура на выходе нейрона, к которому они были приписаны. Но предложенная модель сети способна учитывать то, что эти участки могут быть смежными с участками, на которых контур выявляется явно, и содержать его продолжение, яркость которого снижена изменившимся уровнем освещённости или же связанной с ним потерей информации при сжатии изображения. За выявление контура в таких ситуациях отвечает нейрон динамического отслеживания V_d, обрабатывающий окрестности участков, в которых на предыдущей стадии

был обнаружен контур объекта. Таким образом, по окончании стадии выявления первичных участков контура, наборы из 8 участков, окружающих участки изображения, на которых был выделен контур, проходят вторичную обработку с помощью нейрона V_d, стоящему в топологии сети вне всех подсетей и совмещающего в себе одновременно моделирующие свойства как нейрона, так и подсети. Нейрон динамического отслеживания контура инициализирует свои параметры на основании характеристик участка, окрестность которого ему предстоит обработать - среднего общего уровня яркости $\overline{m}$ и средних вехнего и нижнего уровней яркости m_1 и m_2. Далее на участках окрестности он проводит операцию выявления контура с менее строгим набором условий (B1), (B2), совмещённую с коротким циклом обучения последовательно корректирующим его прототип среднего значения яркости участка, а если в окресности встречается область с признаками первичного участка контура, то динамический нейрон корректирует также свои пороговые значения, от которых зависит результирующая в обрабатываемых участках карта контура.

3.6.2..3 Результат

По завершении обеих стадий результатом обработки является изображение, соответствующее по размеру представленному на входе, монохромное, где контуры объектов, выявленные фильтром, выделены выбранным цветом. Для корректности разбиения на участки, входное изображение, в случае необходимости, должно быть увеличено, чтобы его размеры были кратны размерам участка, и результирующее изображение потребует обратного преобразования, что следует выполнить до перевода матрицы занчений яркости в графический формат.

3.6.3. Обучение

Обучение сети в последовательности действий равноценно многократной обработке входного изображения сетью с предустановленными на начальном цикле параметрами её подсетей и нейронов, с тем лишь схематическим отличием от операции выявления контура, что в нём не участвует динамический нейрон, и, следовательно, распознавание проходит в одну стадию. При этом каждый раз, когда выбирается победитель среди подсетей или нейронов, их параметры корректируются согласно (3.22) и (3.24), достигая, таким образом, оптимальных значений в зависимости от базы данных обучения. Результаты обработки сравниваются наблюдателем с образцами контуров вплоть до получения удовлетворительной оценки. Чтобы сеть сформировала способность абстрагироваться от обучающего набора изображений, следует обучать сеть с помощью наибольшего возможного количества различных по своим характеристикам изображений, что является трудоёмкой задачей. Полученный в результате обучения набор параметров элементов сети следует сохранить и использовать для инициализации параметров сети перед началом её работы.

Предустановку параметров подсетей перед обучением предполагается делать следующим образом: точки на шкале серого, соответствующие прототипам уровней яркости подсетей расположить равномерно от начала до конца шкалы, установив количество подсетей в соответствии с желаемым уровнем подробности выявления контуров и располагаемыми вычислительными ресурсами, которые будут использоваться фильтром. Пороговые параметры нейронов следует установить нулевыми, и они будут инициализированы набором значений первого приписанного нейрону обрабатываемого участка изображения согласно выражению (3.23) для корректирующего множителя. Если по окончании обучения в сети останутся элементы, ни разу не задействованные в процессе обучения, их целесообразнее будет удалить, так как их использование будет сопряжено с потерей машинного времени и ресурсов, не да-

вая при этом желаемого и предсказуемого результата, или же они будут усложнять принятие решения в соревновательном процессе, участвуя в нём лишь номинально.

Глава 4.

Программная реализация

Для реализации программного комплекса был выбран язык C++, так как он предоставляет хорошие возможности как по части математических вычислений, так и по части представления объектной модели. Как было описано ранее, предложенная топология нейронной сети для выделения контуров в изображениях имеет свои структурные особенности и отличия от традиционных моделей нейронной сети. Эти выражается в использовании общей памяти для узлов сети, включающей в себя образцы формы контура, воспринятые сетью в процессе обучения, а также в добавлении цикла с динамическим нейроном, осуществляющим повторный анализ изображения с целью обнаружения менее явных контуров для того, чтобы уменьшить количество пропусков и дефектов в замкнутых контуров и выявить связные контуры. Язык C++ позволяет удобно моделировать такую объектную структуру, предоставляя гибкость в определении количества узлов скрытого слоя, а иерархическая объектная модель даёт большие возможности для использования общей памяти.

Задача требует удобного и эффективного аппарата работы с изображениями. Так как непосредственно создание такого аппарата не входит в поставленные цели, потребовалось найти реализацию такого аппарата в составе существующих библиотек для создания приложений, причём он должен быть проверенным, совместимым с объектной моделью, не требующим добавления сложного интерфейса для взаимодействия с ним. Также эти средства разработки должны распространяться свободно хотя бы для образовательных целей и быть доступными и простыми для понимания. Так как работа с изображениями требует определённой наглядности, а процесс обучения нейронной сети достаточно сложен в настройке параметров и многократном использовании, также было предпочтительно оснастить приложение графическим пользовательским интерфейсом. Это последнее требование представляет собой наибольшую сложность, так как желательно было бы избежать зависимости от конкретной операционной системы для приложения исследовательско-экспериментального назначения.

Всем вышеперечисленным требованиям из рассмотренных вариантов соответствует набор библиотек для разработки платформонезависимых приложений Qt4 фирмы Trolltech. Это обширная система библиотек , предоставляющий поддержку большого числа операционных систем: Microsoft Windows(95/98/NT/2000/XP/Vista), Mac OS X, Linux, Solaris, AIX, Irix, NetBSD, OpenBSD, HP-UX, FreeBSD и другие клоны UNIX с графической подсистемой X11. Qt использует интерфейс программирования приложений низкого уровня, что позволяет приложениям работать так же эффективно, как разработанным в конкретной среде. Сборка программы на иной платформе, чем та, на которой она была создана, не требует внесения в код изменений, а только перекомпиляции. Кроме платформонезависимости как таковой, библиотека Qt предоставляет обширный набор современных средств для создания приложений, работающих с графикой, базами данных, XML, различными протоколами, то

есть готовый инструментарий, позволяющий программисту не решать снова и снова вопросы, которые не входят в сферу его целей и интересов. Библиотека Qt широко используется более чем 4000 компаниями по всему миру, Qt используется в среде для рабочего стола KDE, браузере Opera, в одном из программных продуктов фирмы Adobe. Для некоммерческого использования с условием программирования с открытым кодом фирма Trolltech предоставляет бесплатную полнофункциональную версию своей библиотеки.

Одной из важных особенностей библиотеки является её идейный подход, полностью основанный на объектно-ориентированном программировании, тогда как большинство средств разработки кроссплатформенных приложений не в состоянии предложить удобных средств взаимодействия объектов графического интерфейса между собой и с операционной системой. Её новая концепция ведения межобъектных коммуникаций, именуемая «сигналы и слоты», полностью заменяет былую и ненадёжную систему обратных вызовов. Эта концепция расширяет язык C++, не влияя на парадигму программирования создаваемого приложения, в то время как в большинстве библиотек и операционных систем эта проблема решается за счёт использования средств языка C, что может нарушить общий стиль программы, а в случае с ОС Windows является весьма сложным и неудобным.

Рис. 4.1. логотип Trolltech Qt Toolkit

Таким образом, для внешнего и графического интерфейса приложения, была выбрана библиотека разработки кроссплатформенных приложений Qt4.3.0, на момент создания приложения не самая последняя версия, но наиболее стабильная из проверенных с условием статического связывания, которое требовалось для упрощения переноса приложения на другие машины, на которых не установлена библиотека Qt. Основная часть программы, отвечающая непосредственно за нейронную сеть и реализованный с её помощью контурный детектор, была спроектирована и написана на чистом C++, без использования инструментов Qt, как самостоятельный вычислительный блок, что, возможно облегчит понимание и использование её с другими библиотеками.

4.1. Объектная модель

Как говорилось ранее в главе 3, используемая нейронная сеть имеет иерархическую структуру с совместно используемой узлами и слоями памятью и циклом повторной обработки. На рис. 4.2 показано, как работа нейроннной сети образует процедуру обработки изображения в

оттенках серого до карты контуров. Двигаясь сверху вниз по схеме, удобнее всего объяснять структуру и ход работы разработанного программного продукта.

4.1.1. Изображение в оттенках серого

В программе этот объект представлен классом $GreyscaleWindow$. Это класс-контейнер с дополнительными функциями для представления как изображения в целом, так и его участков для дискретной обработки нейронной сетью. Он включает в себя следующие компонентные данные:

- матрица неотрицательных целых чисел, представляющих уровни яркости каждого пиксела изображения
- общий средний арифметический уровень яркости всех пикселов изображения (только для участков $N \times N$)
- средний арифметический уровень яркости пикселов изображения, не превосходящих средний уровень яркости (только для участков $N \times N$)
- средний арифметический уровень яркости пикселов изображения, превосходящих общий уровень яркости (только для участков $N \times N$)

Значения яркости пикселов используются для выявления контура в участках нейронами, последние три показателя - для прохождения по графу нейронной сети. Средние показатели яркости определяют подсеть и, далее, нейрон, которым участок будет передан на обработку.

Объекты класса обладают следующими основными методами:

- импорт изображения из цветовой модели RGB в представлении библиотеки Qt в оттенки серого(уровни яркости)
- экспорт карты контуров в монохромное изображение в представлении библиотеки Qt
- набор служебных функций для расчёта средних уровней яркости и доступа к эти показателям

Таким образом, можно видеть, что этот класс является связующим звеном между моделью представления изображений в используемой библиотеке и моделью данных, используемой нейронной сетью, выступая в структурном плане в качестве входного и выходного слоя нейронной сети.

4.1.2. Объекты сети

Как говорилось ранее, сеть использует общую память и методы проверки образцов участков контура. Чтобы реализовать доступ к общей памяти и методам, а также найти отражение общим структурным свойствам элементов сети в объектной модели программы, для всех элементов сети и был использован общий прототип со статическими компонентными данными и методами. Это абстрактный класс $EDNetworkMember$, который наследуют все классы, представляющие элементы сети. Использование статических компонентных данных позволяет осуществлять доступ к общей для всех объектов классов памяти, инициализируемой лишь один раз. Таким образом, достигнут известный уровень экономии ресурсов памяти и производительности, так как у объектов нет необходимости обмениваться большим количеством

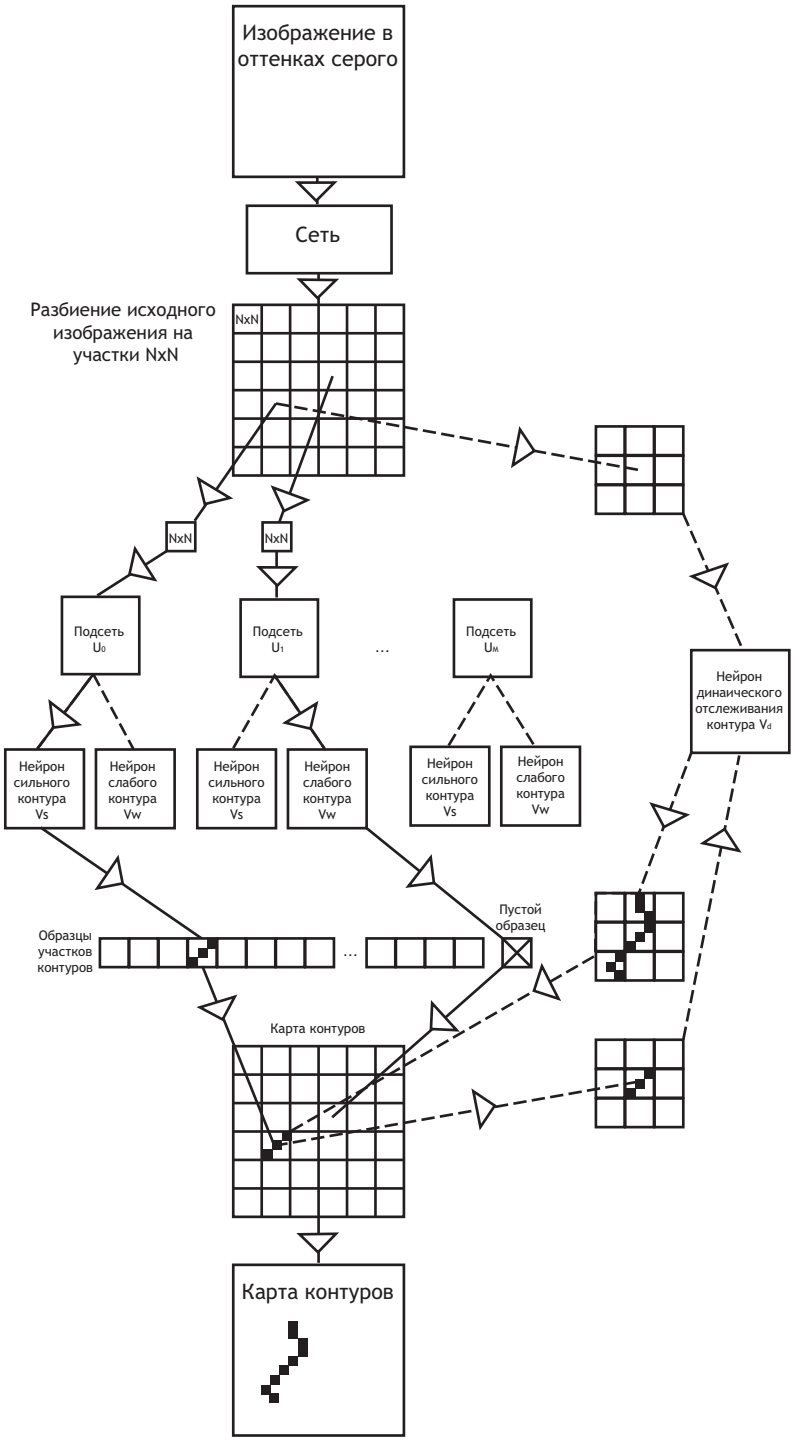


Рис. 4.2. Объектная модель программы

данных или лишний раз осуществлять взаимодействие между собой для получения доступа к инкапсулированным данным. Так же класс отвечает за формирование общей памяти образцов, делая это самостоятельно при проверках, в случае, если сеть находится в состоянии обучения.

Компонентные данные класса "Элемент сети":

- стек образцов участков контура, усвоенных сетью при обучении. Участок контура, не обнаруженный среди этих прототипов, не попадает на результирующую карту контура. Статический компонент

- образец пустого участка - для сравнений, для однообразия выдачи к финальному этапу формирования карты контура
- общее число итераций обучения - это число используется для вычисления корректирующего множителя при обучении подсетей и нейронов. При обучении вычисляется как общее количество участков $N \times N$ во всём наборе, предоставленном сети для обучения
- номер текущей итерации обучения

Основные методы класса:

- функция коррекции. Используется в обучении нейронов и подсетей, а также в работе динамического нейрона отслеживания контура, который в процессе обработки стека участков проходит короткие циклы обучения
- проверка наличия полученного образца участка контура в общей памяти C. Она же заполняет общую память образцов, используя предоставленный образец контура и 8 его центральных поворотов

4.1.3. Образец участка контура

Этот класс является основной единицей обмена информации между узлами сети, наряду с участком изображения. Это класс-контейнер, содержащий в себе результат выделения контура в виде его бинарной карты. Его функции - создание, хранение, передача и проверка участков контура размером $N \times N$. Он перенимает у нейрона функцию проекции участка изображения в участок контура $\mathcal{Q}$. Его основным компонентом является матрица пикселов контура, заполняемая значениями 0 или 1 применением операции $\mathcal{Q}$ над переданным объекту класса нейроном-победителем участком изображения и с использованием пороговых параметров этого нейрона. Также его компонентным методом является центральный поворот полученного участка контура для получения большего числа возможных комбинаций.

4.1.4. Нейрон

Класс *EDNeuron* отвечает за представление нейрона в сети. Его параметрами является весовой вектор нейрона, то есть верхнее и нижнее пороговое значение детектора. Сам по себе нейрон не несёт никакой информации о том, предназначен он для обработки слабого или сильного контура, а лишь обладает методом для обработки участка изображения, то есть соревновательный процесс между нейронами ведётся на уровне подсетей и их средствами. Не обладая никакими данными о том, обрабатывает ли он участок изображения, или же результат его работы будет опущен a priori, как незначительный, нейрон в любом случае участвует в процессе, так как от обучения его пороговых параметров зависит результат выделения контуров. Именно нейрон обновляет счётчик итераций обучения, так как он является последним узлом в обработке участка.

4.1.5. Подсеть

Подсети представлены в программе классом *EDSubnetwork*, содержащим в себе 2 динамически создаваемых нейрона, между которыми сеть проводит соревнование по показателям близости расстояний между пороговыми значениями и средних значений яркости обрабатываемого участка. При обучении сеть обновляет свой параметр среднего уровня яркости, не

инкрементируя счётчика итераций обучения. Выходом подсети служит указатель на образец участка в общей памяти, который выбрал нейрон-победитель, если он оказался нейроном сильного контура, и пустой образец в противном случае. Таким образом, выход подсети полностью замещается выходом нейрона, что соответствует концепции "победитель получает всё".

4.1.6. Нейрон динамического отслеживания контура

Этот отдельный нейрон, находящийся на уровне подсетей, представлен классом $DTNeuron$, и осуществляет обработку 8 участков окружения участка контура, переданных ему сетью после первой фазы выявления контуров. Схема его работы последовательно описана в предыдущей главе. Стоит лишь отметить, что объекты этого класса динамически создаются сетью для каждой операции по отслеживанию контура и инициализируются средними значениями участка, с которого начинается отслеживание. Как было описано выше, он объединяет в себе функции подсети и нейрона, что отражается в наличии у него всех трёх соответствующих этим классам параметров и особого набора условий обработки участков. Результаты его работы записываются в общую карту контуров только для тех участков окружения, в которых не было обнаружено признаков наличия контура на предыдущей стадии.

4.1.7. Сеть

Сеть является основным классом детектора, неся на себе ответственность как за общую организацию и проведение процесса выделения контуров, так и за низкоуровневые задачи по разбиению и обработке изображения. Сеть(класс $EDNetwork$) содержит массив подсетей и метод для начальной инициализации параметров всех узлов значениями, получаемыми симметричным разбиением всего диапазона яркости(оттенков серого в 8-битной палитре) по количеству узлов(см. рис.4.3). В его функции входит:

- изменение размеров изображения для возможности разбиения его на участки $N \times N$ и обратное этому преобразование
- осуществление соревновательного процесса между подсетями и запись результатов первой стадии выделения контуров
- составление стеков окружения участков контура для передачи их нейрону динамического отслеживания и запись результатов отслеживания
- загрузка и сохранение структуры сети и общей памяти в xml-файл

4.2. Структура файла настроек

Для сохранения параметров узлов и структуры сети был выбран наиболее популярный сегодня формат представления данных XML(Extensible Markup Language). Он является наиболее человекопонятным, простым и удобным в обработке и хранении данных. Для возобновления сеанса детектора требуется восстановить сеть и память образцов контура. При помощи инструментов библиотеки Qt в программе осуществляется быстрое сохранение этих данных в следующем виде:

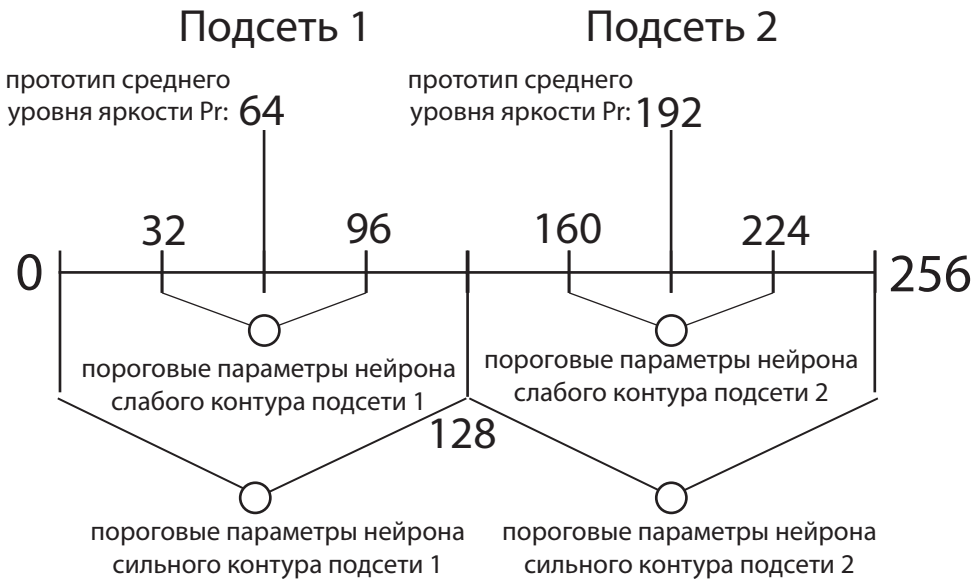


Рис. 4.3. Инициализация параметров сети с 2 подсетями

```
<network>
 <subnetwork illumination="8" >
  <neuron type="weak" >
   <upperthreshold>12</upperthreshold>
   <lowerthreshold>4</lowerthreshold>
  </neuron>
  <neuron type="strong" >
   <upperthreshold>16</upperthreshold>
   <lowerthreshold>0</lowerthreshold>
  </neuron>
 </subnetwork>
</network>
```

Для хранения данных из памяти образцов сети используется следующий формат:

```
<validedges>
 <edgemap>110011000</edgemap>
 <edgemap>011010001</edgemap>
 <edgemap>001011010</edgemap>
 <edgemap>000011101</edgemap>
 <edgemap>000000000</edgemap>
</validedges>
```

4.3. Графический интерфейс

Библиотека Qt предоставляет обширный спектр возможностей для создания графического пользовательского интерфейса для кроссплатформенных приложений. Они представлены в виде набора классов, соответствующих всевозможным элементам графической оболочки операционной системы и способам их организации и использования. Хотя они зачастую реа-

лизуются собственной графической подсистемой библиотеки, с их помощью можно создавать графические интерфейсы, обладающие характерным для выбранной операционной системы обликом и функциональностью. Экспериментальная программа была разработана в операционной системе Apple Mac OS 10.5.2, но может быть собрана в любой другой поддерживаемой библиотекой Qt4 операционной систем простой перекомпиляцией. Для наглядного представления архитектуры сети и работы контурного детектора, а также для удобной реализации процесса обучения на наборах изображений был разработан простой интерфейс, состоящий из одного главного окна, организованного в виде трёх логических разделов для каждой из перечисленных задач. В соответствии с последовательностью алгоритма программы, они расположены в следующем порядке:

- обучение сети
- параметры сети
- обработка изображений

4.3.1. Раздел «Обучение сети»

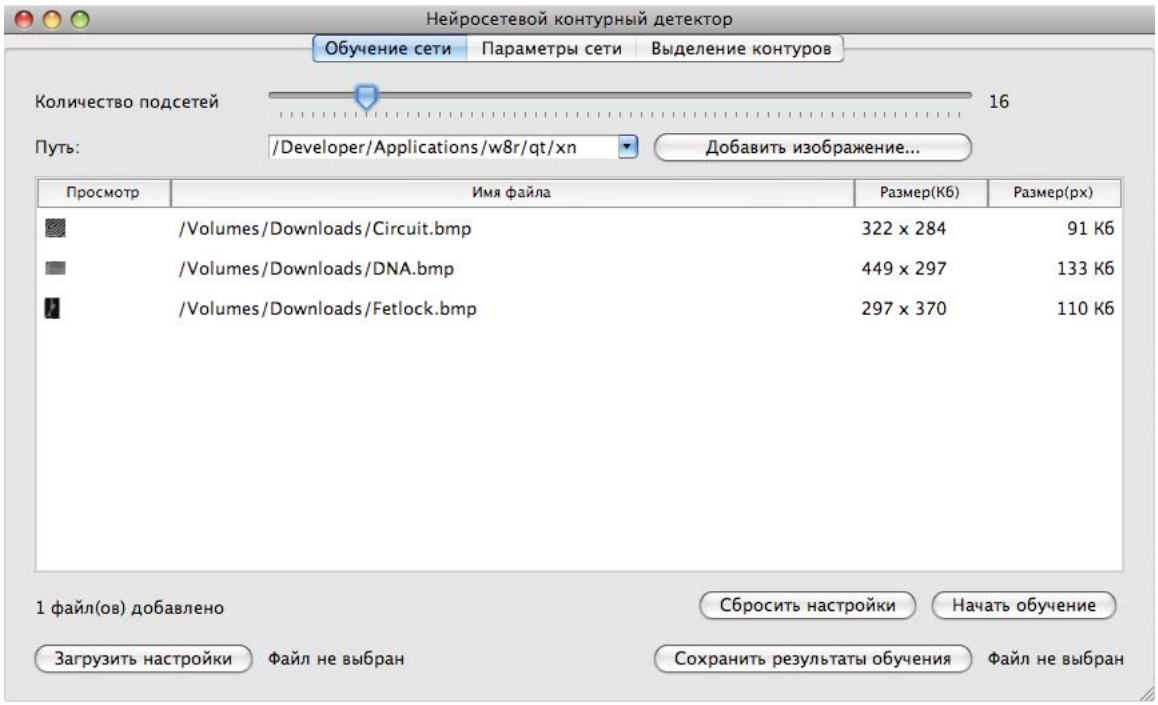
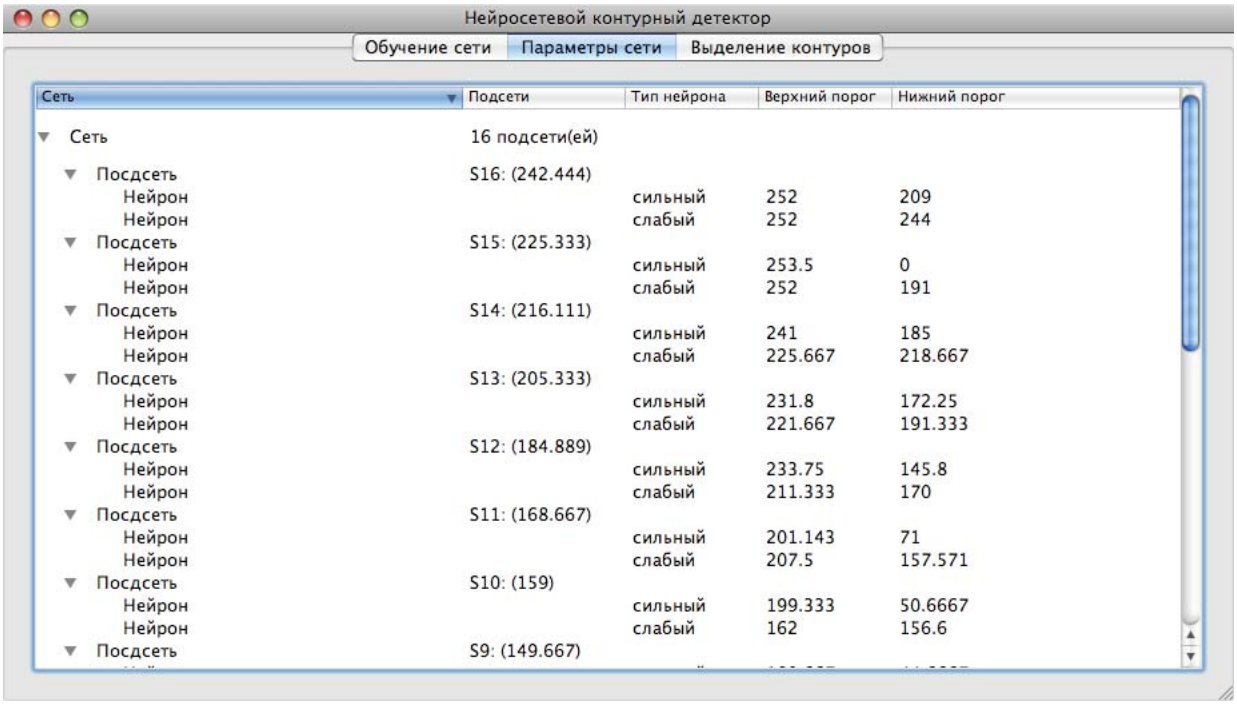


Рис. 4.4. Форма для обучения сети

На изображении 4.4 показана форма программы для обучения сети и загрузки/сохранения её параметров. С помощью верхнего ползунка можно установить желаемое количество подсетей нейронной сети. В следующей главе будет показано, как зависит от этого параметра результат обучения и работы детектора. Диапазон значений параметра указан на шкале. Ниже расположены инструменты для формирования обучающего набора изображений. Кнопка «Добавить изображение» позволяет добавить одно или несколько изображений в список под ней, выбрав их в диалоговом окне. Строка списка содержит в себе уменьшенную копию выбранного изображения, полный путь к нему, его размер в килобайтах и измерения.

В последнем разделе предыдущей главы было объяснено, что содержание и порядок образцов в обучающем наборе очень важны, поэтому изображения в список следует выбирать с особой тщательностью. Расположенные ниже кнопки соответственно сбрасывают существующие настройки сети или начинают процесс обучения. В обоих случаях существующая сеть уничтожается и заменяется новой, содержащей указанное ползунком на шкале число подсетей, которая инициализируется, как показано на рис. 4.3. Об окончании процесса обучения сообщает появление цифры, означающей количество обученных подсетей, под списком, который автоматически очищается. Результаты обучения можно сохранить, воспользовавшись кнопко «Сохранить результаты обучения». Параметры сети и память образцов участков контура можно загрузить в программу с помощью диалога выбора файла настроек, вызываемого кнопкой «Загрузить настройки».

4.3.2. Раздел «Параметры сети»



Сеть	Подсети	Тип нейрона	Верхний порог	Нижний порог
Сеть	16 подсети(ей)			
Посдсеть	S16: (242.444)			
Нейрон		сильный	252	209
Нейрон		слабый	252	244
Посдсеть	S15: (225.333)			
Нейрон		сильный	253.5	0
Нейрон		слабый	252	191
Посдсеть	S14: (216.111)			
Нейрон		сильный	241	185
Нейрон		слабый	225.667	218.667
Посдсеть	S13: (205.333)			
Нейрон		сильный	231.8	172.25
Нейрон		слабый	221.667	191.333
Посдсеть	S12: (184.889)			
Нейрон		сильный	233.75	145.8
Нейрон		слабый	211.333	170
Посдсеть	S11: (168.667)			
Нейрон		сильный	201.143	71
Нейрон		слабый	207.5	157.571
Посдсеть	S10: (159)			
Нейрон		сильный	199.333	50.6667
Нейрон		слабый	162	156.6
Посдсеть	S9: (149.667)			

Рис. 4.5. Форма представления параметров сети

На этой вкладке можно ознакомиться с текущими параметрами сети - количеством подсетей, их прототипами средней яркости, и пороговыми показателями нейронов. Древовидная таблица обновляется автоматически, так как её инициализация связана с получением сообщения об обновлении сети, которое порождается после обучения сети, сброса или загрузки параметров из файла.

4.3.3. Раздел «Обработка изображения»

После обучения сети или загрузки настроек можно выделить контуры на изображении. Для этого следует нажать кнопку «Открыть» (или воспользоваться комбинацией клавиш Ctrl + O) и выбрать в диалоговом окне желаемый файл. Изображение будет показано в

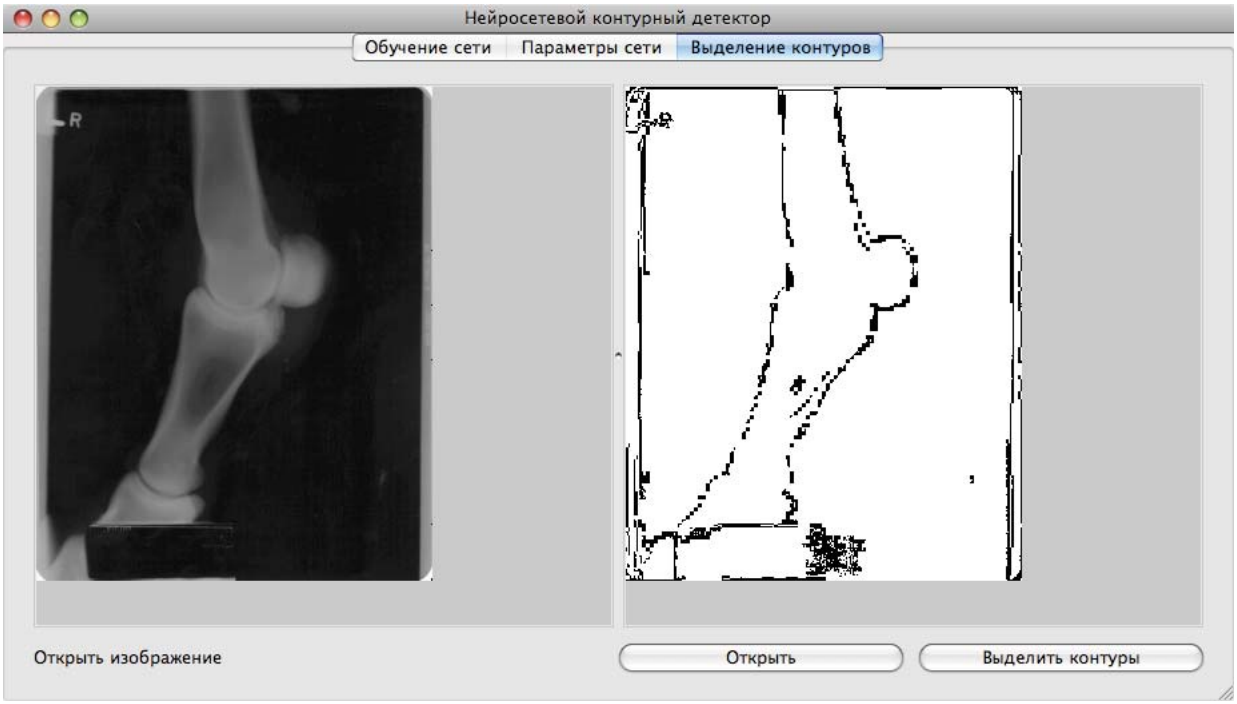


Рис. 4.6. Форма обработки изображения

левой части окна. После этого нужно нажать кнопку «Выделить контуры». Результат будет отображён в правой части окна программы. Для сохранения полученного результата можно воспользоваться командой «Файл>Сохранить контуры», или комбинацией клавиш Ctrl + D . Результат будет сохранён в выбранной директории под именем, заданном пользователем.

Глава 5.

Исследование эффективности разработанной системы

Оценить результаты работы контурного нейросетевого фильтра можно по нескольким критериям, которые можно разделить на две группы - абсолютные и относительные. К абсолютным критериям можно отнести такие, как неразрывность контура, точность локализации границ, помехоустойчивость, сложность настройки и производительность. Ко второй группее следует отнести сравнительную оценку этих характеристик и характеристик работы других фильтров.

Было проведено большое количество экспериментов по обучению и применению нейросетевого фильтра, результаты которых можно описать и оценить на следующих далее примерах.

5.1. Оценка работы нейросетевого контурного фильтра

5.1.1. Неразрывность контура

Фильтр показал хорошие результаты на изображениях разного вида, будучи обучен на небольшом количестве специально отобранных образцов, упорядоченных в наборе таким образом, чтобы можно было теоретически предсказать последовательность и качество усвоения данных нейронной сетью. Программа показала, что настройки сети, полученные в процессе обучения, хотя и позволяют сети абстрагироваться от базы данных обучения, но всё же сохраняют некоторую зависимость от неё. Быстрая сходимость обучения не позволяет сети усвоить образцы из сильно отличающихся классов, и даже если делать начальные циклы очень короткими, параметры, приобретённые сетью будут стремиться к обобщению, так как это заложено в самой структуре сети.

На рис.5.1-5.3 показано, что контур даёт удовлетворительные результаты сплошных контуров на изображении автомобильного номера, простого контрастного изображения, и даже сложного изображения с элементами шрифта, размытыми контурами и градиентными заливками. Тем не менее, явно видно, что в качество контура не во всех случаях удовлетворительно, он требует обобщения. Для сравнения можно взять результаты, полученные с помощью простого фильтра Собела(рис.5.4)(). На них видно, что, несмотря на лучшую форму контура, детектор Собела отбросил часть объектов, а в выделении штрихкода оказался очевидно грубее и хуже.

1.

2.

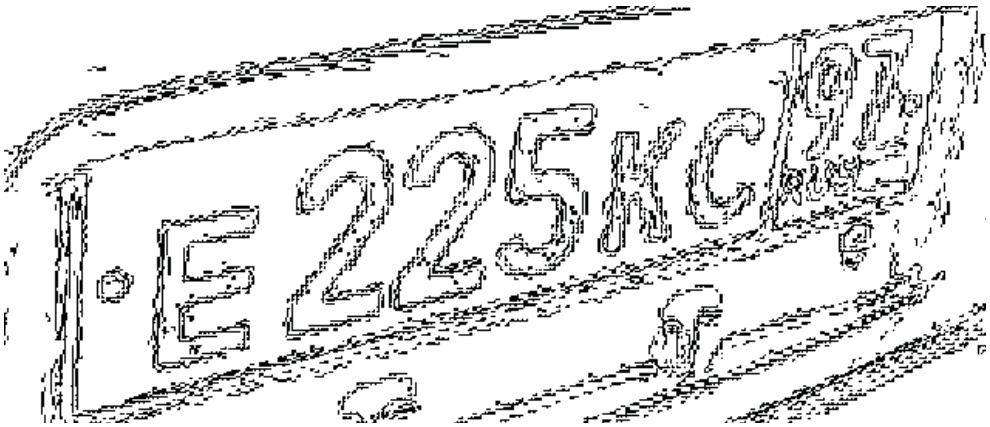


Рис. 5.1. Обработка изображения автомобильного номера нейросетевым детектором(16 прототипов освещённости)

5.1.2. Помехоустойчивость

Для экспериментов по оценке помехоустойчивости использовался образец клейма на металле, на изображении которого присутствовал сильный и яркий шум. Для сравнения тот же образец был обработан детекторами Собела и Кэнни. Сеть была обучена на образцах простых шрифтовых изображений для достижения наибольшего обобщения в её работе. Но результат работы детектора очевидно искажён по причине малого размера плавающего окна, что не позволяет ему обрабатывать участки с сильным шумом. При уменьшении количества подсетей, и, следовательно, чувствительности детектора, на результирующей карте пропадали значимые участки. Для получения образца, сравнимого с результатом работы детектора Кэнни, были произведены следующие манипуляции:

- чувствительность сети была уменьшена путём сокращения количества подсетей с 32 до 16
- было произведено переобучение, причём в обучающий набор был включён дубль текстового образца, обработанный фильтром Гаусса с $\sigma = 1.3$
- обрабатываемый образец был также подвергнут сглаживанию по Гауссу с $\sigma = 1.3$ для подавления шума(для детектора Кэнни использовался тот же коэффициент сглаживания)

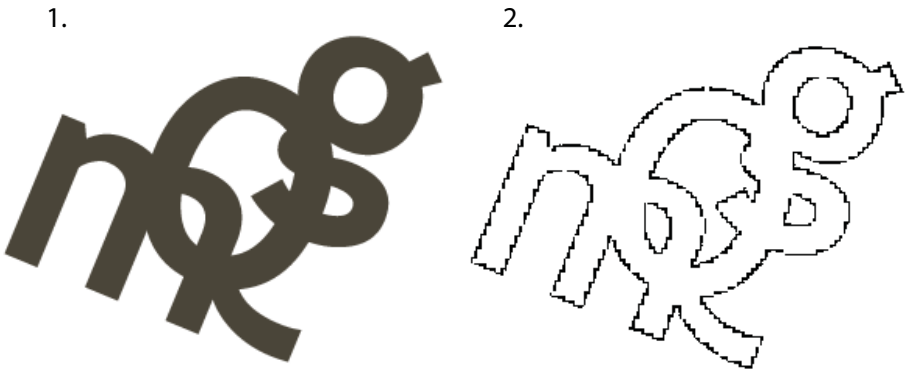


Рис. 5.2. Обработка изображения простых объектов(1) нейросетевым детектором(16 прототипов освещённости)(2)

Такую же динамику можно наблюдать на обработке образца на рис.5.7.

Но контуры, по-прежнему, остаются неудовлетворительными по сравнению с таковыми на результатах работы детектора Кэнни, хотя очевидно превосходят по качеству образцы, полученные с применением оператора Собела и от образца без сглаживания. Из этого явно следует, что для улучшения шумоподавления и формы контура требуется внести непринципиальные изменения в структуру детектора, а также использовать специальные технологии обучения и предварительное сглаживание в классе изображений, требующих этого. Показательным является то, что после переобучения и настройки детектора на обработку такого класса сильно зашумлённых изображений результаты обработки простых изображений остались такими же, как на рис.5.1-5.3, то есть при разработке эффективной технологии обучения детектора для применения к более сложным образцам его функциональность в отношении других задач вполне возможно сохранить.

5.1.3. Точность локализации границ

Предыдущие эксперименты наглядно показывают, что точность локализации границ нейросетевого детектора уступает этой характеристике представленных традиционных детекторов лишь по причине неудовлетворительной формы контура. И хотя пробы подтвердили, что работа нейрона динамического отслеживания контура вносит существенные улучшения в форме и неразрывности контура, слабым местом детектора остаётся малый размер плавающего окна и строгость манипуляций с образцами контуров в памяти детектора.

5.1.4. Производительность

При написании экспериментальной программы наглядность алгоритма имела приоритет над производительностью, что отразилось в существенном объёме оперативной памяти, используемом детектором для хранения изображений и участков, а также в большом количестве операций с массивами и динамически распределяемой памятью. Но, тем не менее, программа показала хорошие результаты в скорости обработки образцов и обучении. Причиной этого является малое количество вычислительных операций и их явная простота. Эта черта является явным преимуществом применения нейросетевой технологии, и если внести некоторые изменения, оптимизирующие код на низком уровне обработки матриц значений яркости и

Рис. 5.3. Обработка сложного изображения с различными конфигурациями контура и освещённости(1) нейросетевым детектором(16 прототипов освещённости)(2)

ввода-вывода файлов, экономичность в отношении ресурсов памяти и призводительности созданного детектора может стать его отличительной чертой на фоне традиционных методов.

5.1.5. Сложность настройки

Настройка и обучение детектора является задачей зачастую интуитивного характера, поэтому весьма сложно прогнозируемой. Несмотря на кажущуюся простоту(сеть обучается на предоставленных образцах «без учителя»), достичь нужного результата можно только после множества испытаний, обладая при этом подробными знаниями об алгоритме работы детектора. Но интерактивность процесса обучения делает его более привлекательным по сравнению с подбором параметров для математических фильтров. Но главной чертой интеллектуальной системы является то, что он может быть настроен на достаточно широкий класс задач и не будет требовать переобучения, работая в рамках этого класса. Это позволяет сделать предположение о том, что сложность настройки будет компенсирована простотой в применении, особенно после изучения предметной области и составления алгоритмов обучения и наборов параметров для её разделов.

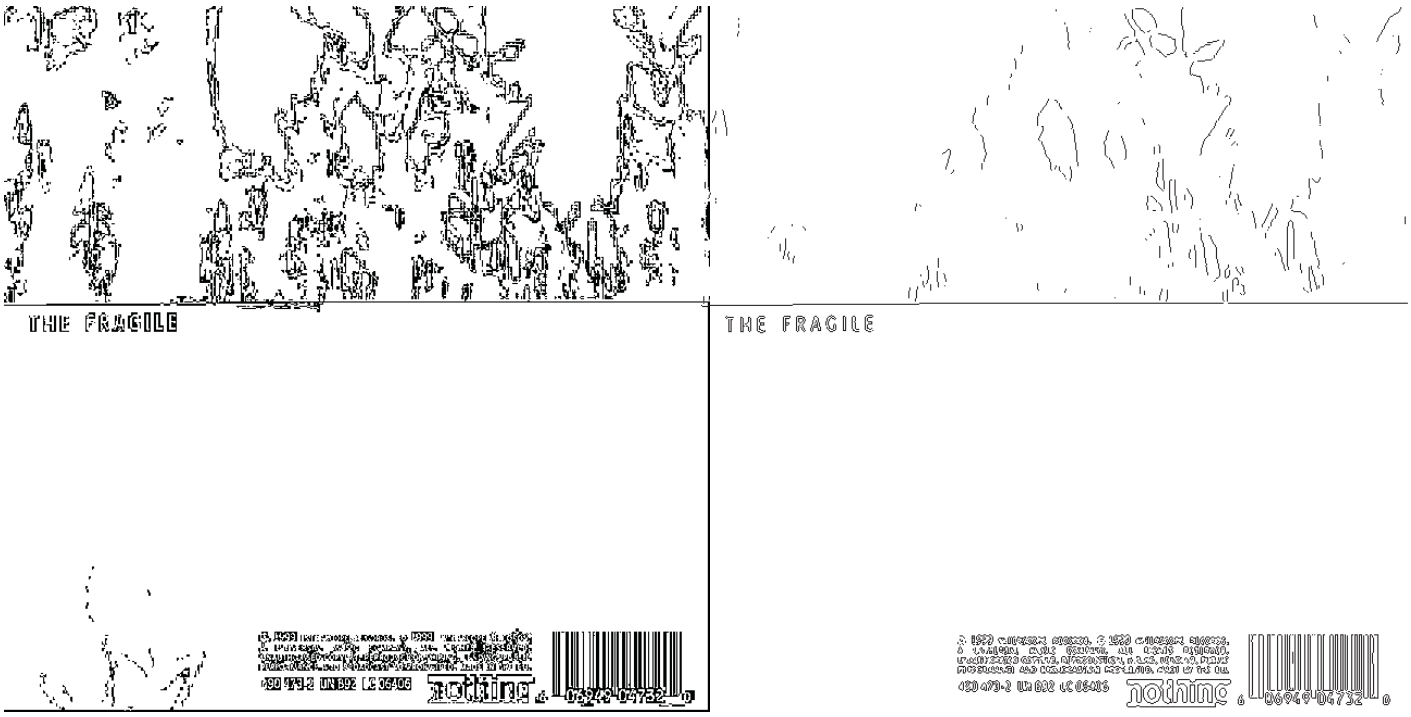


Рис. 5.4. Обработка сложного изображения нейросетевым детектором(16 прототипов освещённости)(1) и детектором Собела(двунаправленным, с автоматическим подбором порога)(2)

Рис. 5.5. Обработка зашумлённого изображения клейма(1) нейросетевым детектором(2)

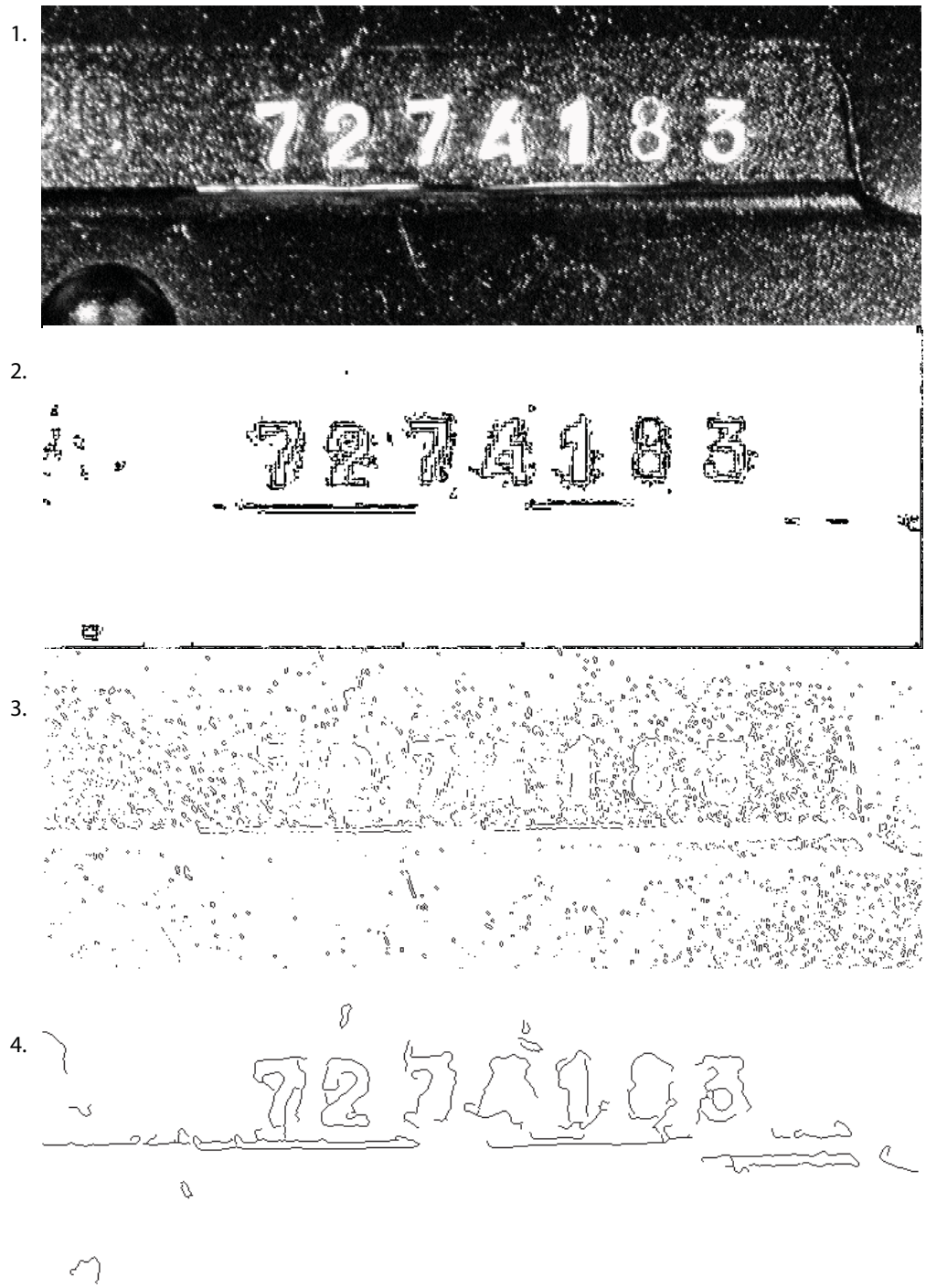


Рис. 5.6. Обработка зашумлённого изображения клейма(1) нейросетевым детектором с предварительным сглаживанием (2), детектором Собела(двунаправленным, с автоматическим подбором порога)(3) и детектором Кэнни(4)($\sigma = 1.4, t_1 = 0.04, t_2 = 0.10$)

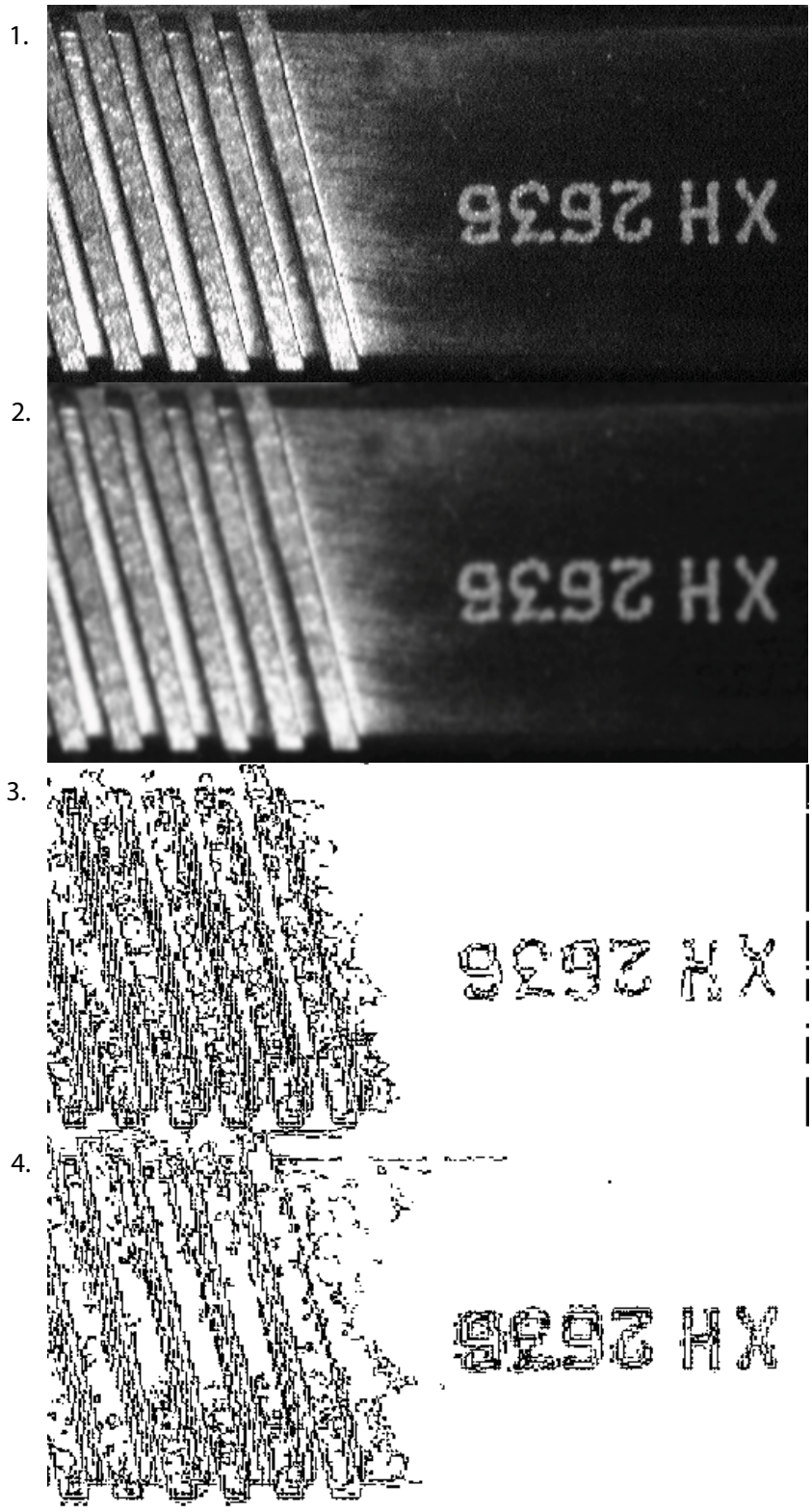
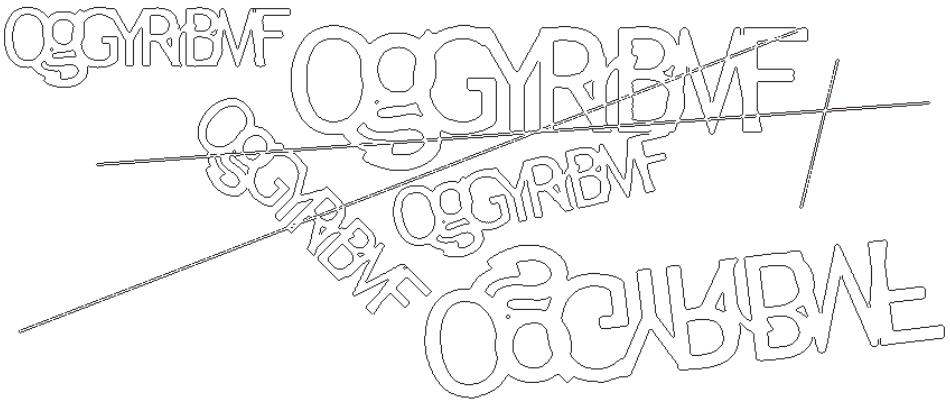


Рис. 5.7. Обработка зашумлённого изображения клейма(1) нейросетевым детектором(10 подсетей)(3) и обработка того же изображения(2), предварительно сглаженного по Гауссу с $\sigma = 1.5$, детектором с теми же параметрами без повторного обучения

Заключение

1.

2.



3.

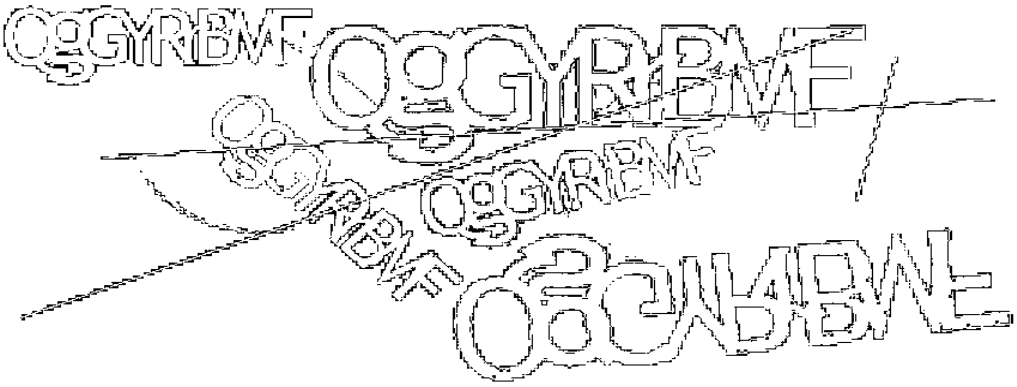


Рис. 5.8. Обработка сложного разнородного изображения(1) детектором Кэнни($\sigma = 0.5, t_1 = 0.04, t2 = 0.10s$)(2) и нейросетевым детектором(3)(10 прототипов освещённости)

Разработанный контурный детектор на основе нейросетевой технологии показал себя в процессе разработки и экспериментов с разных сторон. С одной стороны, его очевидная привлекательность состоит в простоте его реализации и интеллектуальном подходе к настройке через обучение и самообучение. Работа с детектором позволила увидеть как слабые, так и сильные стороны применённой технологии. Сильной стороной является возможность достижения высокой производительности при обработке больших изображений за счёт простоты вычислений, производимых фильтром, сводящихся по большей части к сравнениям действительных чисел. Также нельзя не отметить в числе преимуществ возможность обучить программу для обработки более широкого класса изображений, чем входящий в ограниченный одним набором параметров для традиционных детекторов диапазон образцов. Наглядно это можно увидеть на рис.5.8, демонстрирующем, что, несмотря на худшую форму контура, исследуемый фильтр показал не худшую связность контура и качество его обнаружения, чем великолепно зарекомендовавший себя за более чем два десятилетия использования фильтр Кэнни. При этом, не проходя переобучения, фильтр сохранил способность к обработке изображений другого содержания, и обучение может позволить достичь подобного баланса для большего количества классов изображений, тогда как традиционные фильтры требуют перенастройки для перехода к новой задаче.

Качеством исследуемого фильтра также является разнородность содержания изображения, обработать которую скорее в состоянии нейросетевой фильтр, так в нём заложены возможности по работе с большим количеством параметров освещённости, представленных весовыми коэффициентами нейронной сети. Как объяснялось в главе 3, эта схема интерпретации близка к визуально-психологической модели, используемой человеком. Как было сказано ранее, в ходе экспериментов возникли предположения о возможной оптимизации программы и самого алгорима, которая позволила бы фильтру улучшить полученные результаты.

Недостатками разработанной системы следует считать сложность обучения, руководимого скорее интуицией, чем прогнозами, и чересчур высокое разрешение обработки изображений, приводящее к трудноустранимым искажением при наличии сильного шума в изображениях и нарушениям цельности и формы контура.

Литература

[1] Хайкин, С. Нейронные сети. Полный курс, 2-е изд., испр., 2006.-1104с, ISBN 5-8459-0890-6

[2] Ф. Уоссермен Нейрокомпьютерная техника. Теория и практика. — 1-е. — М.: Мир, 1992. — С. 240. — ISBN 5-03-002115-9

[3] S. W. Perry, H. S. Wong and L. Guan, Adaptive Image Processing: A Computational Intelligence Perspective, CRC Press/ SPIE Press, 2002.

[4] Minsky M. and Papert S. Perceptrons. Cambrige, MA:MIT Press, 1969(рус.:Минский М.Л., Пейперт С. Перцептроны.-М. Мир. - 1971)

[5] Hebb D.O. Organization of behaviour. New York:Science Editions, 1949.

[6] Шлее М., Qt4. Профессиональное программирование на C++. - СПб.: БХВ-Петербург, 2007. - 880с., ISBN 978-5-9775-0010-6

[7] Molkentin D. The Book of Qt4: The Art of Building Qt Applications. 2007 Open Source Press GmbH, 425c., ISBN 978-3-937514-12-13

[8] Kohonen T. 1984. Self-organization and associative memory. Series in Information Sciences. vol/ȧ. Berlin.: Springer Verlag

[9] Chapter 15 Kohonen networks of Neural Networks - A Systematic Introduction by Raul Rojas (ISBN 978-3540605058)

[10] Визильтер Ю.В., Желтов С.Ю., Князь В.А., Ходарев А.Н.: Обработка и анализ цифровых изображений с примерами на LabVIEW и IMAQ Vision, 464 стр.,2007,т.: 1000 экз.,ДМК Пресс [М.] ISBN: 978-5-94074-404-7

[11] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins. Digital Image Processing Using MATLAB, 624 c., Prentice Hall, 2003 ISBN: 978-0130085191

[12] W. Zhang and F. Bergholm: Multi-Scale Blur Estimation and Edge Type Classification for Scene Analysis, International Journal of Computer Vision, Volume 24, Issue 3, pages 219 - 250, 1997

[13] Lindeberg, T., "Edge detection and ridge detection with automatic scale selection International Journal of Computer Vision, 30, 2, стр 117–154, 1998.

Приложение

Листинги программы

greyscalewindow.h

```
14 #ifndef GREYSCALEWINDOW_H
15 #define GREYSCALEWINDOW_H
16
17 #include "globals.h"
18 #include <QImage>
19 #include <QColor>
20
21
22 class GreyscaleWindow {
23 public:
24     GreyscaleWindow();
25     GreyscaleWindow( const QImage& img );
26     GreyscaleWindow( const GMatrix& img );
27     GreyscaleWindow( const GreyscaleWindow& other );
28     uint           pixel( int x, int y );
29     void                  setPixel( int x, int y, uint value );
30     double         upperGreyLevel() const;
31     double         lowerGreyLevel() const;
32     double         averageGreyLevel() const;
33     int                   width() const;
34     int                   height() const;
35     const GMatrix& matrix() const;
36     GreyscaleWindow& operator = ( const GreyscaleWindow& other );
37     QImage         toImage();
38     QImage         toMonochromeImage();
39 private:
40     GMatrix GreyscaleMatrix;
41     double  ugl;
42     double  lgl;
43     double  agl;
44     void    calculateLevels();
45 };
46
47 #endif
```