МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**
Кафедра **«Прикладная математика и информатика»**

01.03.02 ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

БАКАЛАВРСКАЯ РАБОТА

на тему: **Автоматическое выделение границ сосудов на растровых изображениях рентгенограмм**

Студентка Е.С. Попова _____

Руководитель М.Г. Лисовская _____

Консультант по аннотации Н.В. Ященко _____

Допустить к защите
Заведующий кафедрой к.т.н., доцент А.В. Очеповский _____

«_____» _____ 20___ г.

Тольятти 2017

АННОТАЦИЯ

На бакалаврскую работу студентки Поповой Е. С. на тему «Автоматическое выделение границ сосудов на растровых изображениях рентгенограмм».

Объектом исследования является процесс выделения границ сосудов на рентгеновских снимках.

Цель работы: разработать алгоритм и его программную реализацию по выделению границ сосудов на растровых изображениях рентгенограмм.

Предмет исследования: программа определения границ сосудов на снимках.

В бакалаврской работе раскрывается роль проблемы в современном мире, рассматриваются существующие методы решения. Производится тестирование разработанной программы, исправляются найденные ошибки, и формируется окончательный результат.

Разработанная программа выпускной квалификационной работы способна выполнять анализ изображения, полученного на входе, и максимально эффективно выделять необходимые объекты.

Объем бакалаврской работы – 42 страницы, она содержит 20 рисунков, 25 источников литературы.

ANNOTATION

The title of the graduation work is “Automatic selection of the boundaries of vessels on raster images of radiographs”.

The graduation work consists of an explanatory note on 42 pages, introduction, three parts, including 20 figures, 3 tables, conclusion, the list of 25 references including 7 foreign sources and 1 appendices.

The object of the graduation work is the process of choosing the boundaries of vessels on X-ray images.

The subject of the graduation work is to create program to determine the boundaries of vessels on image.

The aim of the work is to develop an algorithm and its software to identify the boundaries of vessels on raster images.

This study examines the relation between medicine and Programming. First of all, we reveal the role of our problem in the modern world. Then we look at the existing methods for solving our problem and choose the most appropriate solution. In a separate part of the project, we consider information about programming languages and development environments, choose the most efficient for us and give an explanation of the choice. We also test the created program, correct the errors received and get the final result.

The conclusions are drawn: the developed program of the graduation work is able to perform an analysis of the image received at the entrance, and as much as possible effectively to allocate the necessary object, the used technology correspond software standards, the received program can be useful for health professionals.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Мировые тенденции в области медицинского приборостроения в последние годы приобрели значительные изменения. В основном это связано с необходимостью улучшения качества диагностики, что приводит к созданию новых улучшенных аналогов диагностических приборов, а так же к усовершенствованию традиционных технологий.

Выделение сосудов и оценка их морфологических признаков, таких как длина, ширина, углы ветвления и другие, в наши дни активно используется для диагностики, мониторинга и выявления различных видов заболеваний.

Научной новизной данной работы будет являться разработка простого алгоритма выделения границ сосудов на изображениях. Разрабатываемый алгоритм должен быть направлен на максимальное выделение всех границ сосудов и на подавление обнаруженных ложных контуров, которые возникают, например, от присутствия импульсного шума.

В качестве исходных данных используются рентгеновские снимки двух видов: прямой и боковой проекций. На всех снимках содержатся изображения сосудов.

Объектом исследования является процесс выделения границ сосудов на рентгеновских снимках.

Предмет исследования: создание программы определения границ сосудов на снимках.

Цель работы: разработать алгоритм и его программную реализацию для выделения границ сосудов на растровых изображениях рентгенограмм.

Задачи, решаемые в ходе выполнения бакалаврской работы:

- анализ предметной области;
- математическое описание задачи;
- изучение методов выделения границ;
- выбор алгоритма для реализации;

- выбор языка программирования;
- реализация поставленной задачи с использованием выбранного языка программирования;
- тестирование разработанного алгоритма.

1 Анализ алгоритма выделения границ сосудов на изображении

1.1 Анализ предметной области

Компьютерные технологии в наши дни очень популярны и пользуются большим спросом. Без них невозможно представить себе, наверное, ни одну сферу деятельности человека.

В современной медицине практически любое лечение не обходиться без исследования и анализа внутренней структуры объекта. Медицинская визуализация - это раздел в медицинской диагностике, который осуществляет исследование организма человека, с целью получения изображения органов и тканей тела человека [12].

Флюорография - метод, позволяющий получить изображение органов и тканей, который был разработан в конце 20-го столетия, спустя примерно год после того, как были обнаружены рентгеновские лучи. Суть флюорографии: через тело человека пропускаются рентгеновские лучи, они отражаются от тканей, которые имеют различную плотность, и выводят на пленку или экран черно-белое изображение грудной клетки. На полученных снимках можно разглядеть склероз, фиброз, инородные предметы, новообразования, воспаления и так далее. Чаще всего делается флюорография грудной клетки, что позволяет выявить такие заболевания, как туберкулез, злокачественная опухоль в легких или груди, а так же другие патологии органов человека. Опять же данную методику применяют для обследования сердца и костей [2].

В настоящий момент плёночная флюорография стала поэтапно заменяться цифровой. Цифровые методы позволяют облегчить работу со снимками, сократить лучевую нагрузку на человека, а так же уменьшить расходы на необходимые материалы (плёнку, проявитель для плёнки).

Среди существующего множества методов особенного успеха достигла рентгеновская компьютерная томография (КТ). Предпосылками к её

появлению являлись недостатки обычной рентгенографии, которые дали повод зарождения идеи получения не одного, а сразу целого ряда снимков, выполненных под разными ракурсами, и обнаружения на них путём математической обработки плотностей исследуемого вещества в ряду сечений. Компьютерная томография, вместо известной флюорографии, позволяет сократить уровень смертности от рака легких примерно на 20%.

Компьютерная томография даёт вероятность увидеть все изменения на рентгеновских снимках. Массовое применение метода КТ связано с более высокой точностью и быстротой исследований. Данная процедура выполняется для диагностики различных заболеваний [13].

Появление новых компьютерных технологий значительно увеличило уровень и качество современной медицинской помощи [1].

Магнитно-резонансная томография (МРТ) - метод отображения, который основан на ядерно-магнитном резонансе, используется предпочтительно для медицинских исследований. Основными преимуществами МРТ перед КТ являются более высокие разрешающиеся способности, большая контрастность изображений, возможность получить срезы в различных плоскостях и отсутствие гамма-лучевого воздействия на пациента. МРТ в сравнении с КТ предоставляет возможность составить более чёткое представление об объёме и неравномерности распространений заболевания.

Рентгенологические методы исследований в настоящее время занимают высокое место в обследовании пациентов. Они позволяют определить анатомические структуры патологических изменений и выявить их характер. При расшифровке теневого изображения на рентгенограмме определяют местоположение поражения, его качественные характеристики, а так же динамику в процессе наблюдения и лечения [23].

Несмотря на то, что в нынешнее время достаточно широко развиваются такие методы диагностики, как КТ и МРТ, имеющие довольно высокую диагностическую информативность, обследование этими методами имеют

определенный ряд недостатков, таких как: дорогое оборудование и специализированное программное обеспечение. Всё это повышает стоимость исследования, и поэтому в настоящее время предложенные методы визуализации используются преимущественно после рентгенографии при подозрении на заболевания, которые требуют дополнительного обследования.

При обработке полученных при обследовании изображений особенно значимой характеристикой представляется яркость пикселей изображения [14]. Как известно, яркость точек изображения может влиять на множество факторов, например: напряжение на рентгеновской трубке, габариты пациента и другие. Это, несомненно, затрудняет врачу постановку диагноза. В связи с этими факторами, разумнее всего будет предпринять следующие действия: уменьшить влияние данных факторов и перейти к оценке относительной яркости точек полученных снимков [5].

Существует обширное множество подходов к выделению границ, и почти все из них можно разделить на две категории: методы, основанные на определении максимумов, и методы, основанные на обнаружении нулей. Методы, которые основаны на определении максимумов, производят выделение границы с помощью вычисления «силы края», в основном выражение первой производной, а затем находятся локальные максимумы силы края, используя при этом наиболее вероятное направление границы, обычно используется перпендикуляр к вектору градиента. Методы, которые основаны на обнаружении нулей, ищут пересечения оси абсцисс выражения второй производной, нули лапласиана либо нули нелинейного дифференциального выражения. В качестве шага обработки выделения границ изредка применяется сглаживание изображения, обычно это делается при помощи фильтра Гаусса [9].

Выделение границ основывается на операторах, определяющих резкую смену яркости на изображении. В результате выполнения данного алгоритма должно быть некоторое количество связанных кривых, которые, в свою

очередь, отображают границы объекта. Обычно границы на изображениях выражены не очень ярко и часто результат обработки изменяется от разрывов контуров и линий до почти полного отсутствия границы [10].

1.2 Описание математического и алгоритмического решения задачи

Рассмотрим наиболее распространенные методы выделения границ изображения – оператор Кэнни, оператор Собеля и оператор Робертса [15].

1.2.1 Оператор Кэнни

Наиболее популярным методом выделения границ на изображении является оператор Кэнни. John Canny описал алгоритмы нахождения границ, которые с тех пор стали одними из наиболее часто используемых. Они стали классикой в области нахождения границ [4].

Оператор Кэнни — это алгоритм нахождения границ изображения. Данный алгоритм был разработан в 1986 году Джоном Кэнни. Оператор использует многоступенчатый алгоритм для выявления широкого спектра границ в изображениях.

Целью Кэнни было разработать наиболее адаптированный алгоритм для выделения границ, который бы соответствовал трём основным критериям:

- хорошая локализация;
- хорошее обнаружение;
- единственный отклик на каждую из границ.

Алгоритм выделения границ не ограничивается только вычислением градиента сглаженного изображения. В контуре границы остаются исключительно точки максимума градиента изображения, а не максимальные точки, которые лежат рядом с границей, удаляются. Также в данном алгоритме используется информация о направлении вектора границы для того, чтобы удалять точки которые лежат рядом с границей и не разрывать саму границу рядом с локальными максимумами градиента. После этого, с

помощью всего двух порогов удаляются слабые границы. Часть границы при этом обрабатывается как целая. Если значение градиента где-нибудь на исследуемом фрагменте превышает верхний порог, то этот фрагмент будет являться «допустимой» границей только в тех местах, где значение градиента падает ниже этого порога, до тех пор, пока не станет ниже нижнего порога. Если же на всем фрагменте не существует точки со значением, которое больше верхнего порога, то он удаляется. Именно такой гистерезис позволяет значительно уменьшить число разрывов в выходных границах [20].

Оператор Кэнни основывается всего на трех критериях. Основная идея алгоритма заключается в том, что используется функция Гаусса для сглаживания изображения. Тогда максимальное значение первой производной также соответствует минимуму первой производной. Другими словами, обе точки с резким изменением серого - масштаба (сильный край) и точек с небольшим изменением в оттенках серого (слабых краев) соответствуют второй производной точки пересечения нуля. Таким образом, эти два порога используются для обнаружения сильных и слабых краев. Тот факт, что алгоритм Кэнни не чувствителен к шуму, позволяет ему успешно обнаруживать истинные слабые края.

Основные этапы алгоритма Кэнни:

1. Сглаживание. Размытие изображения для удаления шума. Алгоритм Кэнни использует фильтр, который может быть приближен к первой производной гауссианы. σ = 1.4:

$$B = \frac{1}{159}\begin{matrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{matrix} * A \qquad (1.1)$$

где $*$ обозначает двумерную операцию свертки, σ – параметр фильтра, определяющий коэффициент пропорциональности и степень размытия.

2. Поиск градиентов. Границы отмечаются там, где градиент изображения приобретает max значение. Они могут иметь различное направление, именно поэтому алгоритм Кэнни использует 4 фильтра для обнаружения горизонтальных, вертикальных и диагональных ребер в размытом изображении.

$$G = \sqrt{G_x^2 + G_y^2}, \quad (1.2)$$

$$\theta = arctg(\frac{G_y}{G_x}), \quad (1.3)$$

где G_x – градиент по оси x, G_y - градиент по оси y, θ – угол направления границы.

3. Подавление не-максимумов. Только локальные максимумы отмечаются как границы. Пикселями границ объявляются пиксели, в которых достигается локальный максимум градиента в направлении вектора градиента. Значение направления должно быть кратно 45°.

4. Двойная пороговая фильтрация. Выделение границ использует два порога фильтрации (верхний и нижний): если значение пикселя выше верхней границы – он принимает максимальное значение (граница является достоверной), если ниже – пиксель подавляется. Точки со значением, лежащие в диапазоне между порогов, принимают фиксированное среднее.

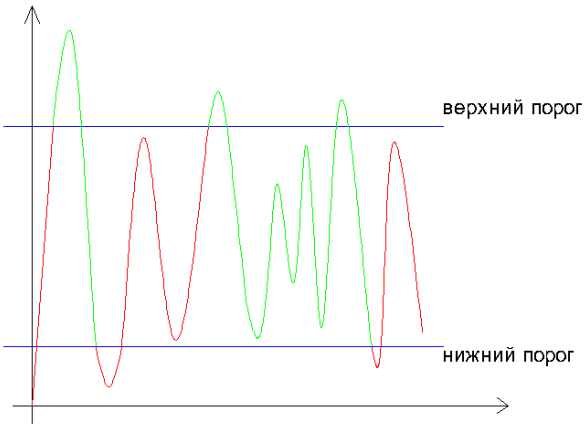

Рисунок 1.1 – Пороги оператора Кэнни

5. Трассировка области неоднозначности. Задача сводится к выделению групп пикселей, которые получили на предыдущем этапе промежуточные значения, и отнесению их к границе (если они связанны с одной из установленных границ) или их подавлению (в обратном случае). Пиксель добавляется к группе, если он соприкасается с ней по одному из 8-ми направлений.

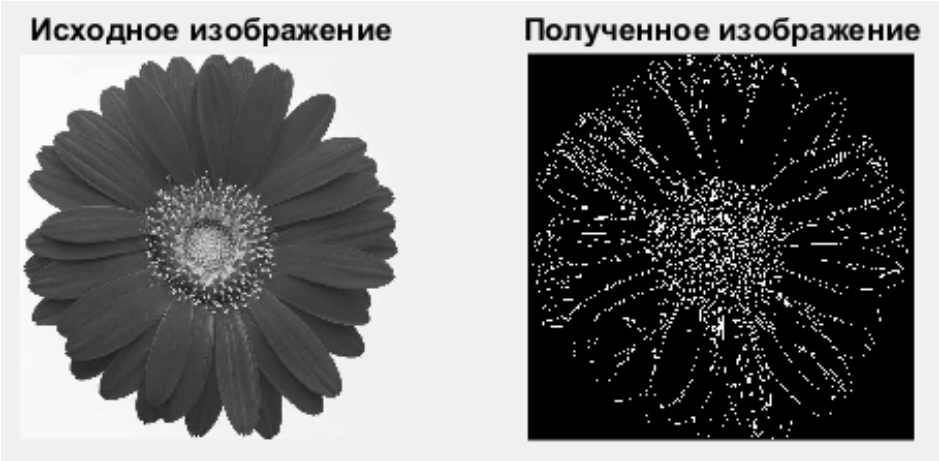

Рисунок 1.2 – Пример работы оператора Canny

1.2.2 Оператор Собеля

Оператор Собеля хорошо известен во всем мире и применяется для множества задач. Строго говоря, оператор использует ядра 3×3, с которыми сворачивает исходное изображение для вычисления приближённых значений производных по горизонтали и по вертикали. Оператор Собеля производит измерение двумерного пространственного градиента на изображении и выявляет области с большим значением этого параметра. Эти области и соответствуют краям. Как правило, он используется для оценки модуля градиента в каждой точке черно-белого изображения.

Оператор вычисляет коэффициент яркости изображения в каждой точке. Таким образом, находится направление наибольшего увеличения яркости, а также величина изменения в данном направлении. Результат будет показывать, насколько «резко» или «плавно» изменяется яркость изображения в каждой его точке, а значит, вероятность нахождения точки на

грани, а также ориентацию границы. В работе вычисления величины изменения яркости намного надежнее и проще в интерпретации, чем расчет направления. Результатом оператора Собеля в точке области постоянной яркости будет нулевой вектор, а в точке, лежащей на границе областей различной яркости — вектор, пересекающий границу в направлении увеличения яркости [22].

Идея алгоритма Собеля заключается в наложении на каждую точку изображения двух масок. Эти маски являются двумя ортогональными матрицами, размерностью 3 × 3, вертикальные маски(1.4) и горизонтальные маски(1.5).

$$\begin{matrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{matrix} \qquad \begin{matrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix} \tag{1.4}$$

$$\begin{matrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{matrix} \qquad \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{matrix} \tag{1.5}$$

Для разрешения вопроса инвариантности в отношении поворота используются так называемые диагональные маски(1.6), предназначенные для обнаружения разрывов в диагональных направлениях.

$$\begin{matrix} 0 & +1 & +2 \\ -1 & 0 & +1 \\ -2 & -1 & 0 \end{matrix} \qquad \begin{matrix} +2 & +1 & 0 \\ +1 & 0 & -1 \\ 0 & -1 & -2 \end{matrix} \tag{1.6}$$

Данные маски обнаруживают границы, которые расположены вертикально и горизонтально на изображении. При раздельном наложении этих масок на изображение можно получить оценку градиента по каждому из направлений G_x, G_y. Конечное значение градиента находится формуле:

$$G = \overline{G_x^2 + G_y^2}\ , \tag{1.7}$$

где G_x – градиент по оси x, G_y - градиент по оси y.

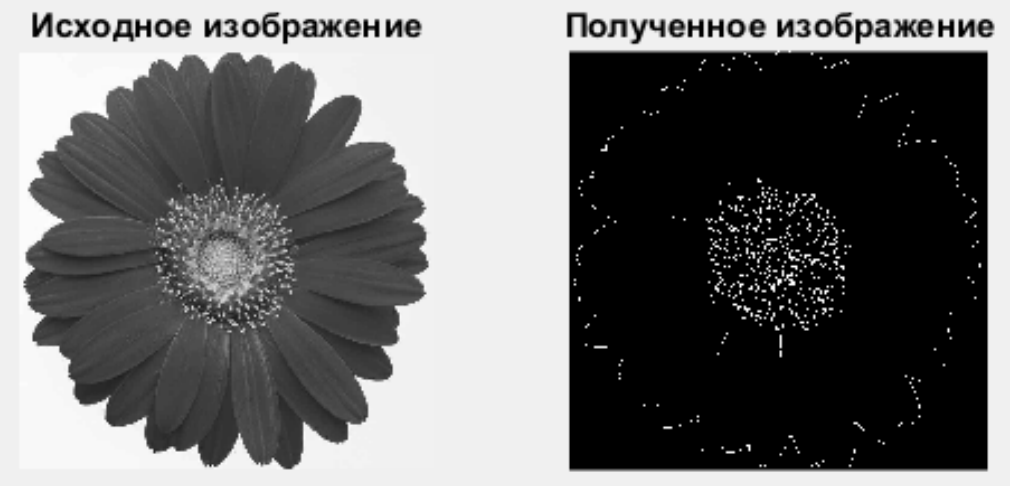

Рисунок 1.3 – Пример работы оператора Собеля

1.2.3 Оператор Робертса

Оператор Робертса — один из самых ранних алгоритмов выделения границ, он вычисляет сумму квадратов разниц между диагонально смежными пикселями. Это может быть выполнено сверткой изображения с двумя ядрами:

$$H_1 = \begin{matrix} +1 & 0 \\ 0 & -1 \end{matrix}, \quad (1.8)$$

$$H_2 = \begin{matrix} 0 & +1 \\ -1 & 0 \end{matrix}, \quad (1.9)$$

где H_1, H_2 – ядра свёртки.

Иными словами, величина перепада G получаемого изображения вычисляется из исходных значений в точках растра $Y_{x,y}$ по правилу:

$$P = Y_{x,y} - Y_{x+1,y+1}, \quad (1.10)$$

$$Q = Y_{x+1,y} - Y_{x,y+1}, \quad (1.11)$$

$$G = \overline{P^2 + Q^2}, \quad (1.12)$$

где P – отклик ядра H_1, Q - отклик ядра H_2.

Оператор Робертса является нелинейным фильтром. Преобразование каждого пикселя перекрёстным оператором Робертса может показать нам производную изображения вдоль ненулевой диагонали, и комбинация данных преобразованных изображений может рассматриваться, как градиент

от двух верхних пикселей к двум нижним. Оператор Робертса всё ещё используется с целью быстрых вычислений, но он значительно проигрывает, в сравнении с другими методами, из-за значительной чувствительности к шуму, что часто не является допустимым. Он выделяет более тонкие линии, чем другие методы выделения границ, что почти равносильно вычислению конечных разностей вдоль координат X и Y. Иногда его называют «фильтром Робертса».

К недостаткам данного оператора относят высокую чувствительность к шуму и ориентацию границ областей, вероятность образования разрывов в контуре, а также отсутствие выраженного центрового элемента. А к достоинствам – малую ресурсоемкость.

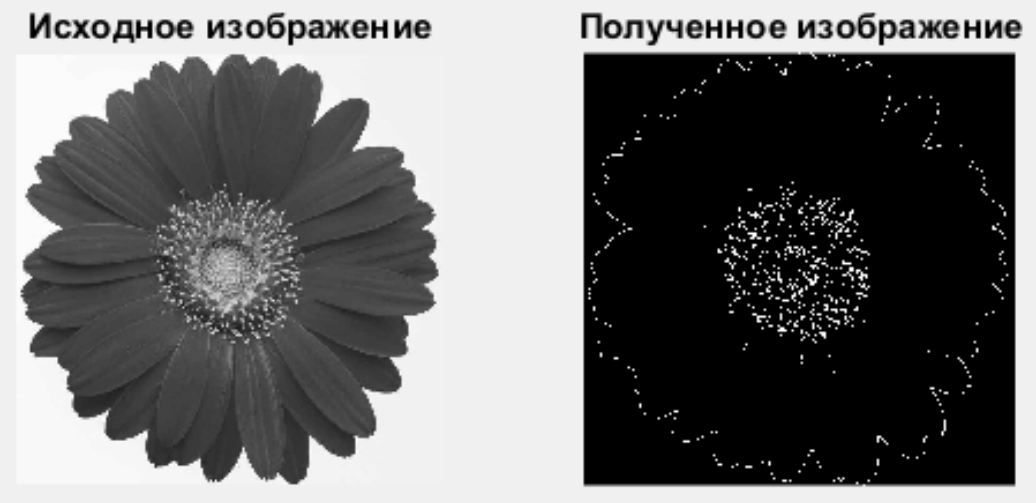

Рисунок 1.4 – Пример работы оператора Робертса

1.3 Выбор алгоритма для реализации выделения границ

Таблица 1.1 - Сравнительный анализ методов нахождения границ изображения

Критерий оценки	Кэнни	Собель	Робертс
Простота разработки	+	-	+
Обнаружение краев в условиях шума	+	-	-
Обнаружение границ и их направления	+	+	-
Экономичный по времени	+	+	+

На основе произведенного анализа был выбран алгоритм Кэнни. Данный алгоритм больше всех удовлетворяет требованиям оценки. Более экономичный по времени, по сравнению с оператором Собеля и более лучший в условиях шума, по сравнению с оператором Робертса.

2 Проектирование и реализация алгоритма

2.1 Анализ и выбор технологии реализации

Выбор технологии для разработки программного продукта важен не только для разработки, но и для дальнейшего внедрения и поддержки программы. Каждый язык программирования обладает как плюсами, так и минусами, что может либо облегчить реализацию, усложнить её для каждого отдельного проекта.

2.1.1 Описание языков программирования

Java — объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года. Официально язык был выпущен 23.05.1995 года. Сначала новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но позже был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения [21].

Программы, написанные на Java, имеют репутацию более медленных и занимающих больше оперативной памяти, чем программы, написанные на языке C++.

C++ - язык программирования общего назначения. Естественная для него область применения - системное программирование, понимаемое в широком смысле этого слова. Кроме того, C++ успешно используется во многих областях приложения, далеко выходящих за указанные рамки. Реализации C++ теперь есть на всех машинах, начиная с самых скромных микрокомпьютеров – до самых больших супер-ЭВМ, и практически для всех операционных систем [19].

Язык программирования C++ задумывался как язык, который будет:

- лучше языка C;
- поддерживать абстракцию данных;
- поддерживать объектно-ориентированное программирование.

C# (си-шарп) - это простой и достаточно многофункциональный язык. В нём собрано множество достоинств от разных языков. C# очень близок к C++ и Java. Поэтому, если вы знаете хотя-бы один из этих языков, то научиться программировать на C# будет намного легче. C# является языком объектно-ориентированным и в этом плане многое перенял у Java и C++. К примеру, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию [11]. Объектно-ориентированный подход позволит решить задачи по построению, с одной стороны, крупных, но и в тоже время гибких, масштабируемых и расширяемых приложений. В настоящее время C# продолжает активно развиваться, и с каждой новой версией появляются все больше интересные функциональности.

Переняв многое от языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — C#, опираясь на практику их использования, исключил некоторые модели, которые показали себя как проблемные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов.

MATLAB - высокоуровневый интерпретируемый язык программирования, который включает в себя достаточно широкий спектр функций, интегрированную среду разработки, основан на матричных структурах данных, объектно-ориентированные возможности, написанные на других языках программирования [6].

Пакет MatLab был создан компанией Math Works уже более десяти лет назад. Работа многих ученых и программистов в наши дни направлена на постепенное расширение его возможностей и совершенствование существующих алгоритмов. В настоящее время MatLab является достаточно мощным, а также универсальным средством для решения задач, которые возникают в различных областях человеческой деятельности.

Программы, которые написаны на языке MATLAB, бывают двух типов. Первый тип - функции, они имеют входные и выходные аргументы, а также собственное рабочее пространство для хранения промежуточных результатов

вычислений и переменных. Второй тип - скрипты, которые используют общее рабочее пространство. Как скрипты, так и функции сохраняются в виде текстовых документов и производят компиляцию в машинный код динамически. Также существует возможность сохранить pre-parsed программы — это функции и скрипты, которые обработаны в вид, удобный для исполнения машиной. В общих случаях такие программы выполняются гораздо быстрее обычных, особенно если функция содержит команды для построения графиков [8].

2.1.2 Сравнительный анализ языков программирования

Сравнительный анализ языков программирования представлен в таблице 2.1.

Таблица 2.1 – Сравнительный анализ языков программирования

Критерии оценки	C#	Java	C/C++	Matlab
Простота разработки	-	+	-	+
Быстродействие программы	+	-	-	+
Простая работа с изображениями	-	-	-	+
Ручное управление памятью	+	-	+	+
Встроенные реализованные функции	-	-	-	+

На основе проведенного сравнительного анализа в качестве языка программирования для разработки был выбран Matlab. Произведя анализ, мы выяснили, что Matlab значительно превосходит другие языки. Главными критериями такого выбора послужили простота разработки, а также быстродействие программы. Немало важно, что в Matlab есть встроенные функции, которые значительно ускоряют процесс разработки нашей программы.

2.2 Разработка алгоритма и программы

В ходе выполнения выпускной квалификационной работы был использован язык разработки Matlab и среда разработки Matlab R2016a.

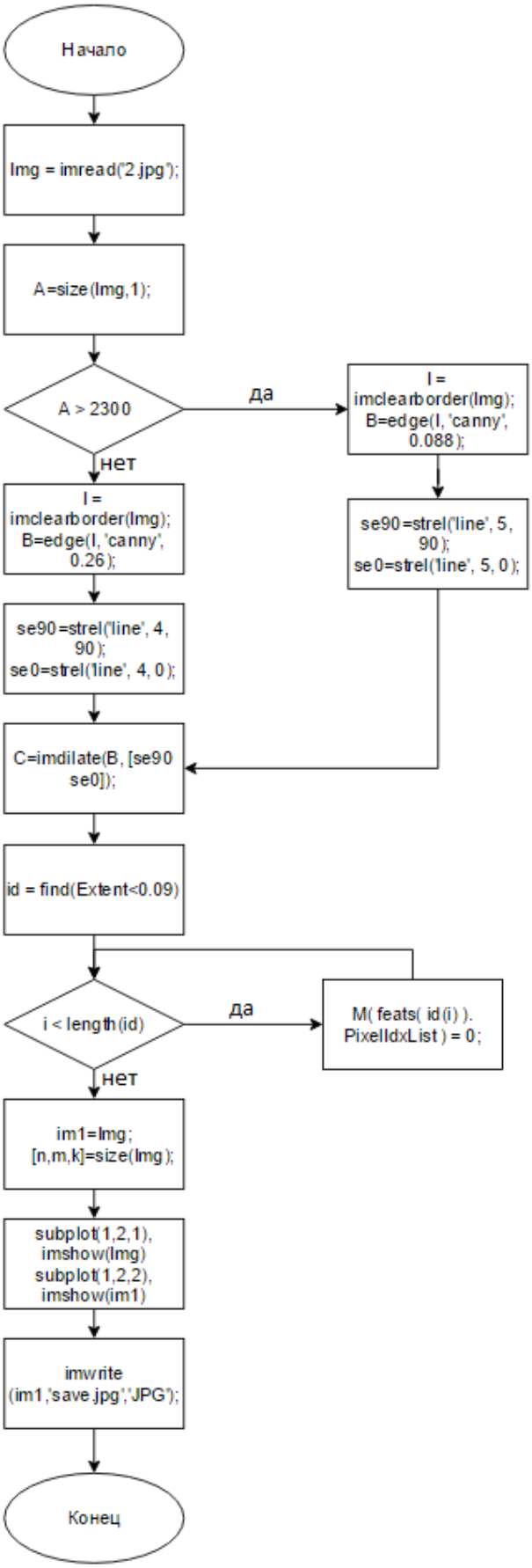

Рисунок 2.1 – Блок-схема алгоритма

Для разработки программы были разработаны алгоритмы, которые обладают следующими характеристиками:

- определение типа снимка (прямая или боковая проекция);
- улучшение области, в которой находятся сосуды;
- выделение границ;
- удаление ненужных границ;
- наложение полученных границ на исходное изображение;
- сохраняем полученное изображение.

Для каждого типа снимка используется собственный алгоритм со своими особенностями.

Рисунок 2.2 – Загружаемое изображение

Каждый снимок имеет свои размеры: длину и высоту. Снимки, сделанные в прямой проекции, имеют высоту более 2300 пикселей, а у снимков в боковой проекции высота менее 2300 пикселей. Для определения

типа снимка, нам необходимо проверить высоту загружаемого снимка. Для этого будем использовать функцию вида A=size(Img,1). Переменная A будет хранить в себе 1 число - высоту изображения Img.

Для того чтобы программа понимала изображение в какой проекции загружается, будем использовать оператор условия if. В нашем случае if A > 2300, если условие истинно, то применяем алгоритм для снимков в прямой проекции, иначе следуем по другому алгоритму.

Функция вида I = imclearborder(Img), осуществляет подавление световой структуры. Img – исходное изображение. В нашем случае данная функция выделяет область грудной клетки, чтобы изначально исключить возможность выделения ненужных границ (рисунок 2.3).

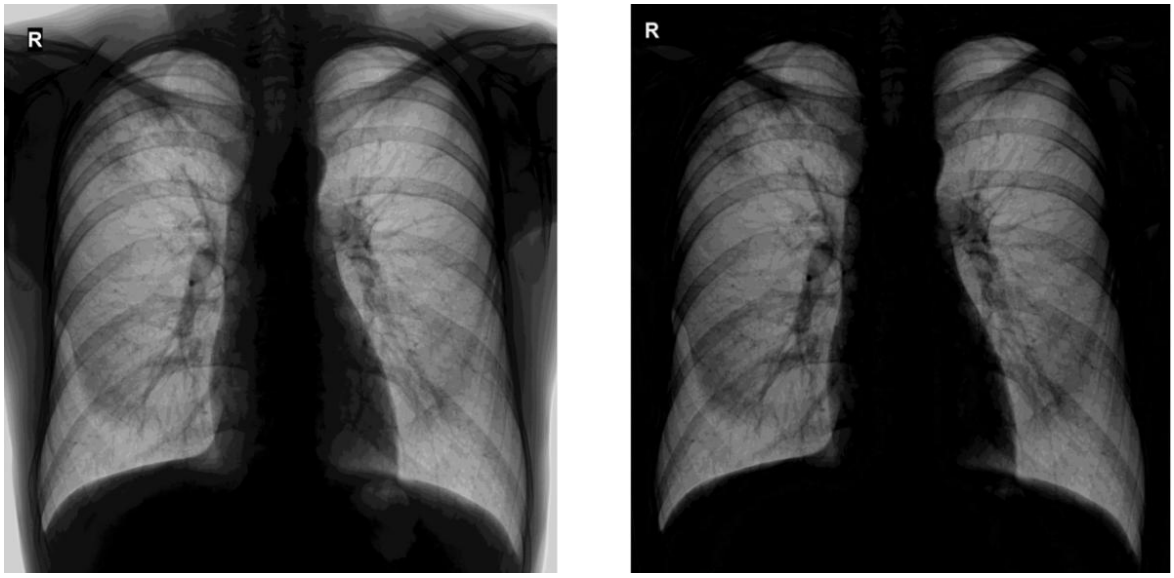

Рисунок 2.3 – Применение функции imclearborder

Для выделения границ в среде Matlab есть вложенная функция вида:

BW=edge(I, method, thresh),

где I – исходное полутоновое изображение, method – метод выделения границ, в нашем случае это canny, thresh – нижний порог, который определяет, принадлежит ли пиксель к границе и BW – полученное бинарное изображение, совпадающее по размеру с входным изображением [18].

Если в функции не указывать нижний порог thresh, то значение порога выбирается автоматически. Пиксель считается относящимся к границе, если

соответствующий ему пиксель результата фильтрации имеет значение, большее thresh. Пороговое значение для выделения границ на снимках прямой проекции примем 0.088, для боковой проекции 0.26. Данные значения были получены экспериментальным путём.

Вид функции для снимков в прямой проекции:

B=edge(I, 'canny', 0.088),

для боковой проекции:

B=edge(I, 'canny', 0.26);

Для того чтобы линии были более точными используем функцию вида:

se90=strel('line', 4, 90);

se0=strel('line', 4, 0);

C=imdilate(B, [se90 se0]).

Функция strel('line', 4) создает линии длинной 4. Функция imdilate выполняет бинарное наращивание со структурными элементами объекта, декомпозируя их, данная функция использует процедуру упаковки бинарных изображений для ускорения операции наращения [16]. B – исходное изображение. C – изображение после применения функции imdilate (рисунок 2.4).

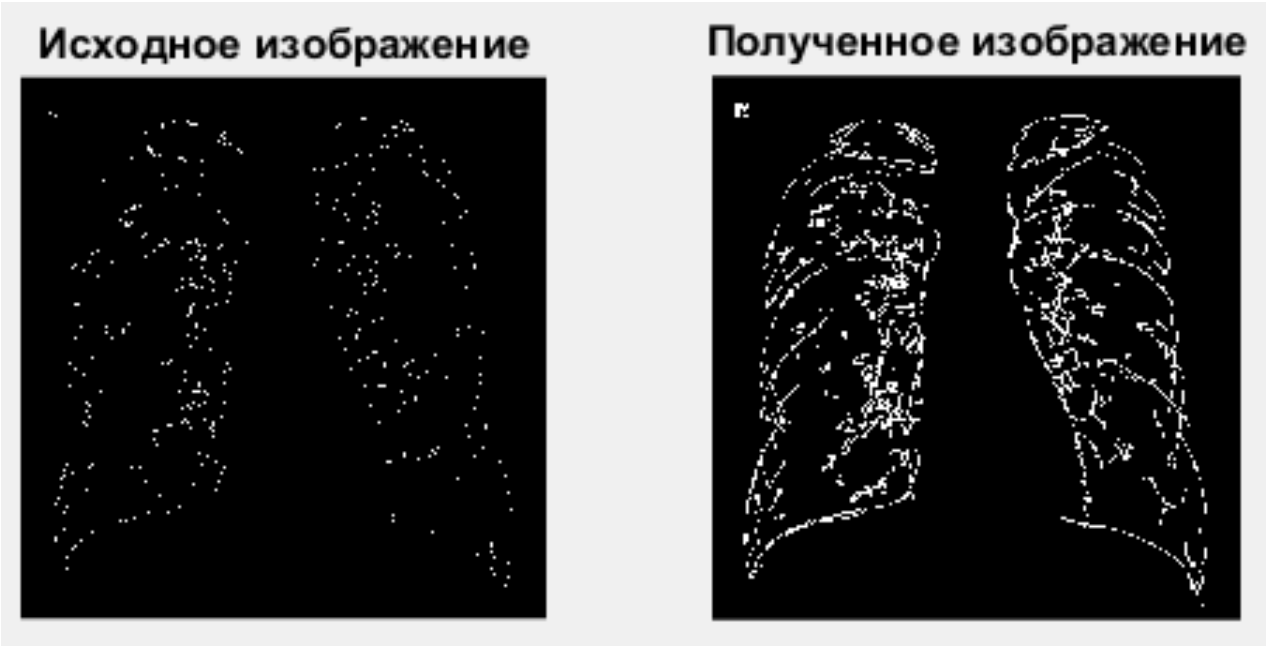

Рисунок 2.4 - Применение функции imdilate

Так как полученное нами изображение содержит не только границы сосудов, а все возможные границы на изображении, нам необходимо удалить ненужные границы. Для этих целей используем алгоритм:

```
[M, num]=bwlabel(C, 8);
feats = regionprops(M, 'Extent','PixelIdxList');
Extent = [feats.Extent]; id = find(Extent<0.1).
```

Функция Extent используется для вычисления коэффициента заполнения, равному отношению площади объекта к площади ограничивающего прямоугольника. Значение id ищет ненужные для нас границы. Чтобы удалить эти границы используется следующий алгоритм:

```
for i=1:length(id)
   M( feats( id(i) ).PixelIdxList ) = 0;
end
```

Программа проходит по всем точкам изображения M и удаляет ненужные для нас границы. Пример продемонстрирован на рисунке 2.5.

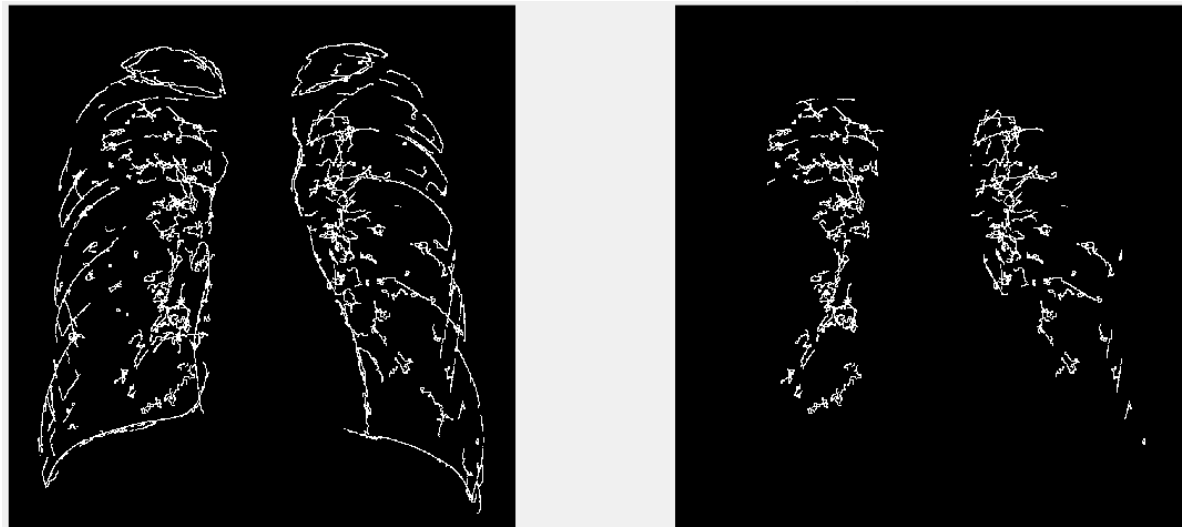

Рисунок 2.5 – удаление лишних границ

Для наложения полученного изображения на исходное используем следующий алгоритм:

```
im1=Img;
   [n,m,k]=size(Img);
    for i=1:n
```

```
for j=1:m
 if (M(i,j)~=0)
 im1(i,j,1)= 0;
 im1(i,j,2)= 0;
 im1(i,j,3)= 0;
 end
end
end
```

Суть алгоритма: программа создаёт новое изображение im1, которое по размеру равно исходному и производит наложение полученного изображения на исходное. Выделенные программой границы, отобразятся на исходном снимке чёрным цветом.

Для вывода изображения на экран используем функцию вида subplot(a,b,c). Данная функция производит разбивку графического окна на несколько подокон [17]. Значение a - число подокон по горизонтали, значение b – количество подокон по вертикали, значение c – номер подокна, куда нужно вывести наше изображение. Для вывода двух изображений, исходного и полученного, мы разбиваем наше окно на 2 подокна по вертикали. Вид итоговой функции:

```
subplot(1,2,1),imshow(Img),title('Исходное изображение');
subplot(1,2,2),imshow(im1),title('Полученное изображение').
```

Пример результата работы программы показан на рисунке 2.6.

В конце выполнения программы мы должны сохранить полученное преобразованное изображение. Делается это с помощью функции:

```
imwrite(im1,'save.jpg','JPG').
```

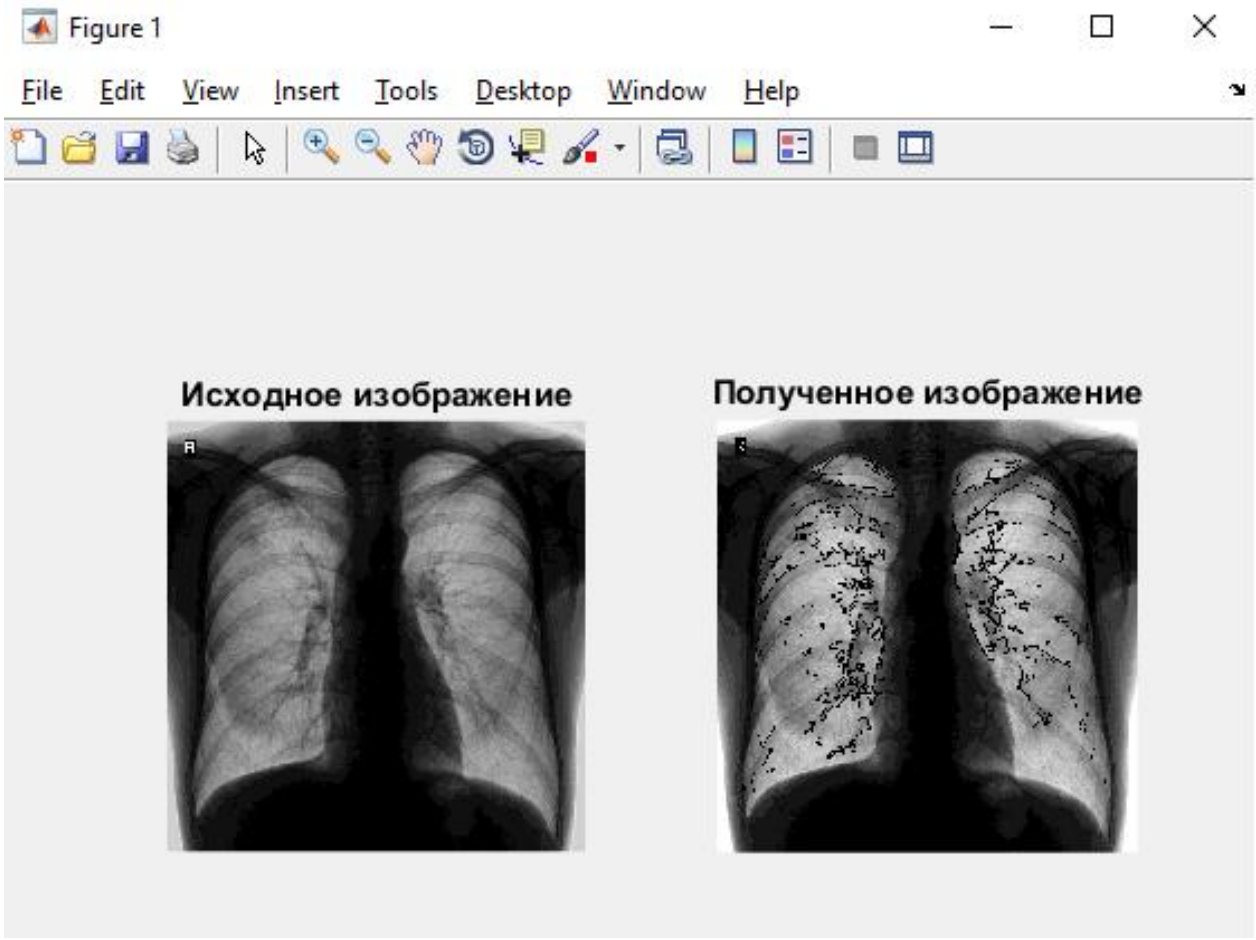

Рисунок 2.6 – Результат работы программы

3 Тестирование и анализ разработанного программного обеспечения

3.1 Тестирование разработанного программного обеспечения

Основной функцией тестирования является обнаружение ошибок. Область включает в себя выполнение кода и изучение аспектов кода – правильно ли работает программа и функционирует ли в соответствии со спецификациями? Рекомендуется начинать тестирование с самых начальных стадий разработки программного обеспечения. Это экономит время и является экономически эффективным.

План тестирования:

- описание тестируемого объекта и целей тестирования;
- перечень этапов тестирования;
- подлежащие оценке количественные и качественные характеристики;
- последовательность проведения и режимы;
- условия и порядок проведения тестирования;
- используемые технические и программные средства;
- методы тестирования (тестовые сценарии).

3.2 Проведение тестирования

Тестирование программы проводилось на компьютере со следующей конфигурацией:

- CPU – Intel Core i5-5200U (2 ядра, 4 потока, частота 2.6 ГГц);
- ОЗУ – 6 Гб;
- GPU – nVidia GeForce 920M (2096 Мб видеопамяти);
- ОС – Windows 10 x64.

Объектом тестирования является программа для выделения границ сосудов на изображении.

Цель тестирования: исключить возможные ошибки при работе программы.

Этапы тестирования:

- отладка;
- минимизация тестовых данных;
- локализация дефектов;
- исправление найденных ошибок;
- тестирование всех входных данных.

Приступая к тестированию программного обеспечения, прежде всего, необходимо провести отладку программы. Отладка – это первоначальное тестирование работоспособности программы, для получения исходной работоспособной версии программы, готовой для последующего тестирования.

Минимизация тестовых данных. Для более быстрого поиска ошибок программы, необходимо сократить набор входных данных до минимума. После того, как будут исправлены все ошибки, и программа будет работать корректно, можно добавлять входные данные.

Локализация дефекта. В нашем случае на вход программы поступают два вида снимков, прямой и боковой проекций. Тестирование программы будет проводиться для каждого вида снимков.

Для начала будем проводить тестирование программы для снимка в прямой проекции при разных заданных пороговых значениях функции canny.

В процессе тестирования рассмотрим статистические ошибки. Примем гипотезу – не все сосуды выделились. Тогда ошибка первого рода – произошло выделение не всех имеющихся сосудов, ошибка второго рода – все сосуды выделены, но кроме них выделились границы других объектов.

В методе Кэнни для классификации перепадов "слабых" границ используется нижний порог. "Слабые" границы отмечаются в результирующем изображении, только если они соединены с "сильными".

При пороге равном 0.1 - B=edge(I, 'canny', 0.1), получим следующий результат.

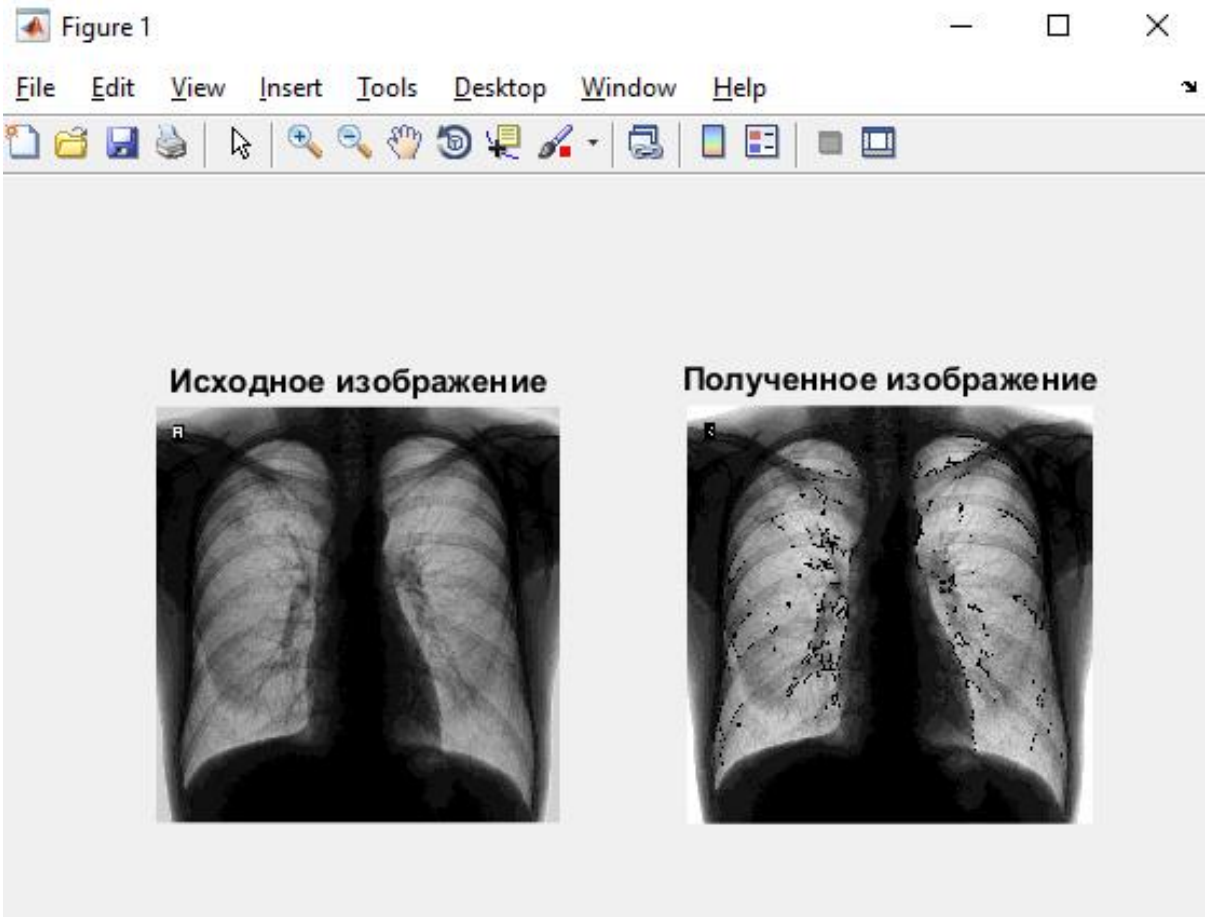

Рисунок 3.1 – Нижний порог 0,1

При выполнении тестирования при пороге 0,1 наблюдается ошибка первого рода. На рисунке 3.1 мы наглядно видим, что сосуды выделились, но многие из них упущены. Для того чтобы исправить полученную ситуацию нужно взять меньший порог, например 0,07.

При выполнении тестирования при входном пороге 0,07 наблюдается ошибка второго рода. Как видно по рисунку 3.2 – кроме сосудов выделились еще многие границы. И программа не смогла их удалить. Значит, нам нужен порог больше 0,07, но меньше 0,1. Возьмем порог равный 0,088 и посмотрим результат выполнения программы.

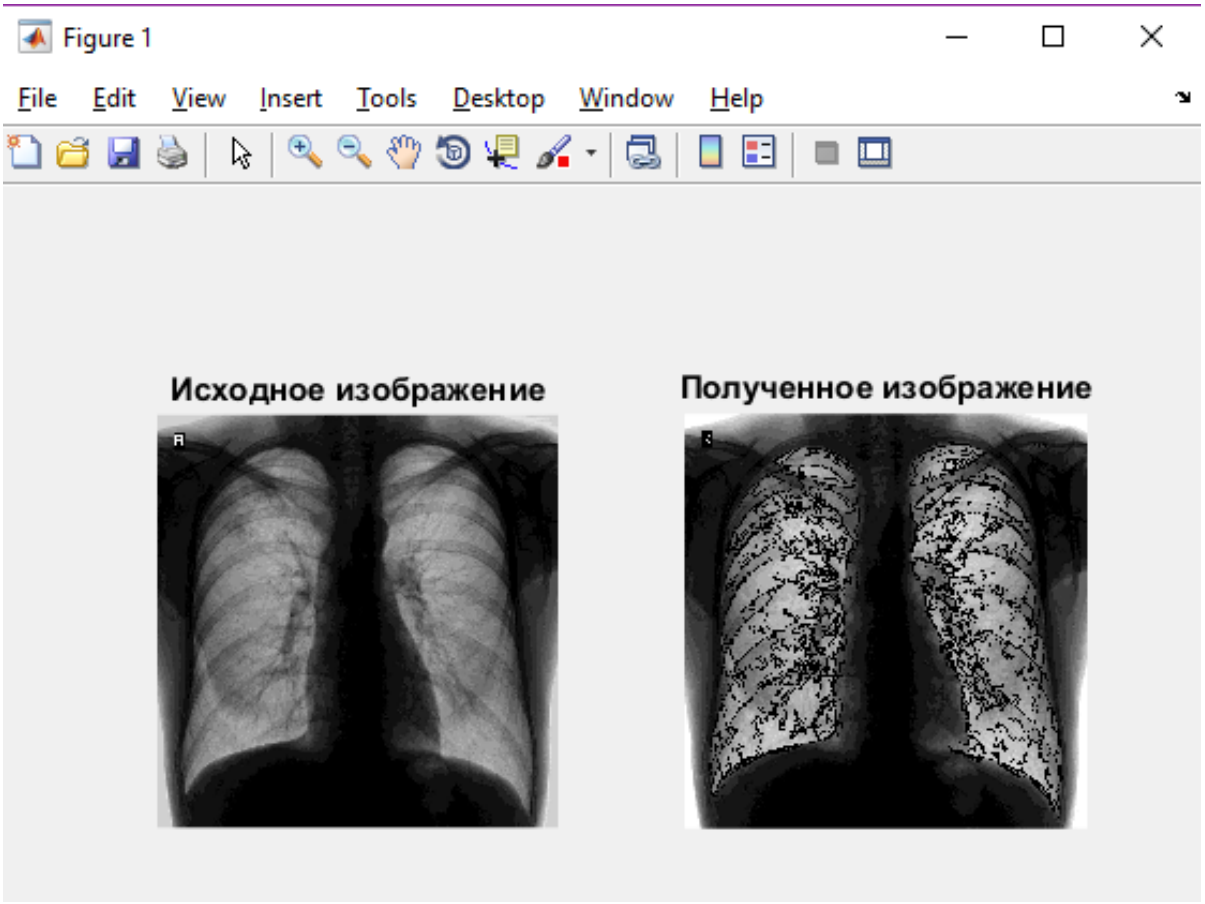

Рисунок 3.2 – Нижний порог 0,07

На рисунке 3.3 продемонстрирован результат программы при пороге функции равном 0,088. Ошибки первого или второго рода при выполнении программы не возникло. Выделение сосудов произошло верно. Однако полного результата данной программе достигнуть не удалось. Для того чтобы программа выделяла полностью все границы сосудов, требуется больше усилий и создание более точных классификаторов.

Теперь рассмотрим работу программы на снимках боковой проекции. На данных снимках присутствуют множество переплетений сосудов. Поэтому для начального порога возьмём стандартный параметр функции равный 0,4.

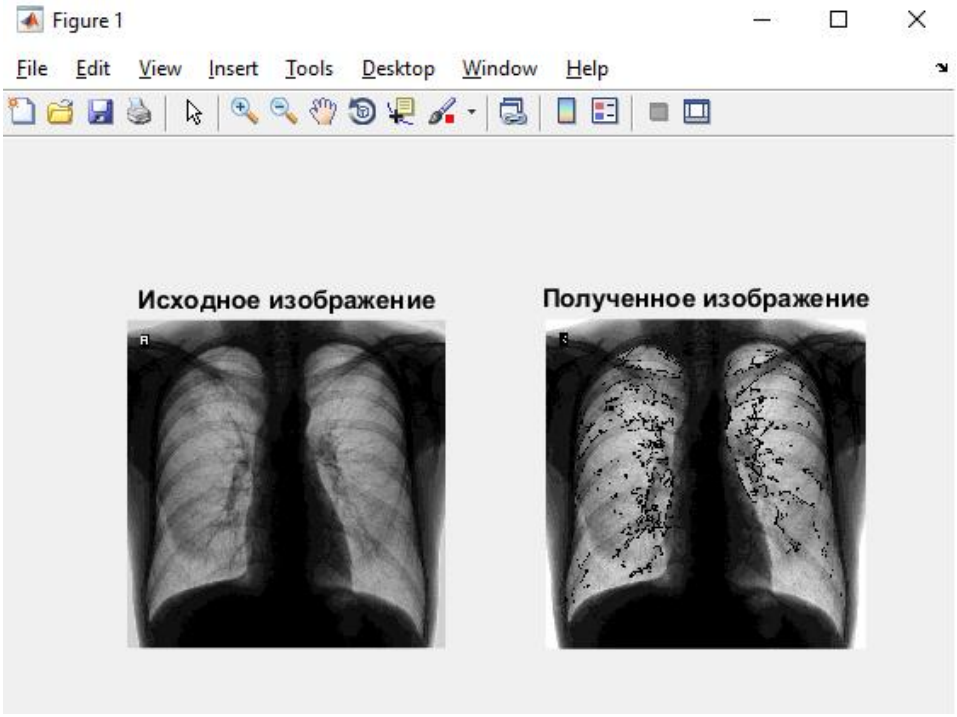

Рисунок 3.3 – Нижний порог 0,088

При выполнении тестирования при входном параметре 0,4 на снимке в боковой проекции наблюдается ошибка первого рода (рисунок 3.4). На рисунке 3.4 видно, что выделение произошло неверно, многие сосуды не выделились. Для исправления данной ситуации необходимо взять входной параметр меньше 0,4, например 0,2 (рисунок 3.5).

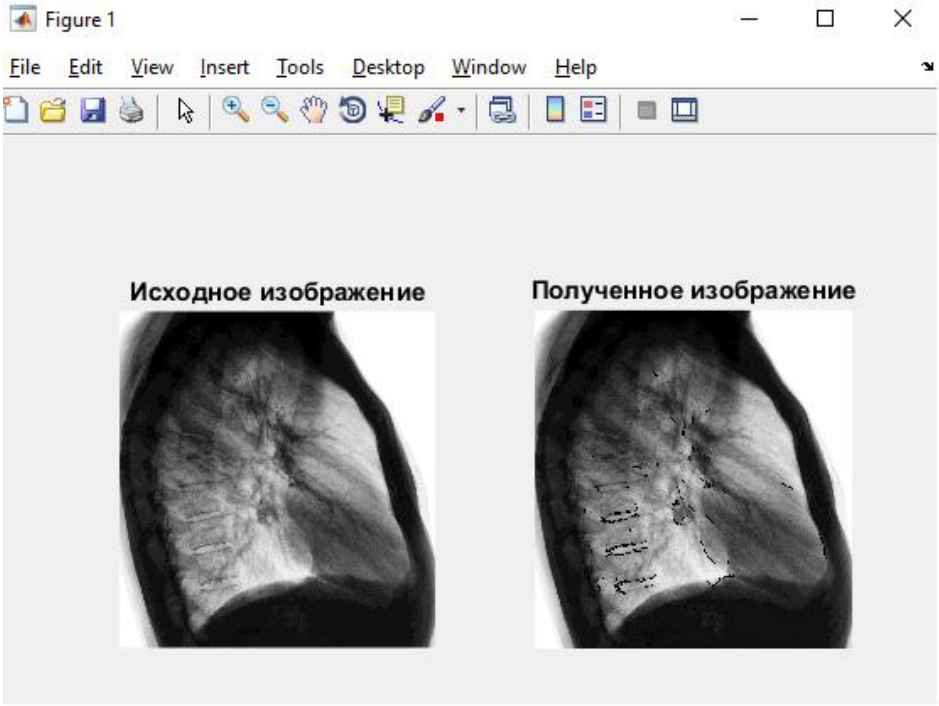

Рисунок 3.4 – Входной параметр 0,4

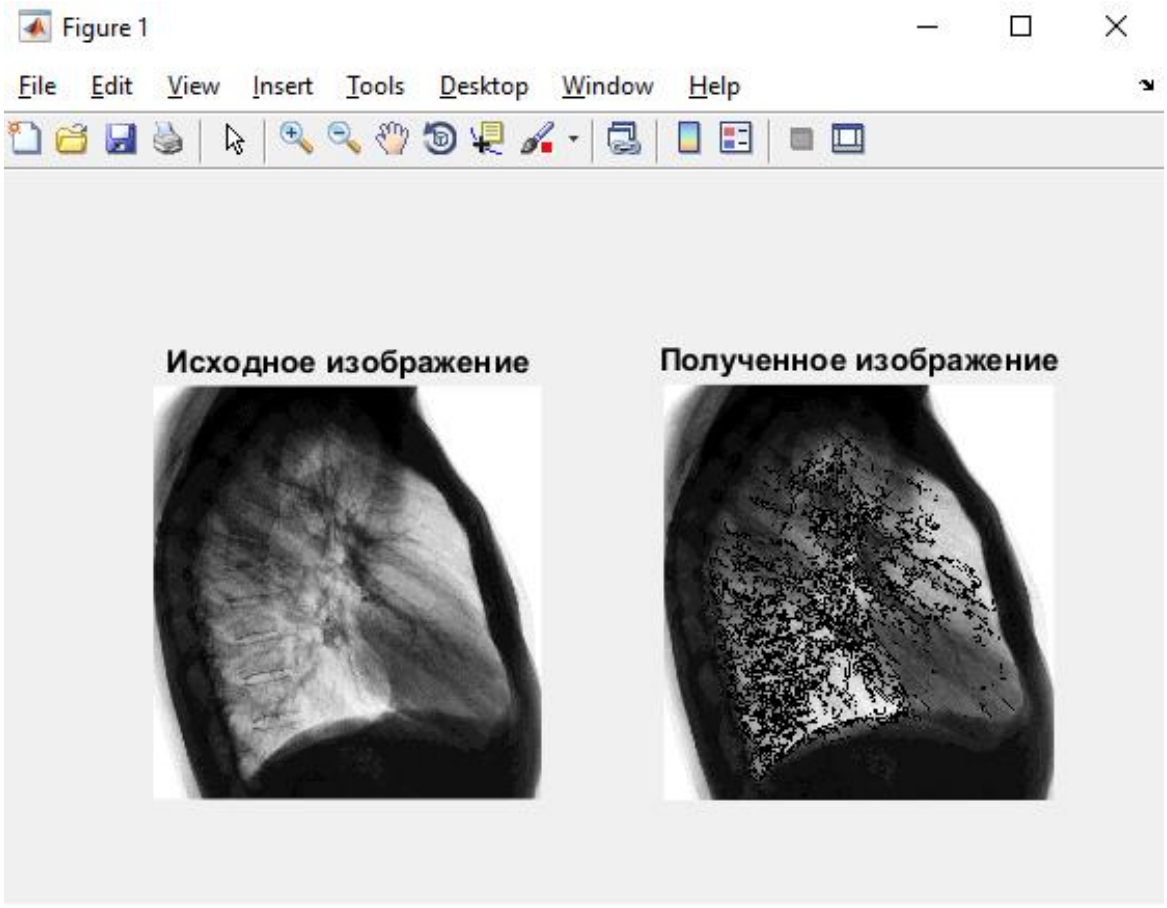

Рисунок 3.5 – Входной параметр 0,2

Входной порог равный 0,2 не дал точного результата, мы получили ошибку второго рода и выделение ложных границ. Для того чтобы избавиться от этих границ, необходимо изменить входной порог и сделать его больше 0,2, но меньше 0,4. Выполнив тестирование с разными пороговыми значениями, я определила максимально эффективный порог равный 0,26.

B=edge(Img, 'canny', 0.26).

На рисунке 3.6 показан результат работы программы при входном пороге равном 0,26. Ошибки первого или второго рода при выполнении программы не возникло. Выделение сосудов произошло правильно. Многие сосуды были выделены, как и должны. Для того чтобы программа работала более точно и выделяла все сосуды требуется создание более точных классификаторов.

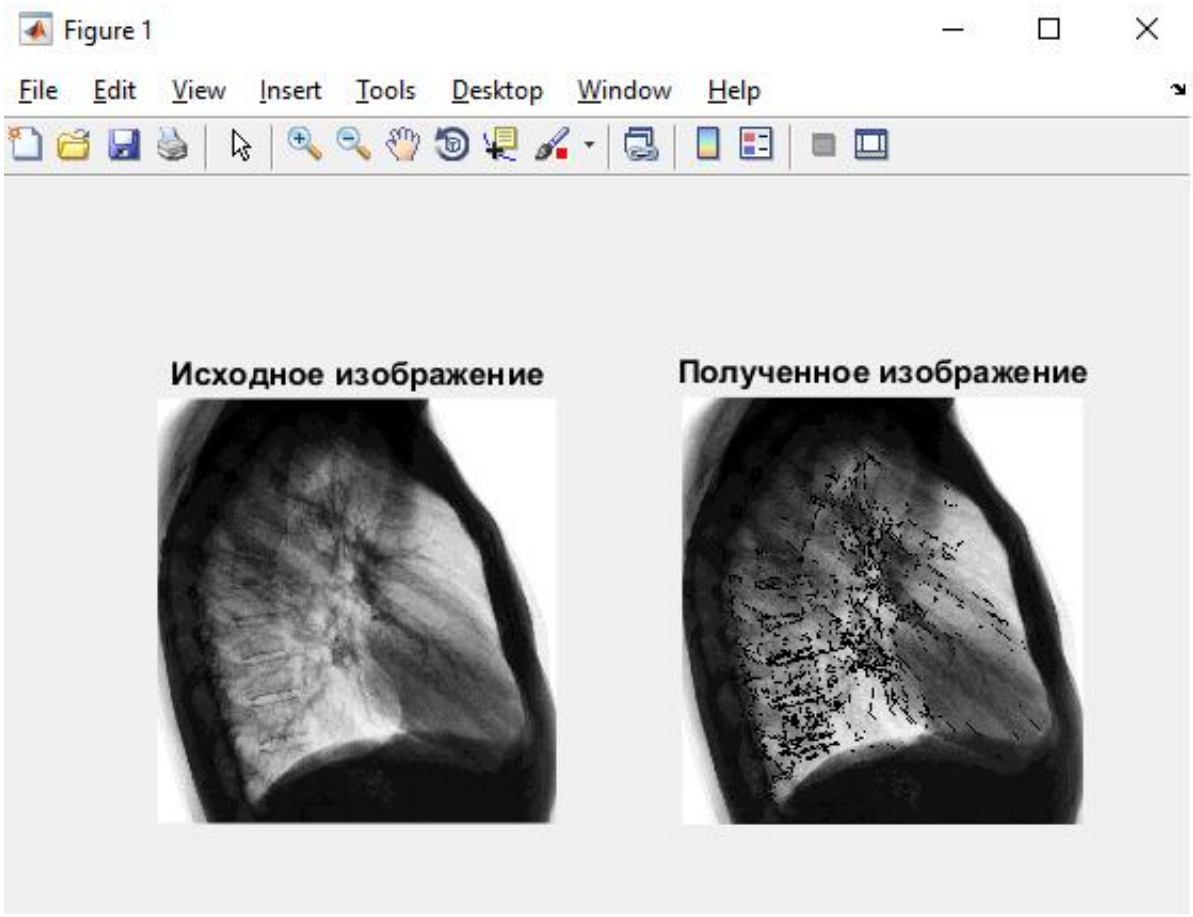

Рисунок 3.6 – Входной параметр 0,26

После того, как мы произвели тестирование на примере выбранных снимков, необходимо провести тестирование на всех имеющихся снимках. На вход мы имеем 12 снимков в прямой проекции и 7 снимков в боковой проекции.

Будем производить тестирование с целью обнаружения ошибок 1 и 2 рода и запишем результаты в таблицу 3.1.

Таблица 3.1 – Ошибки в работе программы

Вид снимка	Ошибки 1-го рода	Ошибки 2-го рода	Общая ошибка
Прямая проекция	0,17 %	0,08 %	0,25 %
Боковая проекция	0,14 %	0,14 %	0,28 %
Все снимки	0,16 %	0,11 %	0,26 %

По результатам тестирования мы выяснили, что при работе программы могут возникать ошибки как первого, так и второго рода. Это объясняется

тем, что не все снимки выполнены в абсолютно одинаковом положении. Для решения данной программы будет целесообразно использовать нейронные сети, либо усовершенствовать программу, путем добавления циклов, с разными входными параметрами функции canny.

Для более точной оценки работоспособности программы возьмём небольшой участок снимка и проведем сравнение работы программы и ручного выделения границ сосудов.

Рисунок 3.7 – Исходное изображение

Вырежем со снимка небольшой участок изображения, содержащий сосуды. Пример снимка приведен на рисунке 3.7.

Далее нам необходимо вручную произвести выделение границ сосудов. Ручное выделение проводилось в стандартной программе Windows – Paint. Для выделения сосудов мы использовали рисование с использованием кисти.

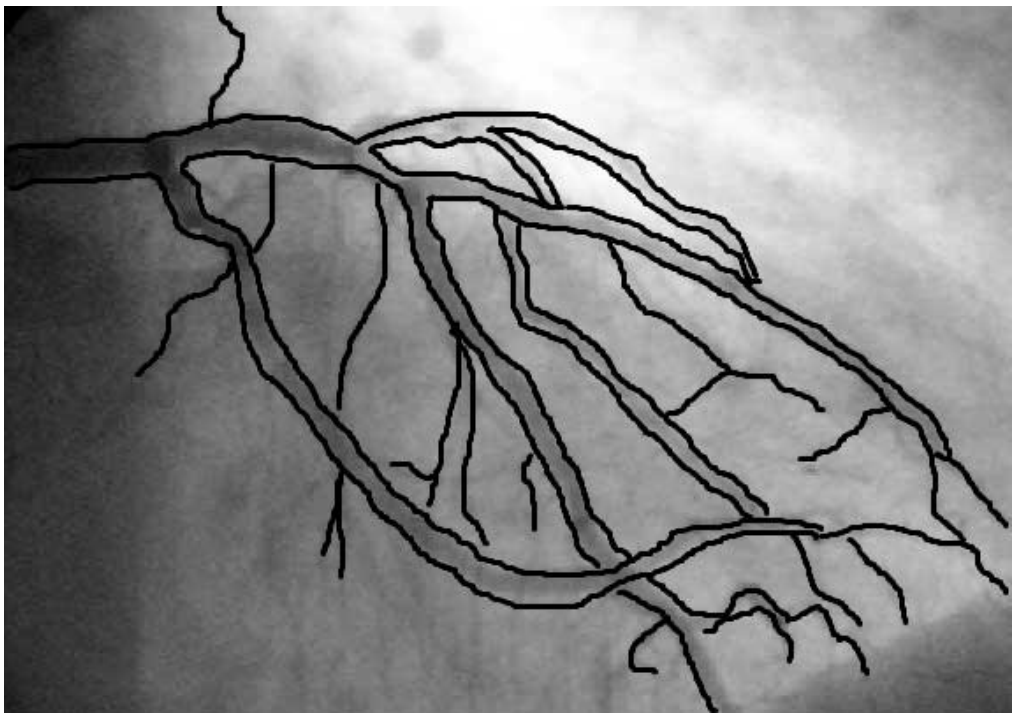

Рисунок 3.8 – Ручное выделение границ сосудов

На рисунке 3.8 показан пример ручного выделения границ. Теперь необходимо произвести выделение при помощи созданной программы.

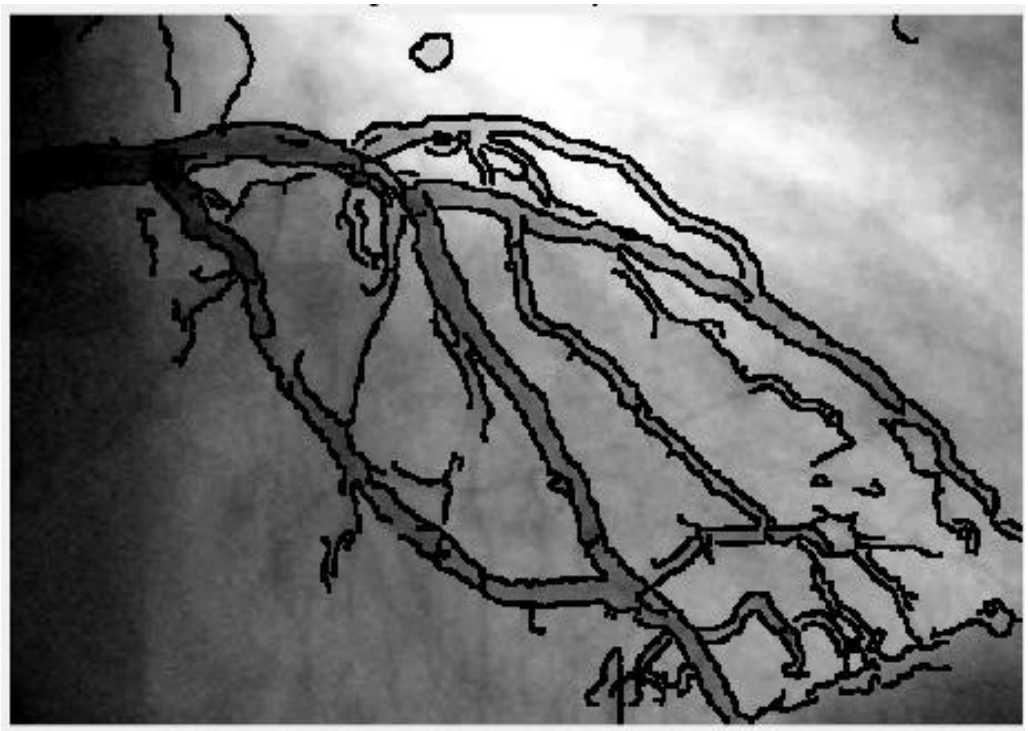

Рисунок 3.9 – Выделение границ программой

Как видно на рисунке 3.9 программа удачно справилась с выделением границ сосудов.

Теперь произведем наложение выделенных границ при помощи программы, они будут серым цветом, на изображение, обработанное вручную, границы чёрного цвета.

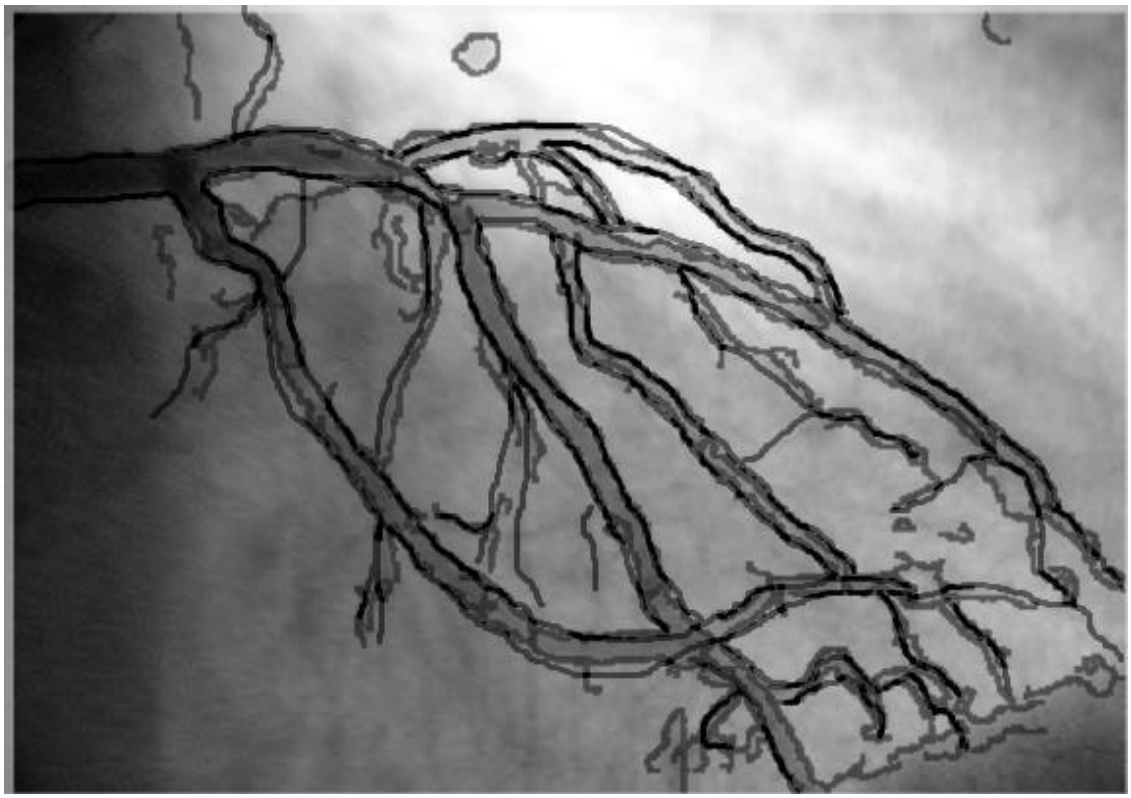

Рисунок 3.10 – Наложение двух снимков

На рисунке 3.10 показан результат наложения двух изображений, программа точно выделила границы, обработанные вручную, а также сосуды, которые при ручном обзоре плохо просматриваются. Это значит, что разработанная программа работает правильно.

ЗАКЛЮЧЕНИЕ

В процессе выполнения бакалаврской работы был спроектирован и реализован алгоритм выделения границ сосудов. В основе выделения границ был использован алгоритм Кэнни.

В работе были рассмотрены и проанализированы три наиболее популярных существующих методов выделения границ: метод Кэнни, Собеля и Робертса. Поскольку алгоритм Кэнни более устойчивый к условиям шума и менее затратный по времени, выбор был остановлен именно на нём.

В ходе работы также были рассмотрены различные языки программирования, выбор остановился на высокоуровневом языке программирования Matlab, главным критерием выбора послужило наличие встроенной функции для реализации алгоритма Кэнни, что позволило сократить длину кода и возможные ошибки в работе программы.

Разработанный алгоритм разделен на две части: алгоритм выделения границ сосудов в прямой проекции и алгоритм выделения границ сосудов в боковой проекции. Данные алгоритмы очень похожи, но различаются численные параметры функций.

Полученная программа способна принимать на вход растровые изображения рентгенограмм в прямой и боковой проекциях, выполнять выделение сосудов, накладывать выделенные линии на исходное изображение и сохранять полученный результат. Разработанная программа выделяет даже те сосуды, которые плохо просматриваются при ручном обзоре, это позволяет врачу более точно увидеть всю картину сосудов и быстрее поставить диагноз, даже при начальном этапе заболевания пациента.

В процессе выполнения работы, мы провели тестирование разработанного программного обеспечения. Тестирование производилось по разным пороговым значениям и входным параметрам. В результате тестирования были выбраны наилучшие значения входных параметров, при

которых программа смогла сделать более точное выделение нужных границ и удалить все лишние границы.

Тестирование показало, что разработанный алгоритм способен достаточно точно выделить границы сосудов на растровом изображении, однако данный алгоритм, возможно, не предоставит полного выделения всех существующих границ сосудов, но сделает это как можно точнее.

Для улучшенного решения алгоритма существуют множество путей развития алгоритма: улучшение качества исходных снимков, усовершенствование алгоритма выделения границ, использование более точных классификаторов.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Научная и методическая литература

1. Безруков Н.С. Автоматизированная система диагностики заболеваний легких/Н.С. Безруков, Е.Л. Еремин, Ю.М. Перельман// Проблемы управления. 2013. №5. С.75-80.
2. Бургенер Ф.А., Кормано М., Пудас Т. Лучевая диагностика заболеваний костей и суставов. / 2017 – 552 с.
3. В. Колдае. Численные методы и программирование: учеб. Пособие. - М.: Форум, 2016. – 336 с.
4. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера. – 2013. – С. 1007.
5. Гуржиев, А.Н. Отображение цифрового рентгеновского снимка на экране компьютера: проблемы и пути их. решения / А.Н. Гуржиев; С. Н; Гуржиев, А. В. Кострицкий // Радиология-практика.: 2014. – 55с.
6. Дьяконов В. П. MATLAB. Полный самоучитель. – М.: ДМК Пресс, 2012. – 768 с.
7. Дьяконов В.П., Matlab обработка сигналов и изображений. — СПб,: Питер, 2014. — 608 с.
8. Дюдин М. В. Методы, модели и алгоритмы анализа и классификации растровых изображений рентгенограмм грудной клетки – Курск: 2016. – 154с.
9. Дюдин, М.В. Способ выделения контура изображения легких на рентгеновском снимке грудной клетки / М.В. Дюдин, В.В. Жилин, П.С. Кудрявцев и др.// Медицинское приборостроение.: 2014. – 114с.
10. Жирков В. Ф. Лекции по дисциплине «Обработка изображений, распознавание образов и мультимедиа». / 2012 – 53 с.
11. Зиборов В.В. Visual C#. / 2013 – 475 с.

12. Корн Дж., Пойнтон К. Рентгенография грудной клетки. / 2017 – 176 с.
13. Л. Шапиро, Дж. Стокман. Компьютерное зрение. / 2013 – 761 с.
14. Рудаков П.И., Сафонов И.В. Обработка сигналов и изображений. MATLAB 6х.- М.: ДИАЛОГ-МИФИ, 2013. – 336 с.
15. Сойфер В.А. «Методы компьютерной обработки изображений» / В.А. Сойфер/. - М.:ФИЗМАТЛИТ, 2013 - 784 с.

Электронные ресурсы

16. MATLAB Toolboxes [Электронный ресурс]. – Режим доступа: http://matlab.exponenta.ru
17. Журавель И.М.Описание ImageProcessingToolbox [Режим доступа]: http://matlab.exponenta.ru/imageprocess/book5/index.php
18. Поиск контуров на изображении/ Хабрахабр [Электронный ресурс]: Оператор Робертса. - Электрон дан. - Режим доступа: http://habrahabr.ru/post/114452/

Литература на иностранном языке

19. Brian Kernighan, Dennis Ritchie. The C++ Programming Language. / 2016 – 288 с.
20. Canny J.F. Finding edges and lines in images. / Master’s thesis, MIT, Cambridge, USA, 2013.
21. Farrell J. Java Programming. / 2015 – 1026 с.
22. Fraz, M.M. Blood vessel segmentation methodologies in retinal images / M.M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A.R. Rudnicka, C.G. Owen, S.A. Barman // Comput Methods Programs Biomed. – 2014. – Vol. 108(1). – P. 407-433
23. Govind B. Chavhan, Bhavin Jankharia. Cross Sectional Anatomy CT and MRI. / 2012 – 263 с.

24. Topal C. Edge Drawing: A combined real-time edge and segment detector / C. Topal, C. Akinlar // Journal of Visual Communication and Image Representation. — 2012. — Vol. 23. — № 6. — Pp. 862–872.
25. Ying-Dong Q. A fast subpixel edge detection method using Sobel–Zernike moments operator / 2005. — 214 c.

ПРИЛОЖЕНИЕ - Листинг программного кода

```
%% Производим очистку памяти
clc;
clear;

%% Выбираем загружаемый файл
Img = imread('2.jpg');
%% Определяем размер изображения
A=size(Img,1);
%% Если изображение в прямой проекции
if A > 2300
 %% Обрабатываем изображение
I = imclearborder(Img); % Подавляем световую структуру изображения
B=edge(I, 'canny', 0.09); % Выделяем границы
se90=strel('line', 5, 90); % Улучшаем выделенные границы
se0=strel('line', 5, 0);
C=imdilate(B, [se90 se0]);

 [M, num]=bwlabel(C, 8);
% Вычисление признаков объектов:
feats = regionprops(M, 'Extent','PixelIdxList');
Extent = [feats.Extent];
id = find(Extent<0.1); % индексы областей, которые нужно удалить
% удаляем области
for i=1:length(id)
    M( feats( id(i) ).PixelIdxList ) = 0;
end
%% Накладываем полученное изображение на исходное
im1=Img;
    [n,m,k]=size(Img);
     for i=1:n
      for j=1:m
       if (M(i,j)~=0)
       im1(i,j,1)= 0;
       im1(i,j,2)= 0;
       im1(i,j,3)= 0;
       end
      end
     end

im1 = rgb2gray(im1);
im1 = imadjust(im1);
subplot(1,2,1),imshow(Img),title('Исходное изображение');
```

```
subplot(1,2,2),imshow(im1),title('Полученное изображение');
  imwrite(im1,'save.jpg','JPG');

%% Если иизображение в боковой проекции
else
     I = imclearborder(Img); % Подавляем световую структуру изображения
B=edge(I, 'canny', 0.26); % Выделяем границы
se90=strel('line', 4, 90); % Улучшаем выделенные границы
se0=strel('line', 4, 0);
C=imdilate(B, [se90 se0]);

[M, num]=bwlabel(C, 8);
% Вычисление признаков объектов:
feats = regionprops(M, 'Extent','PixelIdxList');
Extent = [feats.Extent];
id = find(Extent<0.1); % индексы областей, которые нужно удалить
%% удаляем области
for i=1:length(id)
    M( feats( id(i) ).PixelIdxList ) = 0;
end
%% Накладываем полученное изображение на исходное
im1=Img;
    [n,m,k]=size(Img);
     for i=1:n
      for j=1:m
       if (M(i,j)~=0)
       im1(i,j,1)= 0;
       im1(i,j,2)= 0;
       im1(i,j,3)= 0;
       end
      end
     end

subplot(1,2,1),imshow(Img),title('Исходное изображение');
subplot(1,2,2),imshow(im1),title('Полученное изображение');
  imwrite(im1,'save2.jpg','JPG');
end
```