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 костных тканей на растровых изображениях рентгенограмм»**

Студентка К.Д. Крюкова _____

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 Крюковой К. Д. на тему «Автоматическое выделение границ костных тканей на растровых изображениях рентгенограмм».

Объектом исследования является процесс выделения границ костных тканей на рентгеновских снимках.

Предмет исследования: алгоритмы определения границ костных тканей на снимках.

Цель работы: разработать алгоритмы и их программную реализацию по выделению границ костных тканей на изображениях.

Бакалаврская работа состоит из введения, трех глав, заключения, списка литературных источников и приложения.

Во введении описываются актуальность и новизна поставленной задачи, ставится цель, определяются объект и предмет бакалаврской работы, выявляются задачи.

В первой главе производится математическое описание задачи, обзор и сравнительный анализ возможных путей решения, описание выбранных методов.

Во второй главе происходит обоснование выбора технологии разработки и непосредственно сама разработка.

В третьей главе производится обоснование технологии тестирования, разработка плана тестирования и проведение самого тестирования. В заключении формируются окончательные выводы по рассматриваемой теме.

Выпускная квалификационная работа выполнена на 49 страницах, состоит из введения, трех глав, включающих 30 изображений и 13 формул, заключения, списка используемых источников, включающего 33 источника, и 2-х приложений.

ABSTRACT

The title of the given graduation work is «Automatic Edges Detection of Bones Tissues on Raster Images of Radiographs».

The aim of the work is to develop algorithms and their software implementation for edges detection of bones tissues on images.

The object of the graduation work is edges detection process of bones tissues on X-ray.

The subject of the graduation work is creating algorithms for determining the boundaries of bones tissues on the pictures.

In modern medicine research is being actively conducted on diagnosis of diseases in the initial periods. It's known if earlier disease is detected, the treatment will be more effective and cheaper.

The detection of the boundaries of the object in the image is one of the relevant tasks in digital signal processing. When analyzing images and recognizing objects, the methods and algorithms for borders detection take a significant part as they greatly simplify the work with the image. The developing algorithms will be aimed at maximum detection of all boundaries of objects and at suppression of detected false contours.

The X-ray image is used as input data, which contains the image of the analyzed object. The developed program is capable of performing image analysis, obtained at the input, that is aimed at to facilitate the work of specialists in medical institutions.

The graduation work consists of an explanatory note on 49 pages, introduction, three parts, including 30 figures, 13 formulas, conclusion, the list of 33 references and 2 appendices.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
1 АНАЛИЗ АЛГОРИТМОВ ВЫДЕЛЕНИЯ ГРАНИЦ КОСТНЫХ ТКАНЕЙ НА ИЗОБРАЖЕНИИ 5
1.1 Анализ предметной области 5
1.2 Описание математического и алгоритмического решения задачи 7
1.3 Обнаружение границ на основе нечеткой логики 13
2 РЕАЛИЗАЦИЯ МАТЕМАТИЧЕСКОЙ МОДЕЛИ ВЫДЕЛЕНИЯ ГРАНИЦ КОСТНЫХ ТКАНЕЙ 17
2.1 Обоснование выбора среды программирования 17
2.2 Разработка функций выделения границ 19
2.2.1 Оператор Собеля 19
2.2.2 Оператор Кэнни 23
2.3 Разработка интерфейса программы 26
3 ТЕСТИРОВАНИЕ И АНАЛИЗ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 31
3.1 Описание технологий тестирования 31
3.2 Испытание алгоритмов 32
ЗАКЛЮЧЕНИЕ 36
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 37
Приложение А. Код функции Edge_detection 41
Приложение Б. Код функции canny 46

ВВЕДЕНИЕ

Информационные технологии и сфера их применения стремительно развиваются. Трудно перечислить весь спектр их возможного использования. Одним из важнейших направлений стала медицина. В современной медицине активно проводятся исследования по диагностике заболеваний на начальных периодах. Известно, что чем раньше будет обнаружено заболевание, тем дешевле и продуктивнее выйдет лечение. Многие диагнозы можно поставить, только имея снимок органов человека. Для этого используются такие технологии, как УЗИ, рентген, МРТ и другие. Использование стандартных методов расшифровки снимков имеет серьезные недостатки: необходима высокая сосредоточенность врача, может потребоваться много времени для получения заключения, не исключена возможность ошибки. Поэтому разработка систем, позволяющих автоматизировать расшифровку снимка и, как следствие, обнаружение болезни, является важной и актуальной задачей в современной медицине.

Для такой системы наиболее полезной информацией представляются сведения о границах. Для каждой задачи требуется выделение своих границ, в рамках данной работы будет рассмотрено выделение границ костных тканей. При распознавании объектов и анализе изображений немаловажную роль играют методы и алгоритмы выделения границ, так как они изрядно упрощают работу с изображением.

Научной новизной данной работы является поиск алгоритмов, наиболее точно выделяющие требуемые границы костных тканей. Разрабатываемые алгоритмы будут сосредоточены на максимальном выделении всех границ требуемого объекта и на уменьшение обнаруженных ложных контуров, которые нередко появляются от присутствия импульсного шума.

Программная реализация выделения границ костных тканей представлена в виде программы, написанной на языке высокого уровня Matlab.

В качестве исходных данных используется рентгеновский снимок, содержащий изображение анализируемого объекта.

Объектом исследования является процесс выделения границ костных тканей на рентгеновских снимках.

Предмет исследования: алгоритмы определения границ костных тканей на снимках.

Цель работы: разработать алгоритмы и их программную реализацию по выделению границ костных тканей на изображениях.

Задачи, решаемые в ходе выполнения бакалаврской работы:

- анализ существующих алгоритмов выделения границ на изображении;
- выбор алгоритмов для дальнейшего использования;
- математическое описание алгоритмов;
- реализация поставленной задачи с использованием выбранного языка программирования;
- тестирование разработанных алгоритмов.

1 АНАЛИЗ АЛГОРИТМОВ ВЫДЕЛЕНИЯ ГРАНИЦ КОСТНЫХ ТКАНЕЙ НА ИЗОБРАЖЕНИИ

1.1 Анализ предметной области

Мы живем в век высоких технологий, когда компьютеры прочно вошли в жизнь и стали незаменимыми помощниками в решении разнообразнейших задач. Значительную часть работы, выполняемой раньше человеком, теперь выполняют они, в разы ускоряя ее выполнение и экономя немало средств.

Среди прочего можно выделить работу с изображениями. С ними связан огромный пласт задач, выполняемых компьютером. Трудно найти область, где не используются изображения и какие-либо операции с ними. Медицина, астрономия, радио-, тепло-, гидролокация, телевидение не достигли бы таких высот, как сегодня, без использования компьютерной обработки изображений. Они используются в задачах идентификации личности, распознавании текста и образов, мультипликации, архитектуре и многих других.

Машинное зрение широко используется в медицине. УЗИ, рентгенография – известные каждому примеры такого использования. Но это лишь верхушка айсберга. Так, бинаризация (перевод цветного изображения в монохромное) может помочь в обнаружении опухолей и нарушений кровообращения, а вычитание изображений позволит определить наиболее активные клетки и проследить их передвижение. Обработка снимков со спутников позволяет ученым провести необходимые наблюдения и сделать новые открытия. Они широко используются в навигации, обнаружении лесных пожаров, оценки экологического состояния регионов. Даже в повседневной жизни изображения и задачи их обработки встречаются буквально на каждом шагу. Сегодня трудно найти человека, у которого нет камеры и который хотя бы раз не пользовался услугами фоторедактора.

В рамках данной работы рассматривается применение изображений в медицине, а именно – для поиска границ костных тканей.

В настоящий момент в медицине существует множество различных видов исследований, к которым относятся и такие виды томографии как: УЗИ, рентген, КТ, МРТ, биопсия и другие исследования. Все они предназначаются для диагностики различных заболеваний.

Рентгенологическое исследование органов дает возможность уточнить форму исследуемых органов, их положение, тонус, состояние рельефа слизистой оболочки. Таким образом, можно диагностировать такие заболевания, как пневмония, туберкулез, болезни сердца, переломы.

Исследования по автоматизации рентгенологических исследований ведутся с прошлого века. Стандартные методы расшифровки снимков имеют ряд недостатков, таких как возможная ошибка врача, потеря ценного времени в процессе расшифровки, высокая стоимость услуг эксперта. Автоматизация расшифровки изображений позволяет избежать вышеперечисленных ситуаций, максимально быстро поставить верный диагноз и немедленно приступить к лечению.

Существует довольно много специализированных систем для обработки изображений. Как правило, такие системы узкоспециализированы, и расширение их возможностей сводится фактически к разработке новой системы. Системы «общего назначения» позволяют оперировать обширной библиотекой алгоритмов обработки изображений, но именно эта универсальность является слабым местом таких систем. Они способны работать с разными алгоритмами, но лишь поверхностно, не учитывая особенности конкретного объекта диагностики. Более специализированные системы предназначены для работы с одним конкретным объектом и демонстрируют для них более точный результат.

Наиболее полезной информацией в распознавании объектов на цифровом изображении являются сведения о границах изображения. Под границей изображения в данном случае понимаются линии, проходящие на границах однородных областей, то есть областей, для которых разность яркостей любых двух элементов не превышает заданного порога. В процессе обработки

изображения после предварительной обработки в первую очередь происходит именно поиск границ [1].

Идеей алгоритмов, работающих над выделением границ, является поиск резких изменений яркости на изображении. Результатом работы подобного алгоритма выделения границ является некоторое количество кривых, отображающих границы искомого объекта. В большинстве случаев границы на изображениях выражены недостаточно контрастно, поэтому результат обработки, как правило, включает разрывы в результирующих кривых (фрагментированность) вплоть до полного отсутствия границ [2].

Алгоритмы выделения границ изображений базируются на одном из свойств сигнала яркости: однородность и разрывы [2].

В первом случае используется разбиение изображения на однородные с точки зрения выбранных критериев области [3]. Второй подход заключается в поиске резких разрывов яркости и выделении границ на их основании. Разрывы яркости являются важными простейшими признаками, поскольку часто определяют очертания изображенных объектов. Локальные разрывы в значениях яркости называются яркостными перепадами (яркостными контурами).

1.2 Описание математического и алгоритмического решения задачи

Одним из основных способов выделения границ изображения считается вычисление градиента изображения с использованием разнообразных матриц свертки с дальнейшим сравнением значений градиента с установленным порогом в каждой точке изображения[1].

Подход на основе градиента также называют маской в цифровых изображениях. Дифференциальные приближения в горизонтальном или вертикальном направлении изображения вычисляются с помощью цифровой маски.

P_1	P_2	P_3
P_4	P_5	P_6
P_7	P_8	P_9

Рисунок 1.1 – Интенсивность точек изображения в области 3x3 (маска)

Алгоритм обнаружения границ состоит из следующих шагов [4]:

1. берем цветное изображение;
2. обрабатываем, с целью уменьшения шума, насколько это возможно, без ущерба краям;
3. применяем дифференциацию в целях повышения качества ребер;
4. используем пороговое значение для отклонения шумных краев;
5. получаем изображение после обработки.

Рассмотрим наиболее распространенные методы, вычисляющие градиент изображения – оператор Собеля и детектор границ Кэнни.

Оператор Собеля хорошо известен во всем мире и применяется для многих задач. Строго говоря, оператор использует ядра 3×3, с которыми сворачивают исходное изображение для вычисления приближенных значений производных по горизонтали и по вертикали.

Пусть A – это исходное изображение, а G_x и G_y – два изображения, на которых каждая точка содержит приближённые производные по x и y. Они вычисляются следующим образом [5]:

$$G_x = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} * A, \quad (1.1)$$

$$G_y = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix} * A, \quad (1.2)$$

где $*$ обозначает двумерную операцию свертки.

Координата x здесь возрастает «направо», а y – «вниз». В каждой точке изображения приближённое значение величины градиента можно вычислить путём использования полученных приближенных значений производных:

$$G = \sqrt{{G_x}^2 + {G_y}^2}, \tag{1.3}$$

где G_x и G_y – два изображения, на которых каждая точка содержит приближенные производные по x и y.

Используя эту информацию, можно также вычислить направление градиента:

$$\theta = \arctan\frac{G_y}{G_x}, \tag{1.4}$$

где, к примеру, $\angle\theta = 0$ для вертикальной границы, у которой темная сторона слева.

Этот и следующий метод пространственной фильтрации можно проиллюстрировать следующей схемой (рисунок 1.2) [6].

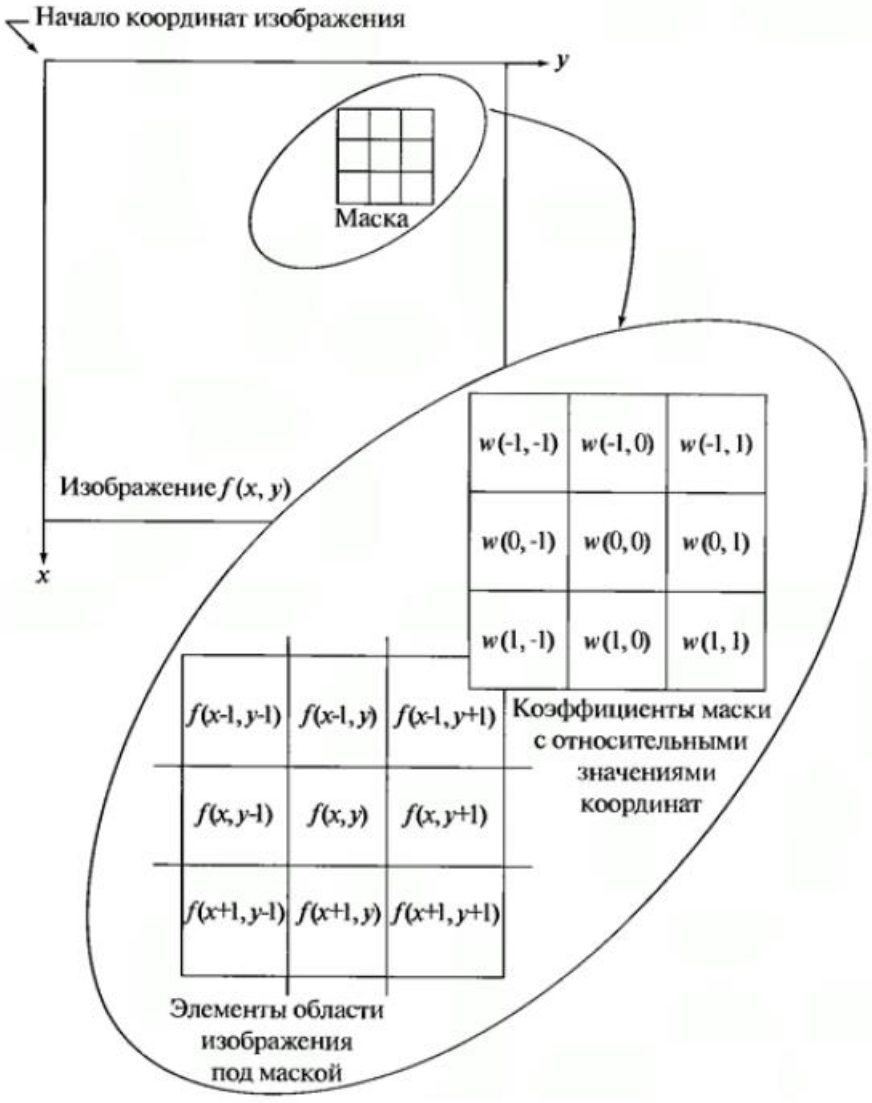

Рисунок 1.2 – Схема пространственной фильтрации

Оператор Кэнни основывается на трех критериях. Основная идея 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 обнаруживать истинные слабые края.

Алгоритм состоит из следующих шагов [7]:

1. сглаживание;
2. поиск градиентов;
3. подавление не-максимумов;
4. двойная пороговая фильтрация;
5. трассировка области неоднозначности.

Детектор границ Кэнни определяет оптимальные края в виде набора критериев, которые обеспечивают максимальную вероятность обнаружения истинных краев при сведении к минимуму вероятность обнаружения ложных краев. Чтобы сгладить изображение, оператор Кэнни использует гауссову свертку, которая контролирует степень сглаживания [8]. Фактически на каждом изображении содержатся шумы. Для того чтобы снизить влияние шума на обнаружение границ, необходимо его уменьшить. Именно для этого используется фильтр Гаусса.

$$G_\sigma = \frac{1}{\overline{2\pi\sigma^2}} exp\ -\frac{x^2+y^2}{2\sigma^2}\ , \qquad (1.5)$$

где σ – степень размытия;

x, y – размер ядра фильтра.

Пример гуассовского фильтра размера 5×5 с σ =1,4 [9]:

$$B = \frac{1}{159}\begin{matrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{matrix} * A, \qquad (1.6)$$

где A – исходное изображение,

* обозначает операцию свертки.

Следует отметить, что выбор размера гауссовского ядра повлияет на производительность детектора. Чем больше размер, тем ниже чувствительность

детектора к шуму. Кроме того, локализация ошибки при обнаружении краев будет немного увеличиваться с увеличением размера ядра фильтра Гаусса. Маска размера 5×5 является хорошим вариантом для большинства случаев, но это также будет зависеть от конкретных ситуаций.

Для определения градиентов на изображении используется оператор Собеля. Вначале производится оценка градиента по направлениям вертикальной и горизонтальной оси. Делается это при помощи двух ядер (1.7) [10]. Следует отметить, что максимумы и минимумы первой производной градиента такие же, как и пересечение нуля второй производной по направлению.

$$K_{Gx} = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}, \qquad K_{Gy} = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}, \tag{1.7}$$

где K_{Gx}, K_{Gy} – две матрицы, где каждая точка содержит приближенные производные по x и по y.

Вычисление значения градиента происходит следующим образом:

$$G = \sqrt{{G_x}^2 + {G_y}^2}, \tag{1.8}$$

где G_x и G_y – два изображения, на которых каждая точка содержит приближенные производные по x и y.

Направление градиента:

$$\theta = \arctan\frac{G_y}{G_x} \tag{1.9}$$

На третьем этапе происходит подавление не-максимумов. Целью данного шага является преобразование «размытых» граней в «четкие». Это достигается путем сохранения локальных максимумов, которые и отмечаются как границы.

На следующем, четвертом, шаге используется двойная пороговая фильтрация. Она необходима для того, чтобы удостовериться, что полученные пиксели действительно обозначают границу и, что никакая ее часть не потерялась. Получается, что чем меньше порог, тем больше границ будет обнаружено, но также и более восприимчивым к шуму окажется результат. С обратной стороны, высокий порог может проигнорировать слабые края или получить границу фрагментами.
Оператор Кэнни использует два порога фильтрации: если значение пикселя выше верхней границы – он принимает максимальное значение, т.е. граница считается достоверной («сильной»), если ниже – пиксель подавляется, точки со значением, попадающим в диапазон между порогов, принимают фиксированное среднее значение.

На заключительном шаге применяется трассировка области неоднозначности. «Сильные» границы считаются распознанными верно. «Слабые» границы попадают в конечный контур, только если они соединены с «сильными». При этом возможность того, что пиксель стал «сильным» за счет воздействия шумов считается маловероятной.

1.3 Обнаружение границ на основе нечеткой логики

Распознавание размытых границ изображений, например, медицинских рентгеновских снимков, – необычайно сложная задача даже для человека, не говоря уже о том, чтобы заставить компьютер понять их содержание. В настоящее время число исследований на эту тему в области искусственного интеллекта постоянно растет, однако проблема не исчерпывается применением только одного какого-либо метода, сомнительно, чтобы можно было найти решение, не используя различные подходы. Одним из таких подходов будет,

как вариант, применение нечеткой логики. В качестве примера ниже описывается исследование, проведенное Пэлом и др. [11].

В качестве примера используются рентгеновские снимки костных образований и предоставляется метод преобразования размытых контуров на снимке в четкие. Для этого используются, к примеру, алгоритм нахождения кривых и петель, предложенный Тау [11], а также сглаживание и сегментация с применением нечетких множеств.

Для начала определяются следующие функции принадлежности трех видов линий (вертикальные, горизонтальные и наклонные), из которых, преимущественно, состоят неясные очертания:

Вертикальные $$\mu_v\ x\ = \begin{matrix} 1 - \frac{1}{m_x}^{F_e},\ m_x\ > 1, \\ 0 \text{ в остальных случаях,} \end{matrix} \quad (1.10)$$

Горизонтальные $$\mu_{\text{н}}\ x\ = \begin{matrix} 1 - \ m_x\ ^{F_e},\ m_x\ < 1, \\ 0 \text{ в остальных случаях,} \end{matrix} \quad (1.11)$$

Наклонные $$\mu_{0\text{в}}\ x\ = \begin{matrix} 1 - \frac{\theta - 45}{45}^{F_e}, 0 <\ m_x\ < \infty, \\ 0 \text{ в остальных случаях,} \end{matrix} \quad (1.12)$$

где $m_x (= tg\theta)$ – наклон линии x (рис. 1.3,а),

F_e – положительный параметр, с помощью которого подбирается степень нечеткости.

«Кривизна» отрезка представляется следующей функцией принадлежности:

$$\mu_{arc}\ x\ = \ 1 - \frac{a}{l}^{F_e}, \quad (1.13)$$

где a – длина линии, связывающей концы отрезка,

l – длина отрезка.

С уменьшением $\frac{a}{l}$ кривизна возрастает (рис. 1.3,б).

Контуры находятся как $M \times N$-мерное изображение серого цвета. При преобразовании изображения направление отрезка представляет собой одно из следующих восьми направлений, в случае преобразование его в одномерную символьную строку (рис. 1.4). При поиске границ изображения с помощью сканирования ищутся элементы, контрастность которых не равняется нулю, притом, стоит заметить, поиск осуществляется в направлении наиболее сильной ориентации. Если направление элементов определять сразу для некоторого количества элементов, а не только для одного, и принимать за направление максимум степени принадлежности, то можно значительно уменьшить объем вычислений и памяти.

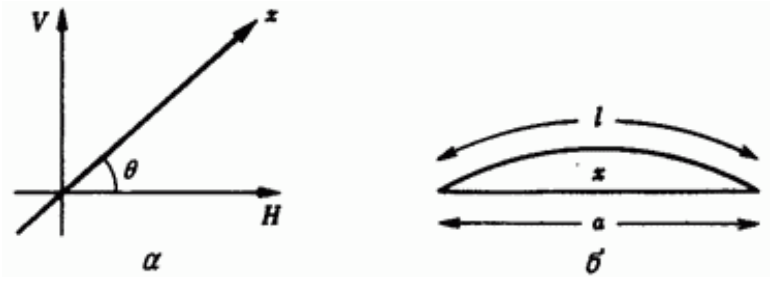

Рисунок 1.3 – Функции принадлежности линии (а) и кривой (б)

Когда существуют два и более сходных сильных направления, следует использовать ранее полученную информацию и выбрать направление наибольшей связности. Полученная информация используется в последующем как развилка. В довершении всего, при достижении конкретной длины контура начало отрезка перемещается. Вследствие этого контуры распознаются в виде петель и предоставляется шанс еще раз трассировать контуры, если в них были пропущены отдельные участки.

Для сглаживания контуров принято применять следующие четыре процедуры, не использующие понятия нечеткости [11]:

1. Если последовательно следуют четыре и более идентичных кода, но между ними встречается один код (или два соседних) другого направления, то этот код либо сменяется на последующий или предыдущий, либо удаляется.

2. Если сперва следуют подряд четыре и более идентичных кода, а следом два других или если значения этих кодов равны 6 и 3, то происходит то же, что и в первом случае.

3. Если два ближайших элемента имеют противоположные направления, то они убираются.

4. Устанавливаются правила вычисления векторной суммы для пары кодов в переходном направлении, которые дают возможность убрать незначительные изменения направления.

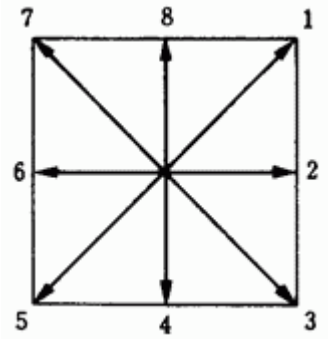

Рисунок 1.4 – Кодирование направлений

2 РЕАЛИЗАЦИЯ МАТЕМАТИЧЕСКОЙ МОДЕЛИ ВЫДЕЛЕНИЯ ГРАНИЦ КОСТНЫХ ТКАНЕЙ

2.1 Обоснование выбора среды программирования

Выбор технологии, с помощью которой будет вестись разработка, важен с точки зрения, как непосредственно разработки, так и дальнейшего внедрения программы и ее поддержки. Каждый язык и стиль программирования обладает своими плюсами и минусами, что может облегчить, или, напротив, усложнить разработку конкретного проекта.

Весь код разработанного программного продукта осуществлен в среде Matlab. Выбор среды программирования основывался на сравнительном анализе между следующими программами, такими как Matlab, Maple, S-PLUS, GAUSS, Mathematica. Matlab – это один из распространенных языков программирования, который довольно часто применяется для выполнения различных математических задач. Благодаря тому, что большинство функций уже встроены в данную среду разработки, происходит сокращение времени при разработке и уменьшение размерности кода. Высокие показатели математических функциональных возможностей среды Matlab представлены на рисунке 2.1 [12].

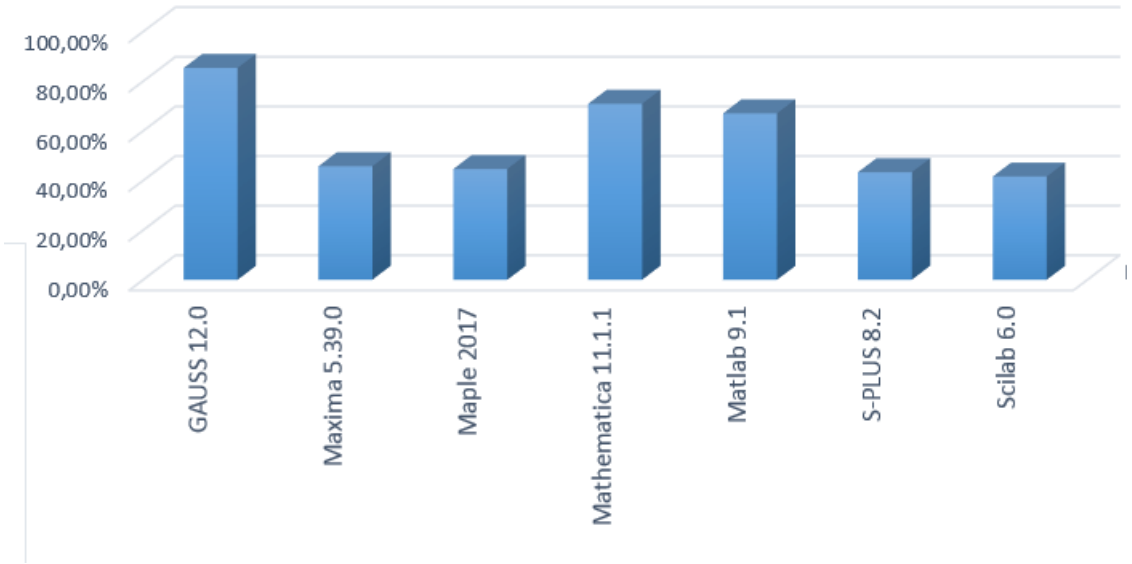

Рисунок 2.1 – Математические возможности

Одним из главных достоинств программной среды Matlab является высокий показатель графических возможностей (рис. 2.2). Довольно часто приходится визуализировать числовые данные (исходные или результирующие) с помощью графических подпрограмм. Поэтому каждая математическая программа должна поддерживать, по крайней мере, основные графические способы представления данных, такие как графики или гистограммы, но и также важные статистические типы диаграмм, такие как дендрограммы, кластерные графики, многомерные графики [12]. В среде Matlab возможна реализация построения изображений как в формате 2D, так и в 3D. Свойства изображений могут задавать пользователи, например, можно изменить цвет графика, толщину линии, контур, форму фигуры. Для комфортной работы с 3D изображениями предусмотрена возможность вращения фигур.

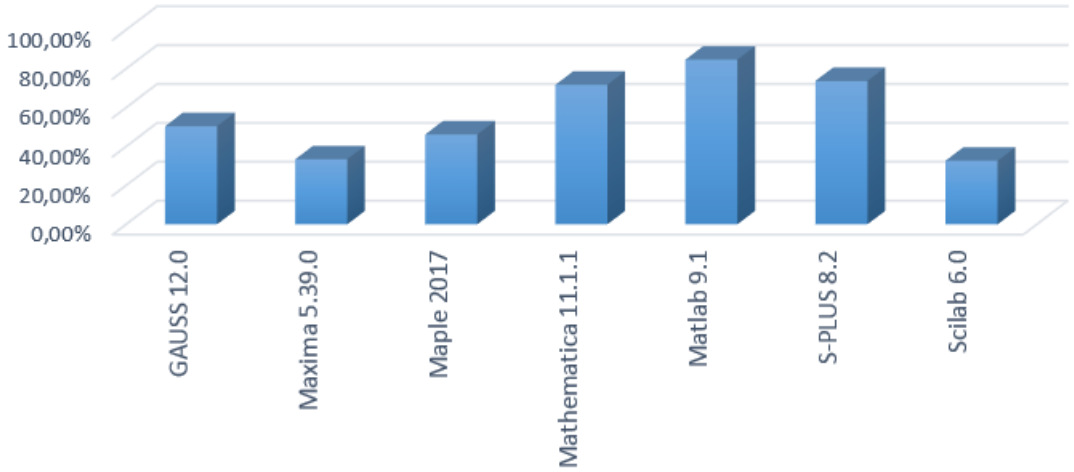

Рисунок 2.2 – Графическая функциональность

Из представленных выше рисунков видно, что Matlab занимает одну из ведущих позиций по сравнению со всеми остальными средами разработки. Все эти факты и послужили решающими аргументами при выборе среды программирования именно в пользу Matlab.

Теперь рассмотрим основные функции, реализованные в данной работе.

2.2 Разработка функций выделения границ

2.2.1 Оператор Собеля

При разработке данного алгоритма использовались следующие функции программной среды Matlab:

- waitbar. Функция waitbar позволяет создать окно с полосой прогресса и заданным текстом над ней, и обновлять ее. Длина полосы прогресса пропорциональна значению x, которое должно быть в пределах от 0 до 1. Выходной аргумент hWB представляет собой указатель на графическое окно с полосой прогресса. Представим фрагмент, демонстрирующий работу функции (рис. 2.3).

```
% Set with dynamic of W
InitScale=3;
Wstep=2;
FinScale=11;
% Set for waitbar
Counter=ceil(FinScale/InitScale)+1;
CounterLoop=0;
hWB = waitbar(0,'Please wait...');
for W=InitScale:Wstep:FinScale
```

Рисунок 2.3 – Фрагмент функции waitbar

- rgb2gray. Функция rgb2gray позволяет создать полутоновое изображение I, преобразуя R-, G-, B-составляющие пикселей исходного полноцветного изображения в подходящие им значение яркости (рис. 2.4).

```
I = rgb2gray(LS);
```

Рисунок 2.4 – Преобразование изображения

- medfilt2. Данная функция, D = medfilt2(S, [m n]), создает полутоновое изображение D, каждый пиксель которого может быть сформирован так. Пиксели первоначального полутонового изображения S, которые соответствуют всем элементам маски размера m×n, образуют упорядоченную последовательность A. Пикселю D(r, c) присваивается значение

медианы последовательности A, где r и c – координаты текущего положения центрального элемента маски. Данная операция применяется нерекурсивно для всех положений маски.

Изображение S на неопределенный срок дополняется необходимым количеством строк и столбов нулевых пикселей при выполнении вычислений, с тем чтобы размеры двух изображений, S и D, были одинаковыми. Формат представления данных результирующего изображения D совпадает с форматом исходного изображения S. Если вектор [m n] при вызове функции D = medfilt2(S) не задан, то в качестве маски фильтра используется маска размера 3 ×3. На следующем изображении представлена данная функция (рис. 2.5).

```
%Подавление шума
I = medfilt2(I,[3 3]);
```

Рисунок 2.5 – Медианная фильтрация

- graythresh. Функция graythresh вычисляет значение глобального порога, используя метод Отсу. Данный метод выбирает порог путем минимизации различных вариантов черных и белых пикселей (рис. 2.6). Значения нормализованных интенсивностей находятся в диапазоне от 0 до 1. У данной функции существуют требования: исходное изображение должно быть представлено неразряженным массивом с форматом представления данных uint8, uint16 или double. Тогда как возвращаемое значение имеет формат представления данных double.

```
(graythresh(I)*.8/10));
```

Рисунок 2.6 – Вычисление глобального порога изображения

- edge. Функция edge выполняет обнаружение границ объекта на изображении, в том числе на участках изображения с перепадом яркости. Для выделения границ на исходном полутоновом изображении I используется Функция BW = edge(I, method). Результатом является бинарное изображение

BW такого же размера, как исходное. Если пиксель I(r, c) принадлежит границе, то BW(r, c) приравнивается к 1. Поиск границ осуществляется несколькими методами. Используемый метод задается в параметре method следующим образом: 'sobel', 'canny', 'prewitt', 'roberts', 'log', 'zerocross'. Если параметр method опущен при вызове функции, то по умолчанию он становится равным 'sobel'.

Для каждого из методов выделения границ можно задать дополнительные параметры. Для этого используются параметры thresh (задает порог с целью определения, принадлежит ли пиксель границе), и P (передает настройки для каждого из методов).

Если параметр thresh не задан, значение порога выбирается автоматически.

Функция BW = edge(I, 'sobel', thresh) использует фильтрацию исходного изображения фильтром Собеля (рис. 2.7); если результирующий пиксель имеет значение больше thresh, он считается относящимся к границе. Если указать дополнительный параметр direction, можно установить, какие границы обнаруживать. Этот параметр может принимать следующие значения:

- 'horizontal' - выделение горизонтальных границ;
- 'vertical' - выделение вертикальных границ;
- 'both' - выделение границ во всех направлениях.

По умолчанию используется значение 'both'.

Значения маски для выделения горизонтальных границ хранятся в функции fspecial, которая находится в edge. Для того чтобы выделить вертикальные границы достаточно транспонировать h.

```
BWs=edge(I, 'sobel', (graythresh(I)*.8/10));
```

Рисунок 2.7 – Фильтр Собеля

На следующем фрагменте представлено наложение контуров на исходное изображение после выделения границ (рис. 2.8).

```
%Наложение контуров на исходное изображение, после выделения границ
im1=LS;
[n,m,k]=size(LS);
for i=1:n
    for j=1:m
        if (BWs(i,j)==1)
            im1(i,j,1)=255;
            im1(i,j,2)=0;
            im1(i,j,3)=0;
        end
    end
end
imshow(im1);
```

Рисунок 2.8 – Наложение контуров

Теперь рассмотрим работу алгоритма, где на рисунке 2.9 будет представлен исходный снимок, а на рис. 2.10 обработанное изображение.

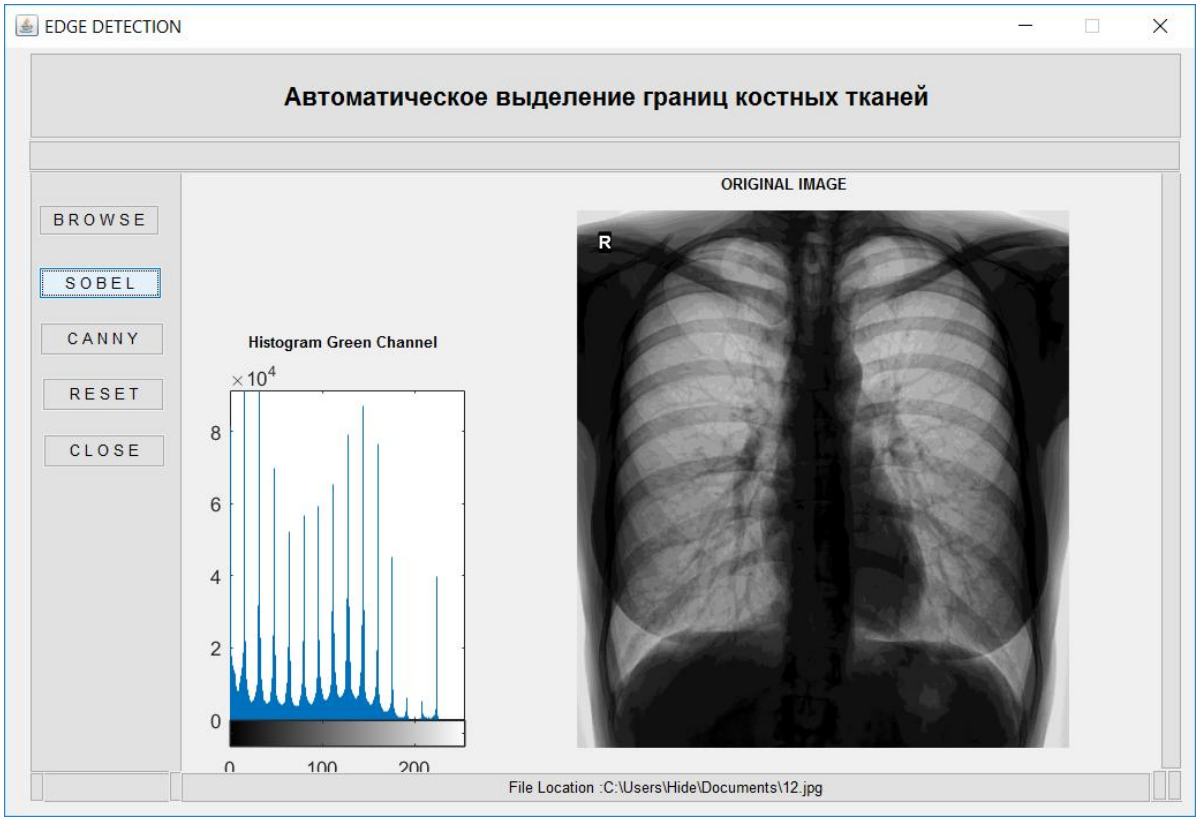

Рисунок 2.9 – Исходное изображение

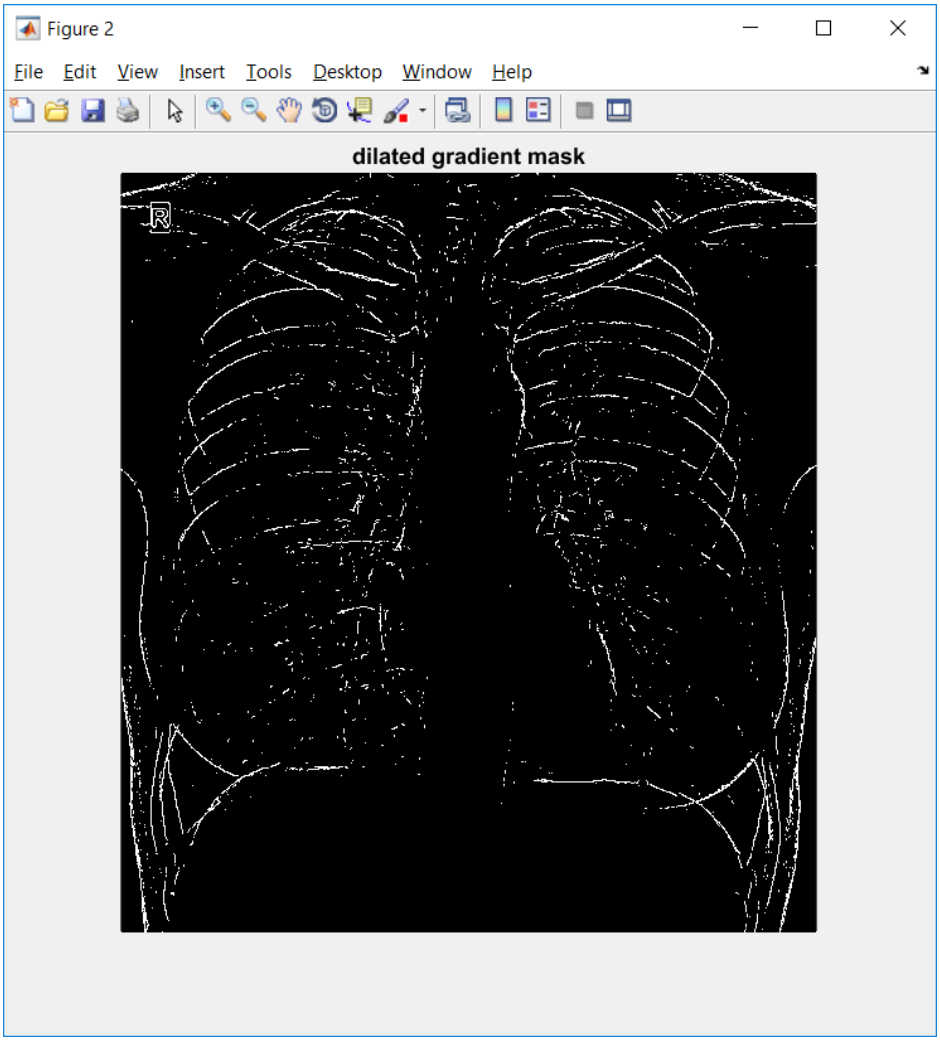

Рисунок 2.10 – Выделение границ фильтром Собеля

2.2.2 Оператор Кэнни

Зададим пороговые значения для нижней и верхней границ (рис. 2.11), которые были выбраны с целью уменьшения влияния шума и получения четких границ на изображении.

```
%Value for Thresholding
T_Low = 0.05275; %0.06675
T_High = 0.02;
```

Рисунок 2.11 – Пороговые значения

Используем фильтр Гаусса для уменьшения шума. Укажем коэффициенты фильтра (рис. 2.12).

```
%Gaussian Filter Coefficient
B = [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 ];
B = 1/159.* B;
```

Рисунок 2.12 – Фильтр Гаусса

Произведем свертку изображения, используя вышеупомянутый фильтр. Далее зададим ядра свертки и вычислим направление градиента (рис. 2.13).

```
%Convolution of image by Gaussian Coefficient
A=conv2(img, B, 'same');

%Filter for horizontal and vertical direction
KGx = [-1, 0, 1; -2, 0, 2; -1, 0, 1];
KGy = [1, 2, 1; 0, 0, 0; -1, -2, -1];

%Convolution by image by horizontal and vertical filter
Filtered_X = conv2(A, KGx, 'same');
Filtered_Y = conv2(A, KGy, 'same');

%Calculate directions/orientations
arah = atan2 (Filtered_Y, Filtered_X);
arah = arah*180/pi;
```

Рисунок 2.13 – Поиск градиента

Вычисление направления вектора градиента, представлено на рисунке 2.14, где он может принимать следующие значения: 0, 45, 90, 135. Если направление края входит в желтый диапазон (от 0 до 22,5 и 157,5 до 180 градусов), то устанавливается значение в 0 градусов. Если попадает в зеленый диапазон (22,5 до 67,5 градуса), то устанавливается в 45 градусов. Если в синий диапазон (67,5 до 112,5 градусов), устанавливается значение в 90 градусов. И, наконец, при любом направлении края в красном диапазоне (112,5 до 157,5 градусов) устанавливается значение в 135 градусов.

```
%Adjusting directions to nearest 0, 45, 90, or 135 degree
for i = 1  : pan
    for j = 1 : leb
        if ((arah(i, j) >= 0 ) && (arah(i, j) < 22.5) || (arah(i, j) >= 157.5) && (arah(i, j) < 202.5) || (arah(i, j) >= 337.5)
            arah2(i, j) = 0;
        elseif ((arah(i, j) >= 22.5) && (arah(i, j) < 67.5) || (arah(i, j) >= 202.5) && (arah(i, j) < 247.5))
            arah2(i, j) = 45;
        elseif ((arah(i, j) >= 67.5 && arah(i, j) < 112.5) || (arah(i, j) >= 247.5 && arah(i, j) < 292.5))
            arah2(i, j) = 90;
        elseif ((arah(i, j) >= 112.5 && arah(i, j) < 157.5) || (arah(i, j) >= 292.5 && arah(i, j) < 337.5))
            arah2(i, j) = 135;
        end;
    end;
end;
```

Рисунок 2.14 – Вычисление угла направления вектора градиента

На следующем рисунке представлен фрагмент подавления не-максимумов. Это выполняется с целью преобразования «размытых» граней в «четкие», т.е. сохраняются локальные максимумы, которые и отмечаются как границы.

```
%Non-Maximum Supression
for i=2:pan-1
    for j=2:leb-1
        if (arah2(i,j)==0)
            BW(i,j) = (magnitude2(i,j) == max([magnitude2(i,j), magnitude2(i,j+1), magnitude2(i,j-1)]));
        elseif (arah2(i,j)==45)
            BW(i,j) = (magnitude2(i,j) == max([magnitude2(i,j), magnitude2(i+1,j-1), magnitude2(i-1,j+1)]));
        elseif (arah2(i,j)==90)
            BW(i,j) = (magnitude2(i,j) == max([magnitude2(i,j), magnitude2(i+1,j), magnitude2(i-1,j)]));
        elseif (arah2(i,j)==135)
            BW(i,j) = (magnitude2(i,j) == max([magnitude2(i,j), magnitude2(i+1,j+1), magnitude2(i-1,j-1)]));
        end;
    end;
end;
```

Рисунок 2.15 – Подавление не-максимумов

Теперь рассмотрим работу алгоритма, где на рисунке 2.9 будет представлен исходный снимок, а на рис. 2.16 обработанное изображение.

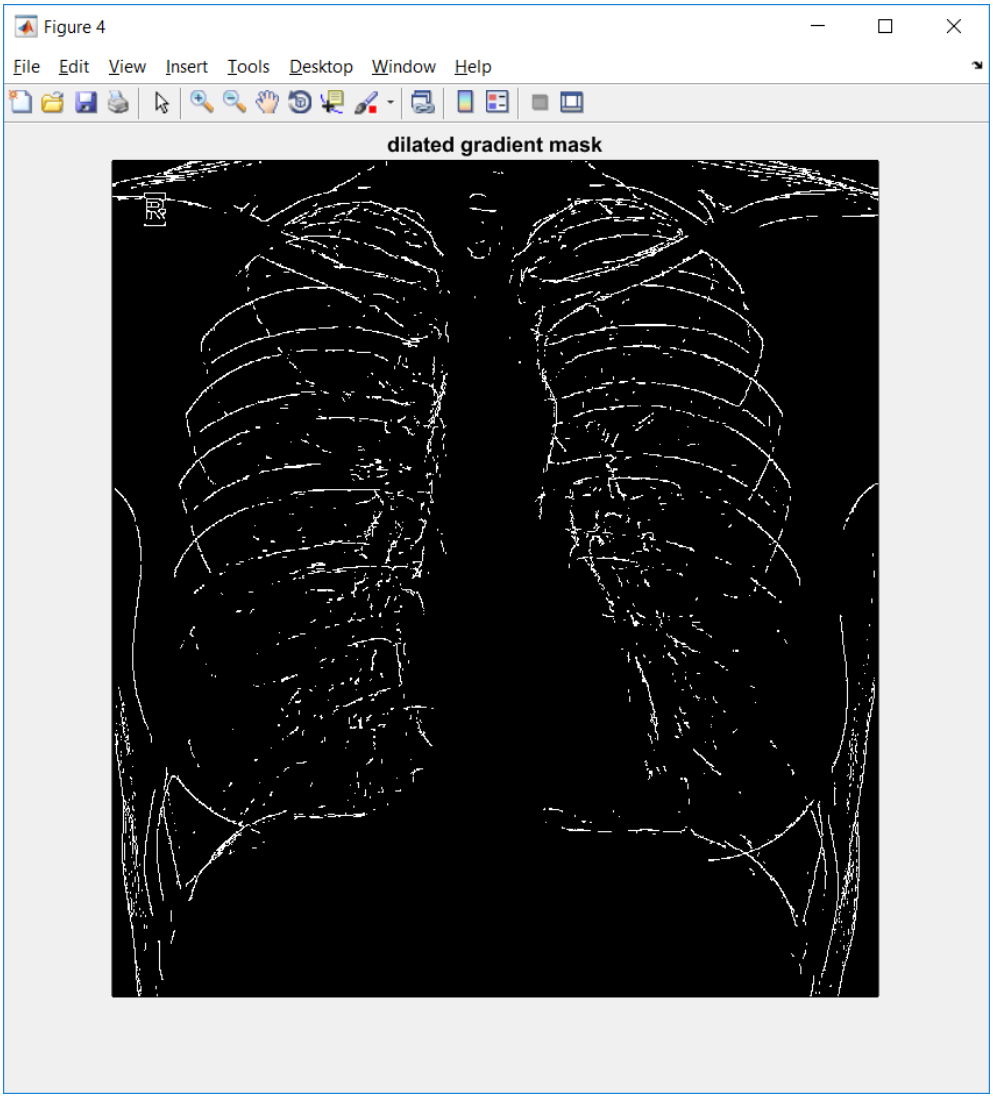

Рисунок 2.16 – Выделение границ оператором Кэнни

2.3 Разработка интерфейса программы

Для создания графического интерфейса программы можно воспользоваться средой GUIDE, которая входит в состав Matlab. Работать в данной среде достаточно просто – элементы управления (раскрывающиеся списки, кнопки и т.д.) размещаются с помощью мыши, а затем программируются события, которые возникают при обращении пользователя к данным элементам управления (рис. 2.17).

Разработанное приложение состоит из одного основного окна, в последствии с помощью которого можно будет выводить графическую и текстовую информацию в отдельные окна.

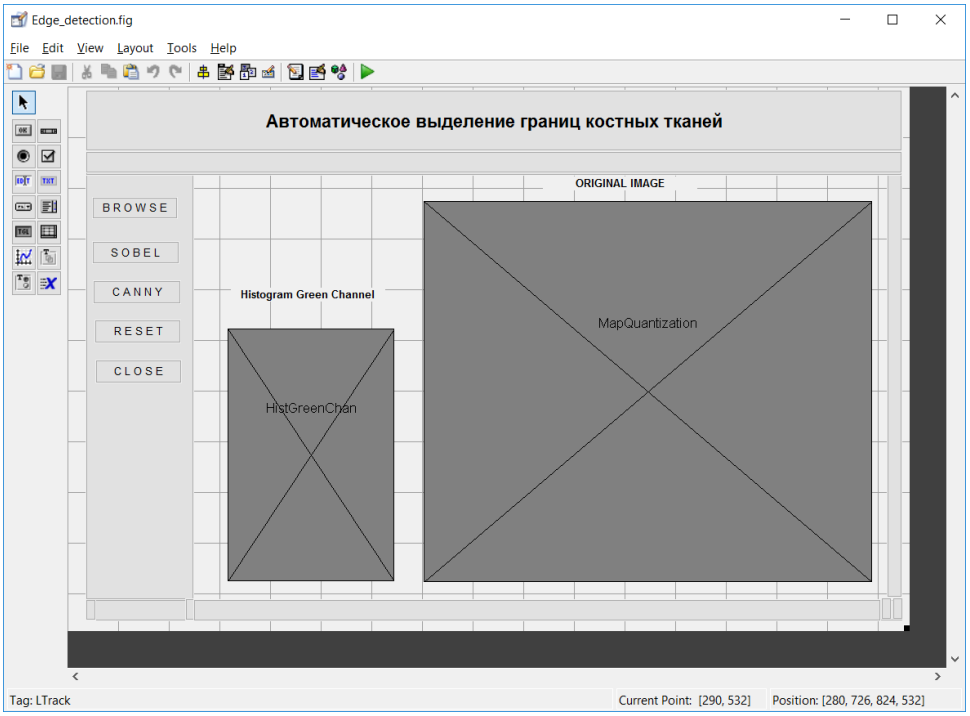

Рисунок 2.17 – Создание главного окна

В данном фрагменте кода запрограммировано событие Callback кнопки BROWSE (рис. 2.18). Функция uigetfile задает открытие файла со входным аргументом – строкой, в которой через точку с запятой записаны шаблоны с нужным расширением: *.tif;*.bmp;*.jpg.

```
[basefilename,path]= uigetfile({'*.tif;*.bmp;*.jpg'},'Open JPG, BMP, TIF Image File');
filename= fullfile(path, basefilename);
```

Рисунок 2.18 – Задание расширений

На следующем фрагменте все той же функции представлено считывание исходного изображения из графического файла и его добавление на форму. Также здесь формируется путь к исходному файлу.

```
filename= fullfile(path, basefilename);
I = imread (filename);
size(I)
MapQuantization = I;
set(LTproject.LTrack,'CurrentAxes',LTproject.MapQuantization);
set(imshow(MapQuantization));
set(LTproject.LTrack,'Userdata',filename);
set(LTproject.MapQuantization,'Userdata',I);
set(LTproject.filebrowse,'String',strcat('File Location : ',path,basefilename));
```

Рисунок 2.19 – Добавление изображения на форму

В обработчике событий кнопки SOBEL происходит считывание изображения с формы в ImageInput и последующая передача его в качестве аргумента в функцию sobel, а также строится гистограмма исходного изображения по зеленому каналу (рис. 2.20). Функция imhist отвечает за построение гистограммы, которая состоит из n столбцов. Значение n можно не указывать, тогда в этом случае будут использоваться значения по умолчанию: n=2 для бинарного изображения или n=256 для полутонового изображения.

```
% --- Executes on button press in proses.
function proses_Callback(hObject, eventdata, handles)
% hObject    handle to proses (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
LTproject = guidata(gcbo);
ImageInput = get(LTproject.MapQuantization,'Userdata');
disp(size(ImageInput));
[GC] = sobel(ImageInput);
GreenChan=GC;
axes (handles.HistGreenChan);
imhist(GreenChan);
```

Рисунок 2.20 – Обработчик событий кнопки SOBEL

Похожим образом был реализован обработчик кнопки CANNY.

Следующий фрагмент отвечает за сброс всех полученных результатов и возвращение к начальному окну (рис. 2.21).

```
% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata, handles)
% hObject    handle to reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
LTproject = guidata(gcbo);
%set(project.CitraAsli, 'String', ''); ;
set(gcf,'WindowStyle','Normal')
frames = java.awt.Frame.getFrames();
frames(end).setAlwaysOnTop(0);
ReBut = questdlg('Are you really want to reset this application?','Reset','Yes','No','default');
% frames(end).setAlwaysOnTop(1);
switch ReBut
case {'No'}
%     set(gcf,'WindowStyle','Modal')
%     frames = java.awt.Frame.getFrames();
%     frames(end).setAlwaysOnTop(1);
% take no action
case 'Yes'
closeGUI = LTproject.LTrack; %handles.LTproject is the GUI figure
guiPosition = get(LTproject.LTrack,'Position'); %get the position of the GUI
guiName = get(LTproject.LTrack,'Name'); %get the name of the GUI
close(closeGUI); %close the old GUI
eval(guiName) %call the GUI again
end
```

Рисунок 2.21 – Сброс приложения

Похожим образом реализуется обработчик события close_Callback (рис. 2.22).

```
% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)
% hObject    handle to close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
LTproject = guidata(gcbo);
set(gcf,'WindowStyle','Normal')
frames = java.awt.Frame.getFrames();
frames(end).setAlwaysOnTop(0);
pos_size = get(LTproject.LTrack,'Position');
% Call modaldlg with the argument 'Position'.
button = questdlg('Are you really want to close this application?','Close','Yes','No','default');
% Set the figure icon
warning('off','MATLAB:HandleGraphics:ObsoletedProperty:JavaFrame');
jframe=get(handles.LTrack,'javaframe');
jIcon=javax.swing.ImageIcon('dental-icon.gif');
jframe.setFigureIcon(jIcon);
switch button
case {'No'}
case 'Yes'
close;
end
```

Рисунок 2.22 – Обработчик закрытия приложения

С учетом вышеизложенного главное окно разработанного приложения выглядит следующим образом (рис. 2.23).

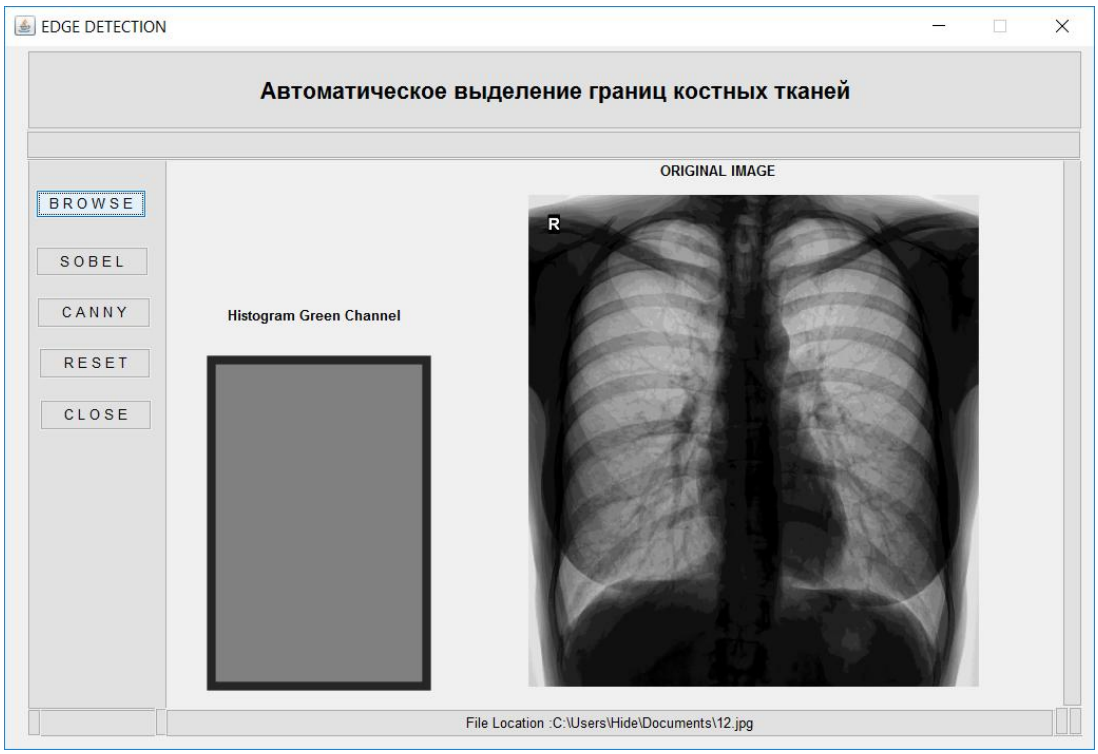

Рисунок 2.23 – Главное окно программы

3 ТЕСТИРОВАНИЕ И АНАЛИЗ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1 Описание технологий тестирования

Тестирование программного обеспечения – важнейший этап в разработке любого ПО. Целью тестирования является проверка возможностей ПО, соответствие ожидаемым результатам. Тестирование должно проводиться на каждом этапе разработки, чтобы избежать появления еще больших ошибок. Стратегии тестирования направлены на устранение багов и ошибок в коде, обеспечение точной и оптимальной производительности программного продукта.

Выделяют две наиболее популярных технологии тестирования [13].

Тестирование методом черного ящика осуществляется без знаний внутренней работы системы. Инженер по тестированию имеет доступ к ПО только через те же интерфейсы, что и пользователь. Как правило, ведется с использованием спецификаций или иных документов, описывающих требования к системе.

Тестирование методом белого ящика учитывает внутреннее функционирование и логику работы кода. Для выполнения этого теста тестировщик должен иметь доступ к исходному коду программы.

Тестирование методом серого ящика – это среднее между тестированием черного и белого ящика. Тестировщик имеет доступ к исходному коду, но при непосредственном выполнении тестов доступ к коду, как правило, не требуется [13]. Тестирование проводится конечным пользователем.

План тестирования предназначен для оценки качества и соответствия программного обеспечения предъявляемым. Для составления плана нужно ответить на следующие вопросы [14]:

– Что надо тестировать? Объектом тестирования являются алгоритмы, выделяющий границы костных тканей на изображении.

– Что будет тестироваться? Качество работы разработанных алгоритмов.

– Как будет тестироваться? Путем сравнения субъективной оценки результатов.

Критерии окончания тестирования: результаты тестирования удовлетворяют критериям качества продукта; во время тестирования не обнаружено багов.

3.2 Испытание алгоритмов

Основной функцией тестирования является обнаружение ошибок. Область включает в себя выполнение кода в различных средах 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.

Тестирование методом черного ящика подразумевает тестирование лишь видимой пользователю части программы без анализа кода. Другими словами, тестирование приложения с использованием разных изображений. В проведении тестирования использовался рентгеновский снимок. Посмотрим на работу программы с первым изображением (рис. 3.1).

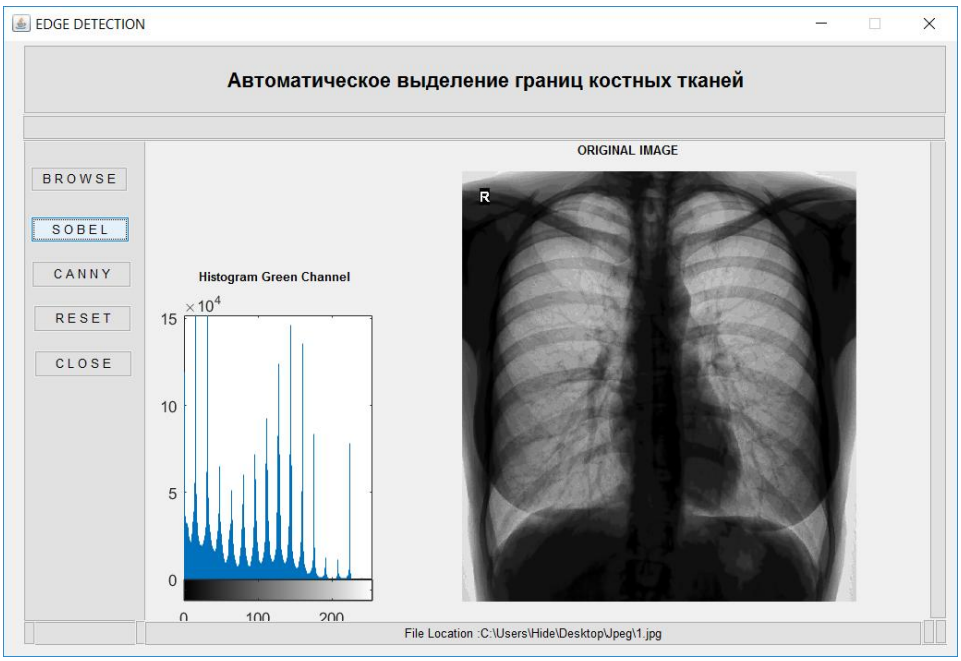

Рисунок 3.1 – Начальное изображение

И на рисунке 3.2 представим рентгеновский снимок после обнаружения границ фильтром Собеля.

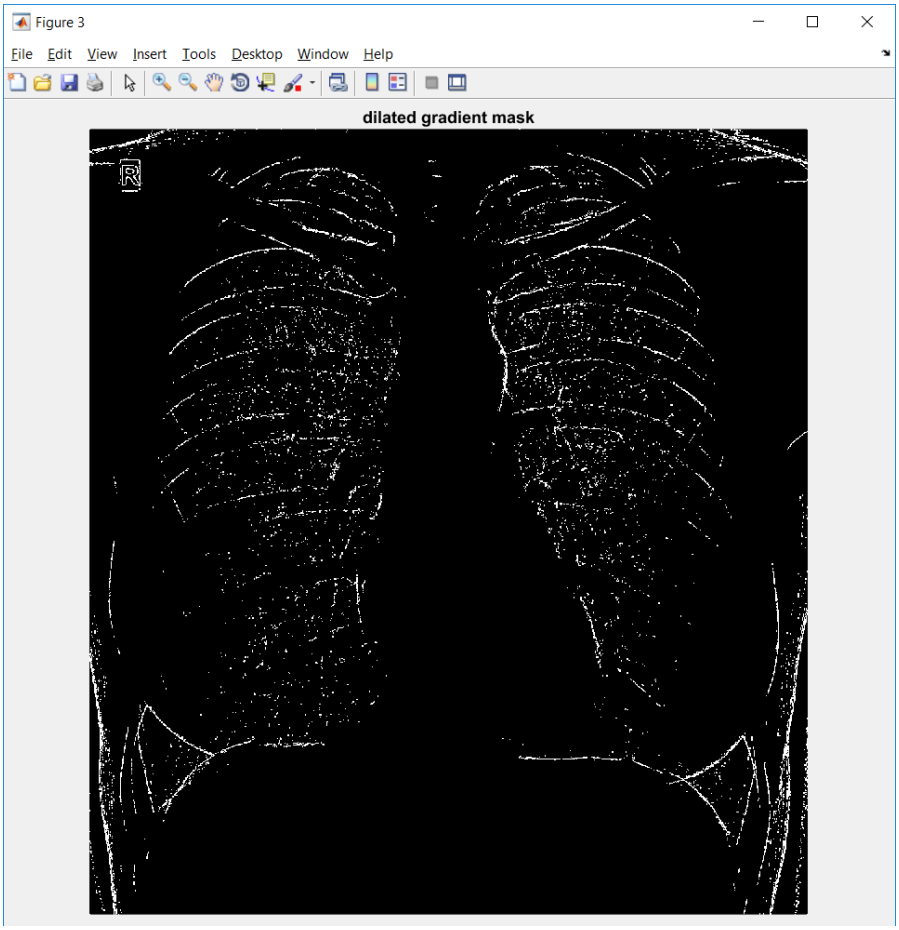

Рисунок 3.2 – Изображение после обработки

Как видно из рисунка 3.2 на изображении присутствуют шумы, фильтр Собеля довольно неустойчив при работе в условиях шума. Посмотрим на работу программы со вторым алгоритмом, оператором Кэнни (рис. 3.3).

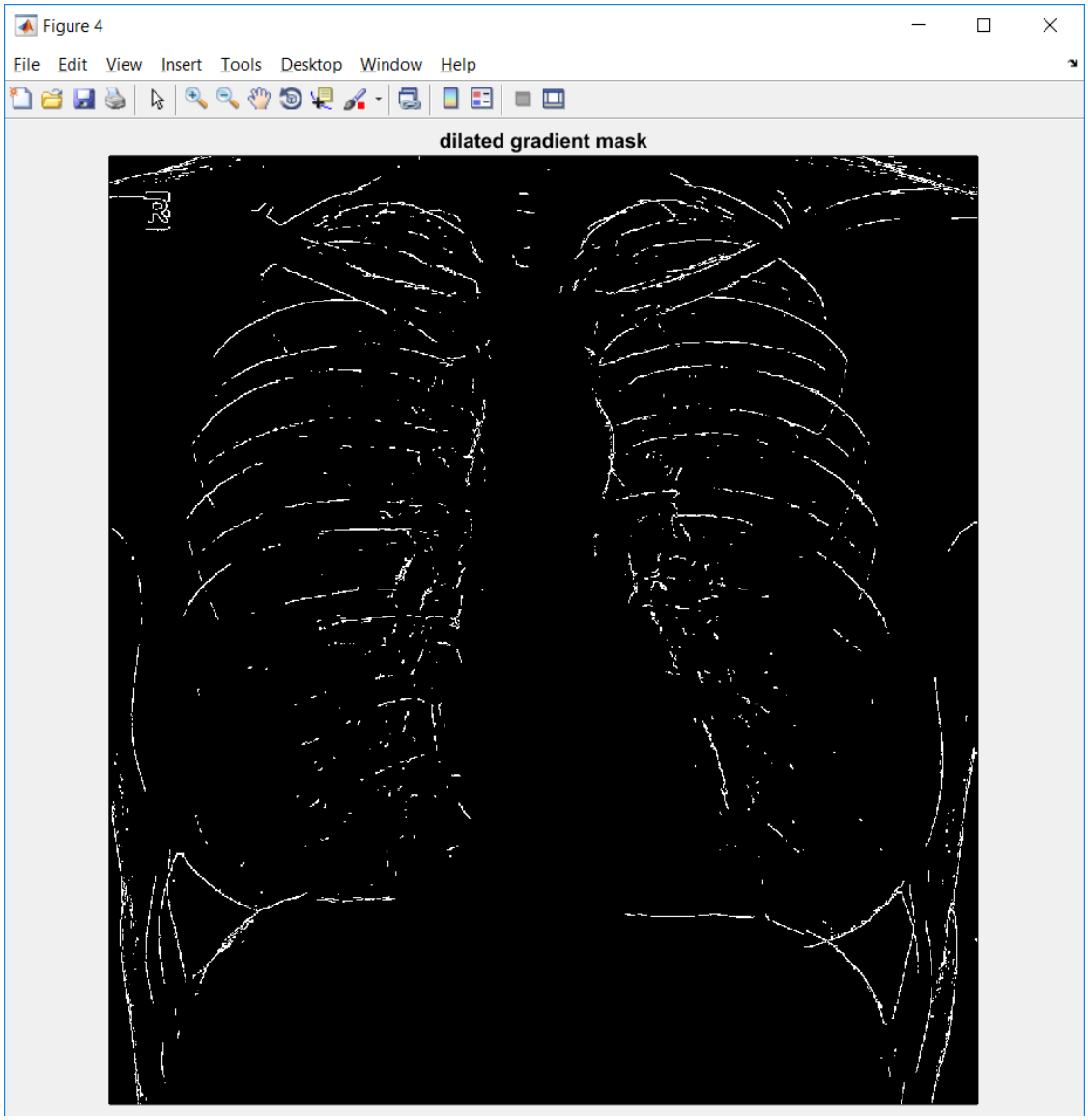

Рисунок 3.3 – Выделение границ оператором Кэнни

Можно увидеть, что на первом изображении присутствуют шумы. Медианный фильтр помог уменьшить уровень шума, но он все равно присутствует в небольшом количестве. Если же говорить про работу второго алгоритма, то на нем его значительно меньше. Фильтр Гаусса, включенный в алгоритм Кэнни, повысил устойчивость результатов. Двойная пороговая фильтрация позволила избавиться от слабых границ. В целом поставленную

цель можно считать достигнутой: разработанные алгоритмы показали свою работоспособность при работе с разными изображениями.

ЗАКЛЮЧЕНИЕ

В ходе выполнения бакалаврской работы были изучены и реализованы алгоритмы выделения границ на рентгеновских снимках. Для этого были рассмотрены и проанализированы наиболее популярные существующие методы обнаружения границ: оператор Собеля, как простой в разработке и используемый во многих задачах прикладного характера алгоритм, и детектор границ Кэнни, который считается одним из лучших детекторов. Для реализации использовалась среда Matlab.

Тестирование на рентгеновских снимках показало, что оба алгоритма справились со своей работой достаточно качественно, границы на изображениях получились четкими. Также тестирование указало на слабые стороны алгоритмов, в частности на присутствие шума после обработки изображений.

Медицинские исследования с помощью современных методов визуализации изображений позволяют заглянуть внутрь объектов живого организма и диагностировать его состояние. Эта задача решается с помощью цепочки этапов обработки изображения с целью анализа и распознавания объектов. Рассмотренные в работе алгоритмы выполнили эту задачу. Тем не менее, основываясь на результатах работы данных алгоритмов, в рамках конкретной задачи лучше себя проявил оператор Кэнни.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Об одном методе выделения контуров на цифровых изображениях [Электронный ресурс]. – Режим доступа: http://cyberleninka.ru/article/n/ob-odnom-metode-vydeleniya-konturov-na-tsifrovyh-izobrazheniyah
2. Реферат – Разработка и анализ алгоритма выделения контура на полутоновом изображении при условии слабоконтрастных границ объектов [Электронный ресурс]. – Режим доступа: http://masters.donntu.org/2014/fknt/metelytsia/diss/index.htm
3. Грузман И.С. Цифровая обработка изображений в информационных системах / И.С. Грузман, В.С. Киричук, В.П. Косых, Г.И. Перетягин, А.А. Спектр // Научное пособие. – Новосибирск: Изд-во НГТУ, 2013. – С. 125–139.
4. Methods of Image Edge Detection: A Review [Electronic resource]. – Mode of access: https://www.omicsgroup.org/journals/methods-of-image-edge-detection-a-review-2332-0796-1000136.pdf
5. Sobel operator [Electronic resource] : [Оператор Собеля]. – Mode of access: https://en.wikipedia.org/wiki/Sobel_operator
6. Исследование методов выделения и применения опорных контуров с целью распознавания лиц [Электронный ресурс]. – Режим доступа: conf58.mipt.ru/static/reports_pdf/532.pdf
7. Детектор границ Канни [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/114589/
8. Comparison between Various Edge Detection Methods on Satellite Image [Electronic resource]. – Mode of access: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.413.7654&rep=rep1&type=pdf
9. Canny edge detector [Electronic resource] : [Оператор Кэнни]. – Mode of access: https://en.wikipedia.org/wiki/Canny_edge_detector

10. Sobel Operator and Canny Edge Detector [Electronic resource]. – Mode of access:http://www.egr.msu.edu/classes/ece480/capstone/fall13/group04/docs/danapp.pdf

11. Обнаружение размытых границ [Электронный ресурс]. – Режим доступа: http://stu.alnam.ru/book_pfuzzy-49

12. Comparison of mathematical programs for data analysis [Electronic resource]. – Mode of access: http://old.exponenta.ru/educat/free/compare/ncrunch.pdf

13. Стратегии тестирования [Электронный ресурс]. – Режим доступа: http://www.4stud.info/software-construction-and-testing/lecture10.html#black-box

14. Тест План (План тестирования) [Электронный ресурс]. – Режим доступа: http://www.protesting.ru/testing/plan.html

15. Построение автоматизированной системы определения контура объекта на примере изображения клеток [Электронный ресурс]. – Режим доступа: http://masters.donntu.org/2005/kita/tribrat/diss/index.htm

16. Распараллеливание алгоритма выделения границ объектов на основе структурно-графического представления [Электронный ресурс]. – Режим доступа: http://www.lib.tpu.ru/fulltext/v/Bulletin_TPU/2013/v323/i5/26.pdf

17. Методы нахождения границ изображения [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/128753/

18. Обзор языков программирования [Электронный ресурс]. – Режим доступа: http://bourabai.ru/alg/classification04.htm

19. Технологии тестирования программного обеспечения [Электронный ресурс]. – Режим доступа: http://www.km.ru/referats/1AE805F6BA394FBB8479B41F861E1085

20. Yu, X, Bui, T.D. Robust Estimation for Range Image Segmentation and Reconstruction, IEEE trans. Pattern Analysis and Machine Intelligence, 2013. – p. 530–538.

21. Разработка тест-планов, программ и методик испытаний [Электронный ресурс]. – Режим доступа: http://www.artwell.ru/services/razrabotka-test-planov-programm-i-metodik-ispytaniy

22. Steinhaus S. Comparison of mathematical programs for data analysis. – Germany: University of Frankfurt, 2013. – p. 48.

23. Ramadevi Y. Segmentation and object recognition using edge detection techniques, International Journal of Computer Science and Information Technology, Vol 2, No.6., 2012. – p. 153–161.

24. Punam Thakare. A Study of Image Segmentation and Edge Detection Techniques, International Journal on Computer Science and Engineering, Vol 3, No.2., 2014. – p. 899–904.

25. Выделение границ [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Выделение_границ

26. Виды томографии [Электронный ресурс]. – Режим доступа: http://mrt-diagnostics.ru/kt/vidy-tomografii/

27. Сравнительный анализ фильтрации медицинских изображений [Электронный ресурс]. – Режим доступа: http://masters.donntu.org/2014/fknt/kondratov/library/theme1.htm

28. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. — Москва: Техносфера, 2012. — С. 148–414.

29. Сойфер В. Методы компьютерной обработки изображений / В.А. Сойфер. — М.: ФИЗМАЛИТ. — 2014. — С. 684–693.

30. Калинкина Д. Проблема подавления шума на изображениях и видео, и различные подходы к ее решению / Д. Калинкина, Д. Ватолин — Москва: Техносфера, 2013. — С. 118–128.

31. Алиев М.В. Выделение контуров на малоконтрастных и размытых изображениях с помощью фрактальной фильтрации / М.В. Алиев, А.Х. Панеш, М.С. Каспарьян // Вестник Адыгейского государственного университета. Серия 4: Естественно-математические и технические науки. 2013. №3. — С. 101–107.

32. Беленский Й. Метод выделения контура на слабоконтрастных размытых изображениях / Й.Й. Беленский, И.В. Микулка // Вестник Винницкого политехнического института. — 2014 г. — № 3. — С. 6–7.

33. Садыхов Р. Обработка изображений и идентификация объектов в системах технического зрения / Р.Х. Садыхов, А.А. Дудкин // Объединенный институт проблем информатики НАН Беларуси, Минск, Беларусь. — 2012 г. — № 3. — С. 10–11.

Приложение А. Код функции Edge_detection

```
function varargout = Edge_detection(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',     gui_Singleton, ...
                   'gui_OpeningFcn',    @Edge_detection_OpeningFcn, ...
                   'gui_OutputFcn',     @Edge_detection_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function Edge_detection_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
warning('off','MATLAB:HandleGraphics:ObsoletedProperty:JavaFrame');
jframe=get(handles.LTrack,'javaframe');
jIcon=javax.swing.ImageIcon('dental-icon.gif');
jframe.setFigureIcon(jIcon);
function varargout = Edge_detection_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function Home_Callback(hObject, eventdata, handles)
LTproject = guidata(gcbo);
function browse_Callback(hObject, eventdata, handles)
```

```
LTproject = guidata(gcbo);
[basefilename,path]= uigetfile({'*.tif;*.bmp;*.jpg'},'Open JPG, BMP, TIF Image
File');
filename= fullfile(path, basefilename);
I = imread (filename);
size(I)
MapQuantization = I;
set(LTproject.LTrack,'CurrentAxes',LTproject.MapQuantization);
set(imshow(MapQuantization));
set(LTproject.LTrack,'Userdata',filename);
set(LTproject.MapQuantization,'Userdata',I);
set(LTproject.filebrowse,'String',strcat('File Location : ',path,basefilename));
function proses_Callback(hObject, eventdata, handles)
LTproject = guidata(gcbo);
ImageInput = get(LTproject.MapQuantization,'Userdata');
disp(size(ImageInput));
[GC] = sobel(ImageInput);
GreenChan=GC;
axes (handles.HistGreenChan);
imhist(GreenChan);
function pushbutton4_Callback(hObject, eventdata, handles)
function pushbutton5_Callback(hObject, eventdata, handles)
function pushbutton6_Callback(hObject, eventdata, handles)
function pushbutton8_Callback(hObject, eventdata, handles)
function pushbutton9_Callback(hObject, eventdata, handles)
function pushbutton10_Callback(hObject, eventdata, handles)
function pushbutton11_Callback(hObject, eventdata, handles)
function pushbutton12_Callback(hObject, eventdata, handles)
function gapi_Callback(hObject, eventdata, handles)
function pushbutton14_Callback(hObject, eventdata, handles)
```

```
function linestrength_Callback(hObject, eventdata, handles)
function linetracking_Callback(hObject, eventdata, handles)
function pushbutton17_Callback(hObject, eventdata, handles)
function about_Callback(hObject, eventdata, handles)
function reset_Callback(hObject, eventdata, handles)
LTproject = guidata(gcbo);
set(gcf,'WindowStyle','Normal')
frames = java.awt.Frame.getFrames();
frames(end).setAlwaysOnTop(0);
ReBut = questdlg('Are you really want to reset this
application?','Reset','Yes','No','default');
switch ReBut
case {'No'}
case 'Yes'
closeGUI = LTproject.LTrack; %handles.LTproject is the GUI figure
guiPosition = get(LTproject.LTrack,'Position'); %get the position of the GUI
guiName = get(LTproject.LTrack,'Name'); %get the name of the GUI
close(closeGUI); %close the old GUI
eval(guiName) %call the GUI again
end
function close_Callback(hObject, eventdata, handles)
LTproject = guidata(gcbo);
set(gcf,'WindowStyle','Normal')
frames = java.awt.Frame.getFrames();
frames(end).setAlwaysOnTop(0);
pos_size = get(LTproject.LTrack,'Position');
button = questdlg('Are you really want to close this
application?','Close','Yes','No','default');
warning('off','MATLAB:HandleGraphics:ObsoletedProperty:JavaFrame');
jframe=get(handles.LTrack,'javaframe');
```

```
jIcon=javax.swing.ImageIcon('dental-icon.gif');
jframe.setFigureIcon(jIcon);
switch button
case {'No'}
case 'Yes'
close;
end
function login_Callback(hObject, eventdata, handles)
function logout_Callback(hObject, eventdata, handles)
LTproject = guidata(gcbo);
set(gcf,'WindowStyle','Normal')
frames = java.awt.Frame.getFrames();
frames(end).setAlwaysOnTop(0);
pos_size = get(LTproject.LTrack,'Position');
button = questdlg('Are you really want to close this
application?','close','Yes','No','default');
switch button
case {'No'}
case 'Yes'
close;
end
function save_Callback(hObject, eventdata, handles)
function pushbutton28_Callback(hObject, eventdata, handles)
function pushbutton29_Callback(hObject, eventdata, handles)
function pushbutton30_Callback(hObject, eventdata, handles)
function pushbutton31_Callback(hObject, eventdata, handles)
function pushbutton32_Callback(hObject, eventdata, handles)
function pushbutton33_Callback(hObject, eventdata, handles)
function pushbutton34_Callback(hObject, eventdata, handles)
function pushbutton35_Callback(hObject, eventdata, handles)
```

```
function pushbutton36_Callback(hObject, eventdata, handles)
function pushbutton37_Callback(hObject, eventdata, handles)
function pushbutton38_Callback(hObject, eventdata, handles)
function pushbutton39_Callback(hObject, eventdata, handles)
function pushbutton40_Callback(hObject, eventdata, handles)
function pushbutton42_Callback(hObject, eventdata, handles)
function pushbutton43_Callback(hObject, eventdata, handles)
function LTrack_DeleteFcn(hObject, eventdata, handles)
function filebrowse_Callback(hObject, eventdata, handles)
function pushbutton45_Callback(hObject, eventdata, handles)
function listbox1_Callback(hObject, eventdata, handles)
function listbox1_CreateFcn(hObject, eventdata, handles)
function pushbutton46_Callback(hObject, eventdata, handles)
function pushbutton47_Callback(hObject, eventdata, handles)
function pushbutton48_Callback(hObject, eventdata, handles)
function pushbutton49_Callback(hObject, eventdata, handles)
function pushbutton50_Callback(hObject, eventdata, handles)
function pushbutton51_Callback(hObject, eventdata, handles)
function pushbutton52_Callback(hObject, eventdata, handles)
function reset_DeleteFcn(hObject, eventdata, handles)
function pushbutton55_Callback(hObject, eventdata, handles)
function pushbutton57_Callback(hObject, eventdata, handles)
LTproject = guidata(gcbo);
ImageInput = get(LTproject.MapQuantization,'Userdata');
disp(size(ImageInput));
[GC] = canny(ImageInput);
GreenChan=GC;
axes (handles.HistGreenChan);
imhist(GreenChan);
```

Приложение Б. Код функции canny

```
function [GreenChannel] = canny(LS)
tic
% Set with dynamic of W
InitScale=3;
Wstep=2;
FinScale=11;
% Set for waitbar
Counter=ceil(FinScale/InitScale)+1;
CounterLoop=0;
hWB = waitbar(0,'Please wait...');
for W=InitScale:Wstep:FinScale
img = rgb2gray(LS);
GreenChannel=img;
img = double (img);
img = medfilt2(img,[3 20]); % [3 3], [3 5]
%Value for Thresholding
T_Low = 0.05275; %0.06675 %0.5575
T_High = 0.02;
%Gaussian Filter Coefficient
B = [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 ];
B = 1/159.* B;
%Convolution of image by Gaussian Coefficient
A=conv2(img, B, 'same');
%Filter for horizontal and vertical direction
KGx = [-1, 0, 1; -2, 0, 2; -1, 0, 1];
KGy = [1, 2, 1; 0, 0, 0; -1, -2, -1];
%Convolution by image by horizontal and vertical filter
Filtered_X = conv2(A, KGx, 'same');
```

```
Filtered_Y = conv2(A, KGy, 'same');
%Calculate directions/orientations
arah = atan2 (Filtered_Y, Filtered_X);
arah = arah*180/pi;
pan=size(A,1);
leb=size(A,2);
%Adjustment for negative directions, making all directions positive
for i=1:pan
        for j=1:leb
                if (arah(i,j)<0)
                arah(i,j)=360+arah(i,j);
                end;
        end;
end;
arah2=zeros(pan, leb);
%Adjusting directions to nearest 0, 45, 90, or 135 degree
for i = 1  : pan
        for j = 1 : leb
                if ((arah(i, j) >= 0 ) && (arah(i, j) < 22.5) || (arah(i, j) >= 157.5) &&
        (arah(i, j) < 202.5) || (arah(i, j) >= 337.5) && (arah(i, j) <= 360))
                arah2(i, j) = 0;
        elseif ((arah(i, j) >= 22.5) && (arah(i, j) < 67.5) || (arah(i, j) >= 202.5) &&
(arah(i, j) < 247.5))
                arah2(i, j) = 45;
        elseif ((arah(i, j) >= 67.5 && arah(i, j) < 112.5) || (arah(i, j) >= 247.5 &&
arah(i, j) < 292.5))
                arah2(i, j) = 90;
        elseif ((arah(i, j) >= 112.5 && arah(i, j) < 157.5) || (arah(i, j) >= 292.5 &&
arah(i, j) < 337.5))
                arah2(i, j) = 135;
```

```
            end;
        end;
end;
sprintf('W = %d%.3f%%',W)
CounterLoop=CounterLoop+1;
waitbar(CounterLoop/Counter);
end
close(hWB);
% delete(hWB);
figure, imagesc(arah2); colorbar;
%Calculate magnitude
magnitude = (Filtered_X.^2) + (Filtered_Y.^2);
magnitude2 = sqrt(magnitude);
BW = zeros (pan, leb);
%Non-Maximum Supression
for i=2:pan-1
        for j=2:leb-1
            if (arah2(i,j)==0)
            BW(i,j) = (magnitude2(i,j) == max([magnitude2(i,j), magnitude2(i,j+1),
        magnitude2(i,j-1)]));
            elseif (arah2(i,j)==45)
            BW(i,j) = (magnitude2(i,j) == max([magnitude2(i,j), magnitude2(i+1,j-
        1), magnitude2(i-1,j+1)]));
            elseif (arah2(i,j)==90)
            BW(i,j) = (magnitude2(i,j) == max([magnitude2(i,j), magnitude2(i+1,j),
        magnitude2(i-1,j)]));
            elseif (arah2(i,j)==135)
            BW(i,j) = (magnitude2(i,j) == max([magnitude2(i,j),
        magnitude2(i+1,j+1), magnitude2(i-1,j-1)]));
            end;
```

```
        end;
end;
BW = BW.*magnitude2;
figure, imshow(BW);
%Hysteresis Thresholding
T_Low = T_Low * max(max(BW));
T_High = T_High * max(max(BW));
T_res = zeros (pan, leb);
for i = 1  : pan
        for j = 1 : leb
                if (BW(i, j) < T_Low)
                T_res(i, j) = 0;
                elseif (BW(i, j) > T_High)
                T_res(i, j) = 1;
                %Using 8-connected components
                elseif ( BW(i+1,j)>T_High || BW(i-1,j)>T_High || BW(i,j+1)>T_High ||
        BW(i,j-1)>T_High || BW(i-1, j-1)>T_High || BW(i-1, j+1)>T_High || BW(i+1,
        j+1)>T_High || BW(i+1, j-1)>T_High)
                T_res(i,j) = 1;
                end;
        end;
end;
edge_final = uint8(T_res.*255);
%Show final edge detection result
figure, imshow(edge_final);
se90=strel('line', 3, 90);
se0=strel('line', 3, 0);
BWsdil=imdilate(edge_final, [se90 se0]);
figure, imshow(BWsdil), title('dilated gradient mask');
```