Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА ТЕОРЕТИЧЕСКАЯ ИНФОРМАТИКА И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Методы выделения границ на рентгенограммах

Студент ИУ9-51б _____ М. Ибрагимов
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы _____ А. В. Брагин
(Подпись, дата) (И.О.Фамилия)

Консультант _____ А. В. Брагин
(Подпись, дата) (И.О.Фамилия)

Москва

2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Цель курсовой работы – разработка алгоритмов для выделения контуров на медицинских изображениях (рентгенограммах).

В настоящее время все большее применение находит машинная графика. Компьютерное зрение применяется во многих сферах жизни: камеры видеонаблюдения, автономные транспортные средства, анализ документов и т.д. Так компьютерное зрение не обошло стороной и медицину.

Задача анализа медицинских изображений на данный момент очень популярна. “Правильная” обработка изображений способна очень сильно облегчить жизнь специалистам, сократив время постановления диагноза и уменьшив вероятность ошибки при постановлении диагноза.

Базовая задача — это выделение контуров на изображениях. Во-первых, это удаляет всё “ненужное” со снимков, и специалисту легче рассматривать такие изображения. Во-вторых, обработанные такими методами изображения могут использоваться в задачах машинного обучения для автоматического постановления диагноза.

В данной работе тестирование реализованных алгоритмов будет применяться на рентгенограммах костей, а именно костей кисти.

Снимки были взяты из датасета для соревнований по глубокому обучению от Стэнфорда – Bone X-Ray Deep Learning Competition.

1. ПОСТАНОВКА ЗАДАЧИ

Цель данной работы – выделение контуров на рентгенограммах.

Для достижения данной цели необходимо решить следующие задачи:

- Изучить существующие методы, для данной задачи.
- Выбрать список технологий, с помощью которых лучше реализовывать данные алгоритмы
- Тестировать полученные результаты.

Результатами данной работы являются снимки с выделенными контурами.

В заключении будет произведено сравнение методов на основе скорости их работы и точности выделения границ.

2. ОПИСАНИЕ ТЕХНОЛОГИЙ ДЛЯ НАПИСАНИЯ ПРОГРАММЫ

Алгоритм – это конечная совокупность точно заданных правил решения некоторого класса задач или набор инструкций, описывающих порядок действий исполнителя для решения определённой задачи.

Для реализации алгоритмов выделения контуров необходимо выбрать:

- Высокоуровневый язык программирования
- Библиотеки для работы с изображениями

2.1 Язык программирования Python

Язык программирования Python версии 3.8 был выбран как язык для реализации данной задачи.

Python имеет простой синтаксис, на нем реализовано огромное количество библиотек, для работы с изображениями.

По скорости работы, реализации на Python будут выполняться примерно столько же, сколько на C/C++, т. к. основная сложность — это вычисления, а они в свою очередь выполняются с помощью библиотеки NumPy, которая написана на C.

2.2 Библиотека PIL

Библиотека PIL (Python Image Library) – библиотека языка Python, предназначенная для работы с растровой графикой.

Возможности библиотеки:

- Поддержка бинарных, полутоновых, индексированных, полноцветных изображений.
- Поддержка большого набора форматов изображений (BMP, EPS, GIF, JPEG, PDF, PNG, PNM…) на чтение и запись
- Поддержка форматов (MPEG, ICO, PSD, PCX, WMF) только для чтения
- Преобразование изображений из одного формата в другой
- Правка изображений (использование фильтров, рисование, взятие значения пикселя)

Изображение в PIL представляет собой объект (object).

В данной работе PIL нужен для чтения и записи к значениям пикселей. Обычно в PIL работают с RGB-значениями, но в данной работе будет обработка изображения в оттенках серого, поэтому у пикселя будет одно значение.

Для чтения значения пикселя применяется метод `getpixel()`. В качестве аргументов в данный метод передается кортеж (x, y) – координаты.

```
Image.getpixel((100, 100)) #255
```

Также, придется и изменять значения пикселей, следовательно нужен метод для записи. В PIL за запись в пиксель отвечает метод `putpixel()`. В качестве аргументов в данный метод передается кортеж (x, y) – координаты и значение пикселя.

```
Image.putpixel((100, 100), 255)
```

2.3 Библиотека OpenCV

OpenCV - библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом.

Имеет реализации во множестве языков. Может свободно использоваться в академических целях.

В OpenCV доступен огромный набор возможностей, для работы с изображениями. В ней уже реализованы операторы для выделения границ, но т. к. это было целью данной работы, поэтому операторы будут реализованы с нуля, а OpenCV будет использоваться как библиотека для предварительной обработки изображений, и для удобного доступа к пикселям.

Основные модули библиотеки:

- Cxcore – ядро, содержит реализации алгоритмов для работы с матрицами, базовые функции 2D графики и т.д.
- CV – модуль обработки изображений и компьютерного зрения: базовые операции над изображениями (фильтры, геометрические преобразования); анализ изображений (поиск контуров); анализ движения, слежения за объектами; обнаружение объектов (лиц)
- HighGUI – модуль для ввода/вывода изображений и видео, создание пользовательского интерфейса
- Cvaux – экспериментальные функции (нахождение черт лица)
- CvCam – захват видео.

В данной работе OpenCV будет использована для предварительной обработки изображений, а именно: для применения фильтра Гаусса (сглаживание методом Гаусса). Описание и применение предварительной обработки будет описано далее.

2.4 Библиотека Matplotlib

Библиотека Matplotlib - библиотека на языке программирования Python для визуализации данных двумерной графикой (3D графика также поддерживается). Получаемые изображения могут быть использованы в качестве иллюстраций в публикациях.

Matplotlib является гибким, легко конфигурируемым пакетом, который вместе с NumPy, SciPy и IPython предоставляет возможности, подобные MATLAB.

Пакет поддерживает многие виды графиков и диаграмм:

- Графики
- Диаграммы разброса
- Столбчатые диаграммы
- Круговые диаграммы
- Контурные графики
- Поля градиентов

С помощью Matplotlib можно работать с изображениями, например подмодуль Matplotlib – Pyplot используется в данной курсовой работе для красивого вывода изображений.

Набор поддерживаемых форматов:

- JPEG
- PDF
- PNG
- SVG
- TIFF
- RGBA (сырой формат)

3. ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ

3.1 Основные понятия

Рентгенограмма - зарегистрированное на светочувствительном материале (фотоплёнке, фотопластинке) изображение объекта, возникающее в результате взаимодействия рентгеновских лучей с веществом.

Рентгенография применяется для диагностики: Рентгенологическое исследование органов позволяет уточнить форму данных органов, их положение, тонус, перистальтику, состояние рельефа слизистой оболочки.

Сегментация изображения – процесс разделения цифрового изображения на несколько сегментов.

Контраст изображения, диапазон яркостей изображения — отношение яркостей самой светлой и самой тёмной частей изображения.

Градиент — характеристика, показывающая направление наискорейшего возрастания некоторой величины, значение которой меняется от одной точки пространства к другой.

RGB формат изображения – аддитивная цветовая модель, описывающая способ кодирования цвета для цветовоспроизведения с помощью трёх цветов (R – red (красный), G – green (зеленый), B – blue (синий)), которые принято называть основными.

Оттенки серого - цветовой (одноканальный) режим изображений, которые отображаются в оттенках серого, размещённые в виде таблицы в качестве эталонов яркости белого цвета.

Свёртка – операция над парой матриц, результатом которой является так же матрица. В операции перемножаются соответствующие элементы, затем сумма произведений записывается в центральный элемент матрицы (применимо к матрицам с нечетным порядком).

3.2 Приведение RGB изображения в оттенки серого

Большинство изображений изначально представлено RGB формате. Для выделения контуров важно только значение контраста. Поэтому логично перевести изображение в оттенки серого.

Изначально, загрузив изображение, с помощью библиотеки PIL или же с помощью OpenCV, в режиме по умолчанию и, прочитав любой пиксель, нам вернется кортеж из 3 значений (R, G, B) ($0 \leq R, G, B \leq 255$).

Наиболее простой способ перевести изображение в одноканальный режим – это взять среднее арифметическое из 3 значений (формула (1)).

$$X = (R + G + B) / 3 \quad (1)$$

где R – значение красного канала; G – значение зеленого канала; B – значение синего; X – результат в одноканальном формате (яркость пикселя $0 \leq X \leq 255$)

Не стоит забывать о том, что наши глаза воспринимают яркости разных цветов по-разному. Например, для человеческого глаза зеленый цвет в 10 раз светлее, чем синий. Для того чтобы отношения яркостей при конвертации сохранялись, были подобраны коэффициенты, для вычисления одноканальных значений. (формула (2)).

$$X = 0.2126 * R + 0.7152 * G + 0.0722 * B \quad (2)$$

где R, G, B, X – это те же переменные, что и в формуле (1).

Данные коэффициенты не рекомендуется использовать, если яркости изображения высокие. Поэтому для очень ярких изображений используют гамма-компрессию. Гамма-компрессия, простыми словами, это процесс затемнение изображения. Вычисление с гамма-компрессией показано в формуле (3).

$$X = 0.299 * R + 0.587 * G + 0.114 * B \quad (3)$$

где R, G, B, X – это те же переменные, что и в формуле (1).

Есть несколько возможностей конвертировать изображение из RGB в оттенки серого с помощью Python.

1) Прямое чтение R, G, B значений из пикселя и запись в отдельно созданную новую матрицу (двумерный массив) вычисленных значений.
2) Инкапсуляцией, т.е. конвертацией с помощью методов, реализованных в библиотеках.

Например, конвертировать в оттенки серого из RGB с помощью библиотеки OpenCV или PIL можно прямо при загрузке изображения.

Ниже приведён пример загрузки изображения в оттенках серого с помощью OpenCV:

```
cv2.imread('./images/brush.png', 0)
```

Ниже приведён пример загрузки изображения в оттенках серого с помощью PIL:

```
Image.open('./images/brush.png').convert('L')
```

Конвертация в оттенки серого из RGB в PIL и OpenCV происходит без гамма-компрессии, т.е. по формуле (2).

4. АНАЛИЗ ПРЕДОБРАБОТАННЫХ ИЗОБРАЖЕНИЙ

Когда изображения переведены в оттенки серого, можно приступать к основной задаче – к задаче выделения контуров.

Процесс обнаружения точных разрывов яркости на изображении называется процессом выделения границ. Разрывы — это резкие изменения в группе пикселей, которые являются границами объектов. Классический алгоритм обнаружения границ задействует свертку изображения с помощью оператора, который основывается на чувствительности к большим перепадам яркости на изображении, а при проходе однородных участков возвращает нуль.

Вычисление свертки – это, по факту, вычисления градиента (величина перепада в дальнейшем). Один из исторически первых методов – это перекрёстный оператор Робертса. Позже Прюиттом был предложен оператор на основе понятия центральной разницы. Основным недостатком является чувствительность таких методов к шуму. Наиболее известным из дифференциальных методов выделения контуров является оператор Собеля.

4.1 Методы и реализации выделения контуров

На сегодняшний день реализовано большое количество операторов выделения границ. Исследования в этой области ведутся с середины XX века.

Ниже будут приведены описания некоторых операторов.

4.1.1 Перекрёстный оператор Робертса

Область 3x3, показанная на рисунке ниже, представляет собой значения яркости в окрестности некоторого элемента изображения.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Рис. 3 – окрестность данного изображения

Оператор выделения границ Робертса введен Лоуренсом Робертсом в 1964 году. Основан он на перепадах яркости.

По сути, перекрёстный оператор Робертса вычисляет на плоском дискретном изображении сумму квадратов разниц между диагонально смежными пикселями.

Это может быть выполнено сверткой изображения с двумя ядрами:

-1	0
0	1

0	-1
1	0

Рис. 4 Маски оператора Робертса

Градиенты вычисляются по формулам ниже:

$$Gx = (z_9 - z_5) \quad (4)$$

где Gx — это величина перепада яркости по абсциссе

$$Gy = (z_8 - z_6) \quad (5)$$

где Gy — это величина перепада яркости по ординате

$$G = \sqrt{Gx^2 + Gy^2} \quad (6)$$

где G – результирующая величина перепада.

Реализация масок размерами 2x2 не очень удобна, т.к. у них нет четко выраженного центрального элемента, что существенно отражается на результате выполнения фильтрации. Но этот “минус” порождает очень полезное свойство данного алгоритма – высокую скорость обработки изображения.

4.1.2 Оператор Прюитт

Оператор Прюитт - метод выделения границ в обработке изображений, который вычисляет максимальный отклик на множестве ядер свертки для нахождения локальной ориентации границы в каждом пикселе. Он был создан доктором Джудит Прюитт для обнаружения границ медицинских изображений в 1970 году.

Метод Прюитта оперирует с окрестностью (см. рис. 3), по факту, так же, как и оператор Робертса, но отличие в том, что в методе Прюитта для свертки используются 2 маски размером 3x3.

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Рис. 5 Маски оператора Прюитта

Градиенты вычисляются по формулам ниже:

$$Gx = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \quad (7)$$

где Gx — это величина перепада яркости по абсциссе

$$Gy = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \quad (8)$$

где Gy — это величина перепада яркости по ординате

$$G = \sqrt{Gx^2 + Gy^2} \quad (9)$$

где G – результирующая величина перепада.

4.1.3 Оператор Собеля

Оператор Собеля введен Собелем в 1970 году. Данный оператор очень схож с оператором Прюитта, различия только в коэффициентах матрицы.

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Рис. 6 Маски оператора Собеля

Градиенты вычисляются по формулам ниже:

$$Gx = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \quad (10)$$

где Gx — это величина перепада яркости по абсциссе

$$Gy = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \quad (11)$$

где Gy — это величина перепада яркости по ординате

$$G = \sqrt{Gx^2 + Gy^2} \quad (12)$$

где G – результирующая величина перепада.

$$theta = \tan^{-1}(Gy / Gx) \quad (13)$$

где theta – направление градиента.

4.1.3 Оператор Шарра

Оператор Собеля сглаживает паразитные эффекты на изображении, вызываемые чисто центрально-дифференциальным оператором, но не обладает полной вращательной симметрией. Шарр исследовал улучшение этого свойства и пришёл к выводу, что лучшие результаты дают следующие выражения.

$$Gx = (3z_1 + 10z_2 + 3z_3) - (3z_7 + 10z_8 + 3z_9) \quad (14)$$

где Gx — это величина перепада яркости по абсциссе

$$Gy = (3z_1 + 10z_4 + 3z_7) - (3z_3 + 10z_6 + 3z_9) \quad (15)$$

где Gy — это величина перепада яркости по ординате

4.1.4 Оператор Лапласа

Оператор Лапласа для выделения границ работает аналогичным методом (как предыдущие), но использует другую маску для свертки.

0	1	0
1	-4	1
0	1	0

Рис. 7 Маска оператора Лапласа

Формула для вычислений здесь только одна :

$$G = |z_2 + z_4 + z_6 + z_8 - z_5| \quad (16)$$

где G — это величина перепада яркости

4.1.4 Оператор Кирша

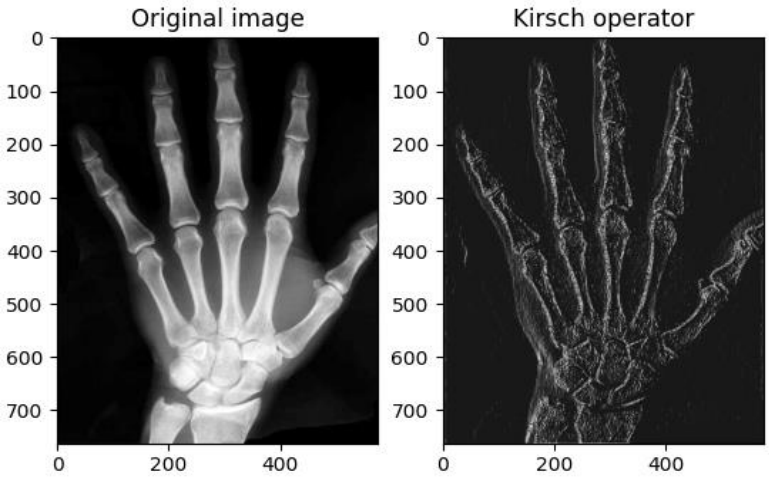
Обнаружение границ этим методом было введено Киршем в 1971 году. Алгоритм основан на использовании всего одной маски, которую вращают по восьми главным направлениям: север, северо-запад, запад, юго-запад, юг, юго-восток, восток и северо-восток. Маски имеют следующий вид:

$$K1 = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} \quad K2 = \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix} \quad K3 = \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad K4 = \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

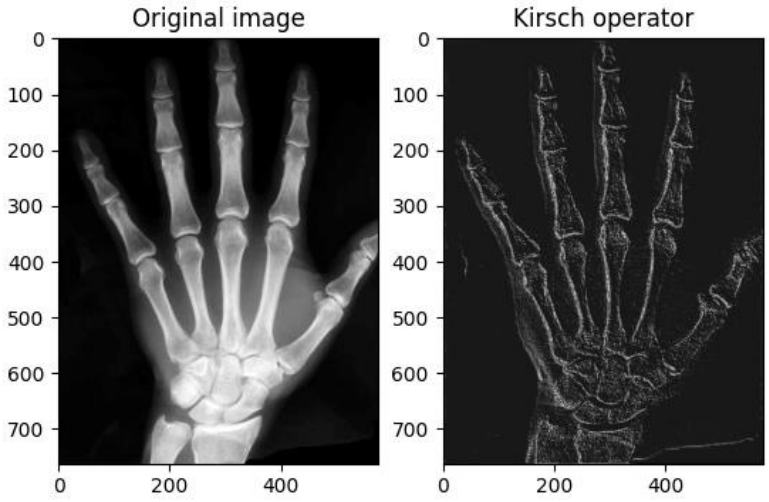
$$K5 = \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} \quad K6 = \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix} \quad K7 = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \quad K8 = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$$

Рис.8 Маски оператора Кирша

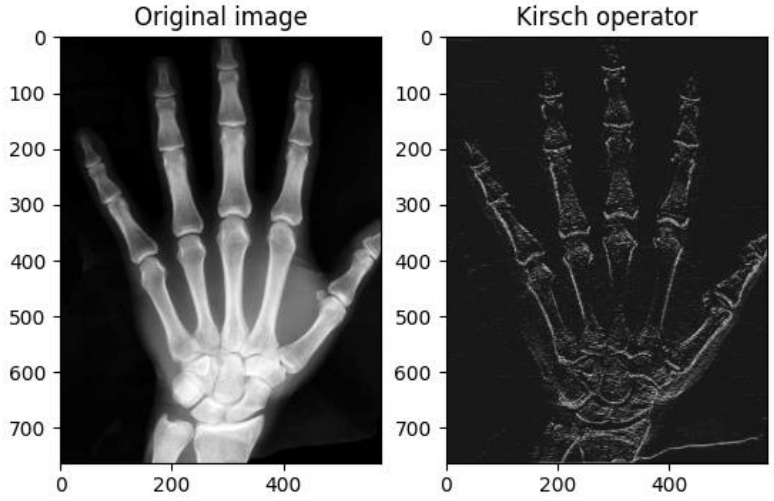
Ниже показаны несколько результатов вычисления по первым трем маскам.



а)



б)



в)

Рис. 9: а) – К1, б) – К2, в) – К3

Можно заметить, что на результирующих изображениях много шума, контуры выделены плохо. Это проблема решается выбором максимальной свертки на каждой итерации.

4.1.4 Оператор Робинсона

Метод Робинсона, введенное в 1977, подобен методу Кирша, но является более простым в реализации в силу использования коэффициентов 0, 1 и 2. Маски данного оператора симметричны относительно центральной оси, заполненной нулями. Достаточно получить результат от обработки первых четырех масок, остальные же можно получить, инвертируя первые.

-1	0	1
-2	0	2
-1	0	1

r_0

0	1	2
-1	0	1
-2	-1	0

r_1

1	2	1
0	0	0
-1	-2	-1

r_2

2	1	0
1	0	-1
0	-1	-2

r_3

1	0	-1
2	0	-2
1	0	-1

r_4

0	-1	-2
1	0	-1
2	1	0

r_5

1	2	1
0	0	0
-1	-2	-1

r_6

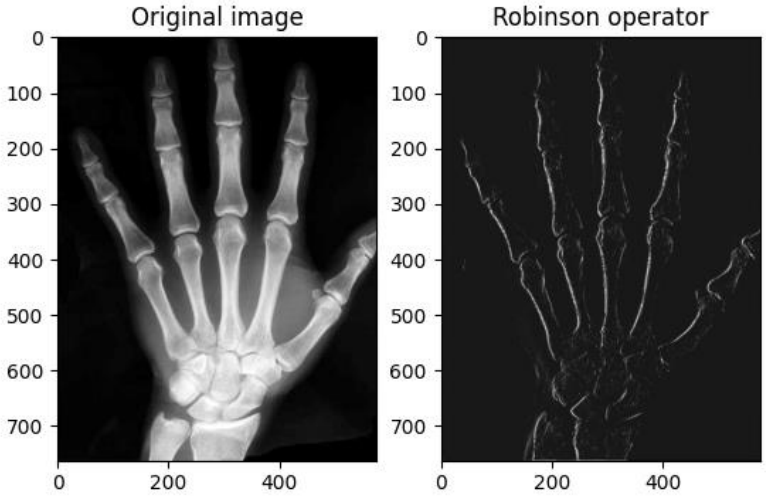
-2	-1	0
-1	0	1
0	1	2

r_7

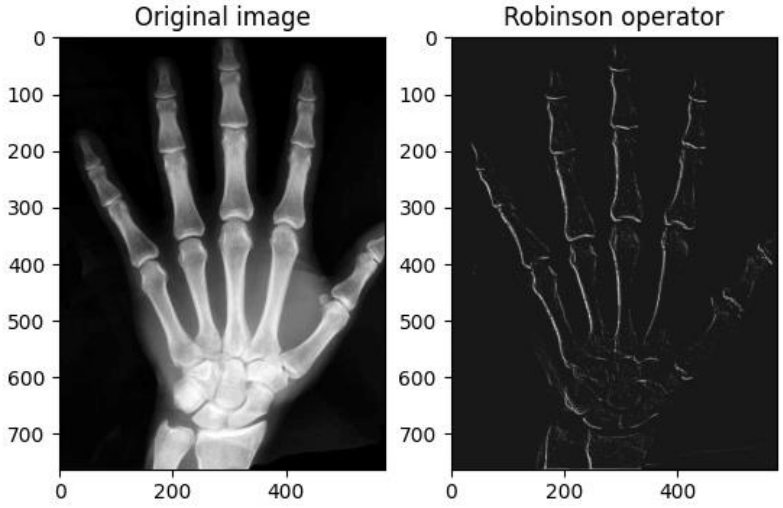
Рис. 10 – Маски оператора Робинсона

Вычисления выполняются аналогично предыдущим методам.

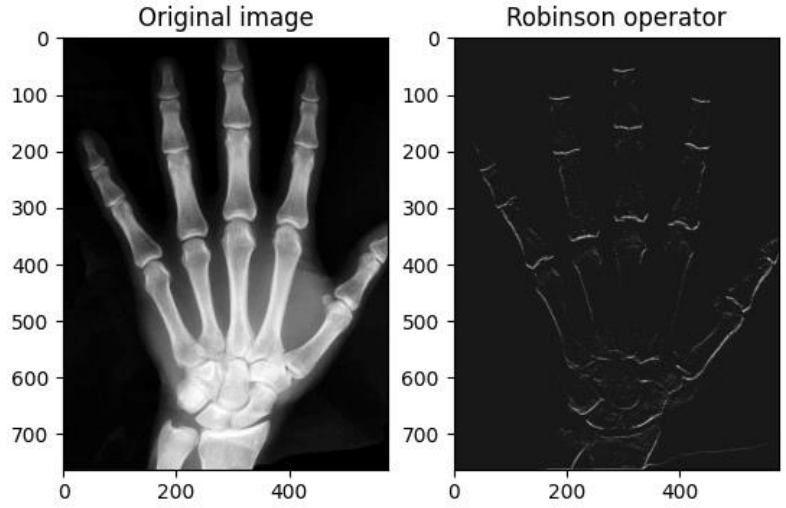
Ниже показаны несколько результатов вычисления по первым трем маскам.



а)



б)



в)

Рис. 10: а) – r_0, б) – r_1, в) – r_2

В результате, так же как и в операторе Кирша, мы получаем плохо выделенные контуры. Это решается так же – выбором максимальной свертки.

4.1.4 Оператор Кэнни

Детектор границ Кэнни является одной из самых популярных алгоритмов обнаружения контуров. Впервые он был предложен Джоном Кэнни в магистерской диссертации в 1983 году, и до сих пор является лучше многих алгоритмов, разработанных позднее.

Алгоритм состоит из 5 этапов:

1) Сглаживание (размытие – удаление шумов)
2) Поиск градиентов
3) Подавление не-максимумов - только локальные максимумы отмечаются как границы
4) Двойная пороговая фильтрация - потенциальные границы определяются порогами
5) Трассировка области неоднозначности. Все “сильные границы” отмечаются белым цветом, а все слабые границы удаляются (отмечаются черным цветом).

Сглаживание. Сглаживание, по сути, является одним из этапов предобработки изображения. В последнее время на медицинских изображениях все меньше шумов, но из-за специфической работы оператора Кэнни, шум проявляется в результирующем изображении в виде границ, поэтому Джоном Кэнни было решено включить в алгоритм оператора удаление шумов.

Для удаления шумов в операторе Кэнни используется размытие Гаусса.

Вычисление значения “размытых” пикселей, применяется по формуле

$$G(x,y,\sigma) = \frac{1}{2\cdot\pi\cdot\sigma^2}\cdot e^{-\frac{(x2+y2)}{2\cdot\sigma2}} \quad (17)$$

где σ – входной параметр, влияющий на размытие; x и y координаты.

Но на практике удобнее применять свертку. К окрестностям изображения (рис. 3)

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Рис. 11 Маска фильтра Гаусса

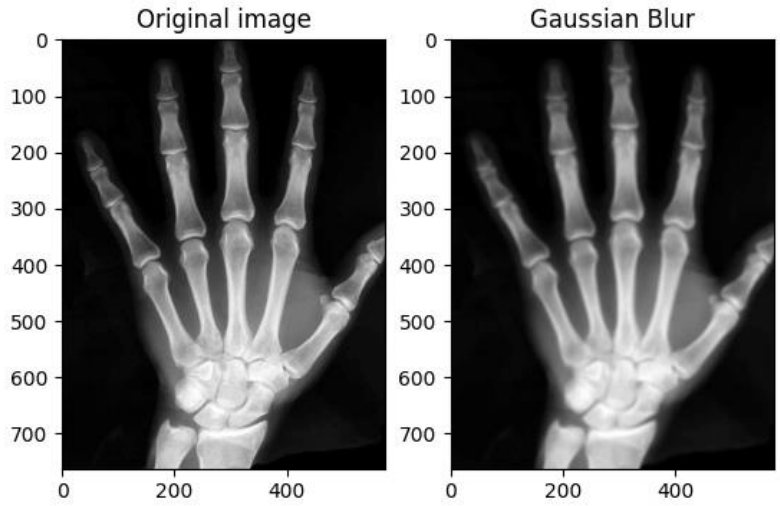


Рис.12 Применение размытия Гаусса

Поиск градиентов. Поиск градиентов является главным шагом в алгоритме оператора Кэнни. На данном шаге можно взять, например, оператор Прюитта или Собеля.

Оператор Кэнни в оригинале, применяет оператор Собеля. Именно в операторе Кэнни, важно направления градиента, т. к. для следующего этапа будет оно будет использоваться.

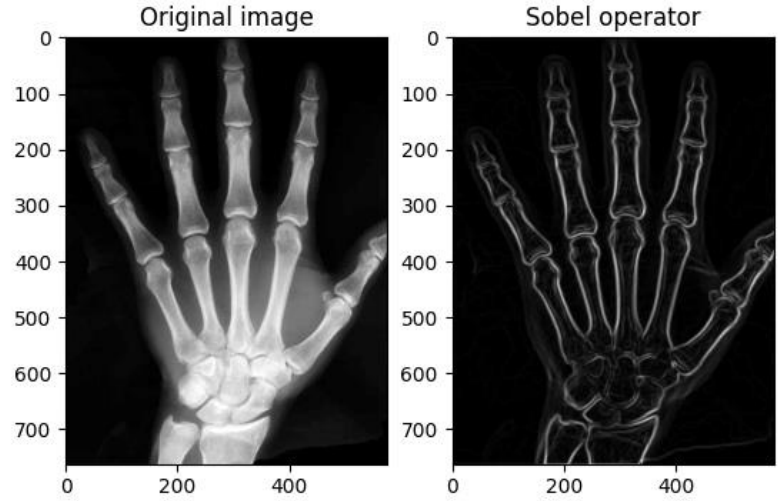


Рис. 13 Применение оператора Собеля

Подавление не-максимумов. Локальный максимум (пиксель) – это такой пиксель, значение которого больше соседних.

Направление градиента подсказывает нам, с какими пикселями сравнивать главный текущий пиксель.

Выделяют 8 направлений градиента.

1) $0 > theta > \pi/4$
2) $\pi/4 > theta > \pi/2$
3) $\pi/2 > theta > 3\pi/4$
4) $3\pi/4 > theta > \pi$
5) $\pi > theta > -3\pi/4$
6) $-3\pi/4 > theta > -\pi/2$
7) $-\pi/2 > theta > -\pi/4$
8) $-\pi/4 > theta > 0$

Возьмем любую окрестность исходного изображения (рис. 3). Мы находимся в центре матрицы (элемент z5).

Если значение направление градиента 1), то мы сравниваем z5 с z4 и с z6. Если, например, z5 меньше хотя бы одного соседа, то мы присваиваем z5 значение 0.

Далее, если направление градиента 2), то мы сравниваем z5 с z3 и с z7.

Если направление 3), сравниваем z5 с z2 и с z8.

Если направление 4), сравниваем z5 с z1 и с z9.

5), 6), 7), 8) направления – по факту выполняем то же, что и в 1), 2), 3), 4) – соответственно.

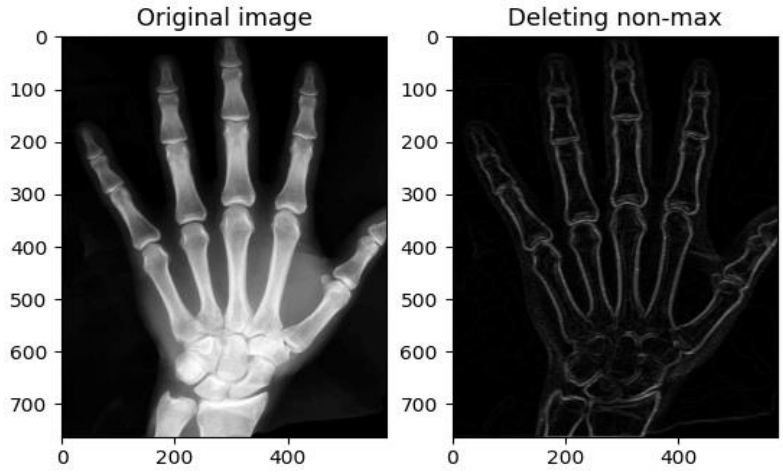


Рис. 14 Удаление локальных не-максимумов

Двойная пороговая фильтрация и трассировка области неоднозначности очищают конечный результат от шумов, и слабых контуров.

Эти два этапа оператора Кэнни, можно реализовать совместно. Оператор принимает на вход два параметра, например, p и q ($0 \leq p, q \leq 255$). Все пиксели, имеющие значения < p объявляются слабыми (значит, что граница в этом месте слабая). Все пиксели > q объявляются сильными.

Затем всем сильным пикселям мы придаем значение 255 (белый цвет), а всем слабым придаем значение 0 (черный). Существуют реализации, где пиксели, значения которых находятся в диапазоне между p и q, никак не изменяют, либо придают какое-либо значение, введенное вручную.

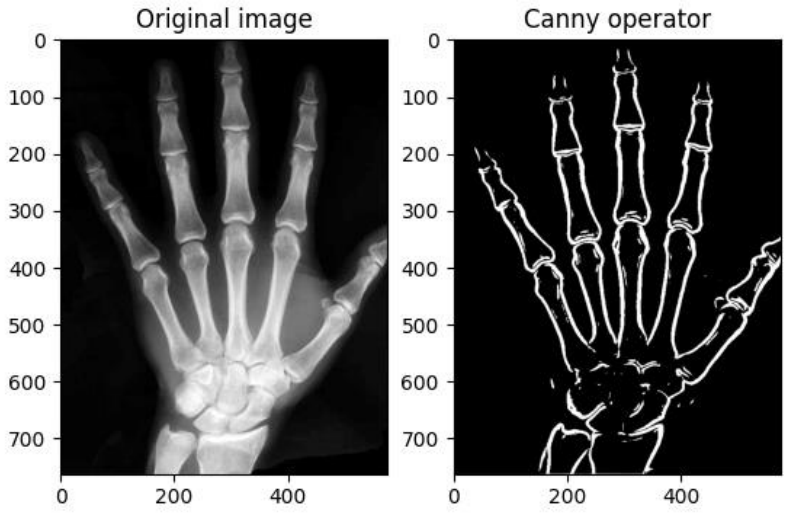


Рис. 15 Применение 4) и 5)

5. РЕЗУЛЬТАТЫ ТЕСТОВ

Перекрестный оператор Робертса хорошо показывает себя на тестах, т. к. выполняется он быстрее всех (благодаря матрице 2x2, а не 3x3 или больше, коэффициенты в матрице очень маленькие), без фильтра Гаусса на изображении мало шумов, контуры тонкие и видны хорошо.

Проблема в выделении контуров с помощью оператора Робертса может возникнуть, если изображение само по себе темное, либо если контраст на изображении небольшой.

На данный момент, оператор Робертса применяется из-за быстроты вычислений, в остальных показателях данный оператор проигрывает. Например, если нам подается на вход сильно зашумленное изображение, то в результате контуров даже не будет видно. В медицине таких изображений на данный момент нет, поэтому выбор оператора Робертса оправдан.

Оператор Робертса можно усовершенствовать, добавив, например, увеличение контраста в предобработку.

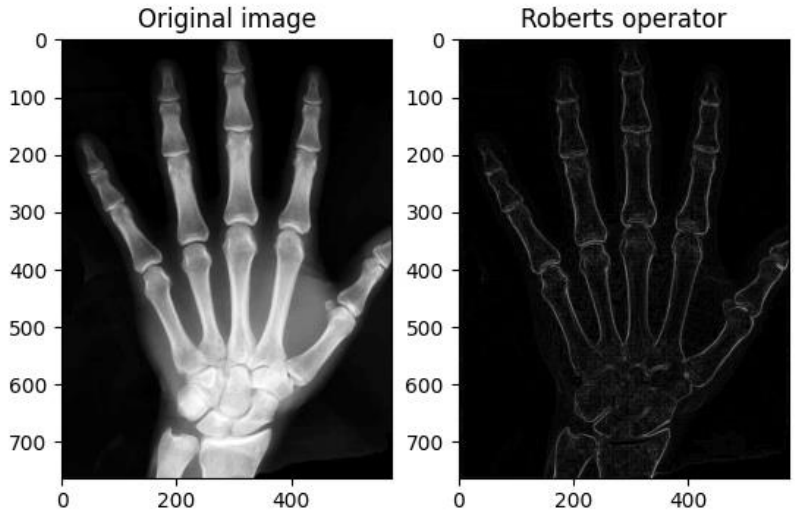


Рис. 16 Перекрестный оператор Робертса

Оператор Прюитт так же, как и оператор Робертса, эффективен в выполнении задачи. Лучше выделяет контуры на изображениях, чем оператор Робертса, но и работает дольше из – за того, что там в свертке участвуют две маски 3x3. Но из-за небольших коэффициентов, программа работает относительно быстро.

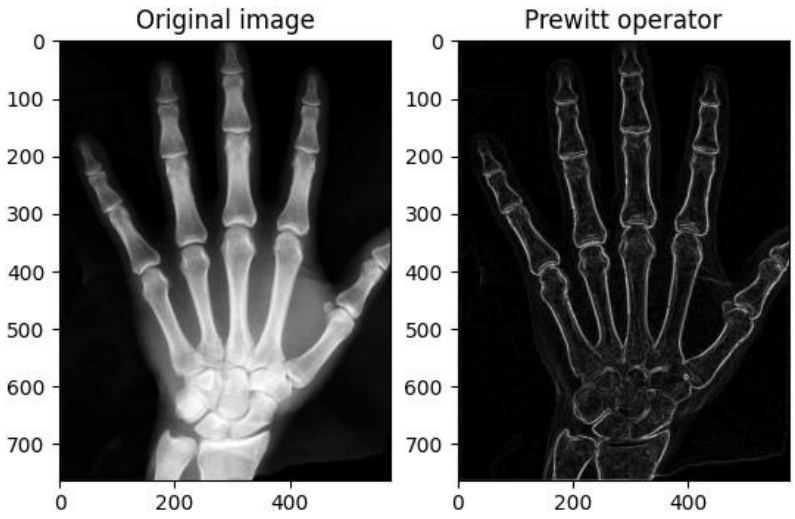


Рис. 17 Оператор Прюитт

Оператор Собеля сильно похож алгоритмом на оператор Прюитт, поэтому и результаты похожи. Но если брать тёмные изображения оператор Собеля, проявляет себя лучше, чем оператор Прюитт. Коэффициенты в масках Собеля не намного больше, чем в масках Прюитта, поэтому свертка занимает примерно одно и то же время. Поэтому оператор Собеля выигрывает у оператора Прюитт.

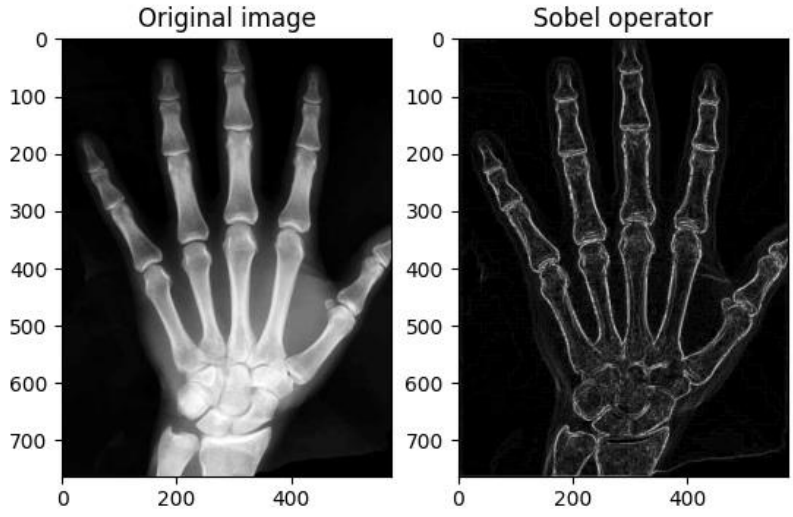


Рис. 18 Оператор Собеля

Оператор Шарра из-за больших весов, слишком сильно чувствителен к шуму, поэтому на практике он малоэффективен. Возможно улучшение – это применение фильтра Гаусса для удаления шумов, но результат будет похожим на оператор Собеля, поэтому лучше просто использовать оператор Собеля.

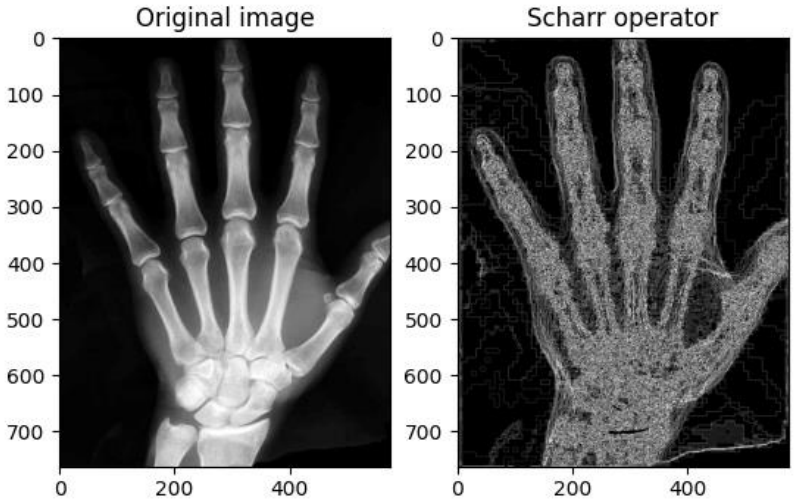


Рис. 19 Оператор Шарра

Оператор Лапласа сильно чувствителен к шуму. Контуры выделяются слабо, может быть применим из-за быстрой скорости работы, т. к. используется всего одна маска. Но время работы примерно одинаковое с временем работы перекрестного оператора Робертса, а результирующее изображение намного хуже, поэтому выбор данного оператора не оправдан.

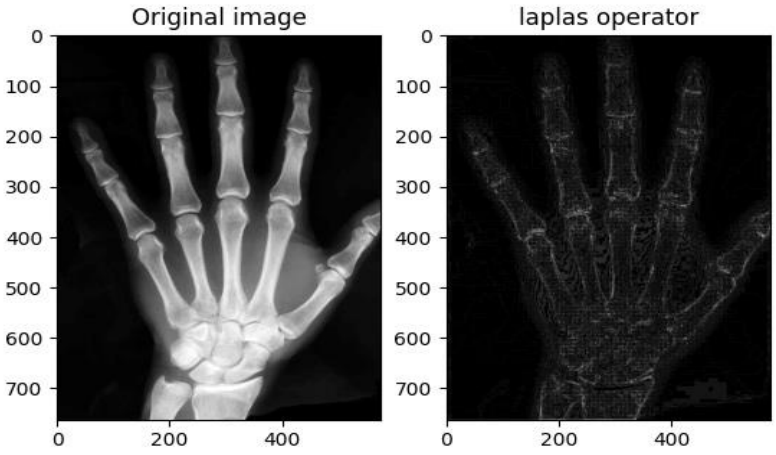


Рис. 20 Оператор Лапласа

Оператор Кирша работает очень долго даже на относительно мощных компьютерах, из-за вычисления большого количества сверток. Так же из-за больших весов маски, оператор сильно чувствителен к шуму. Поэтому данный оператор на практике малоэффективен.

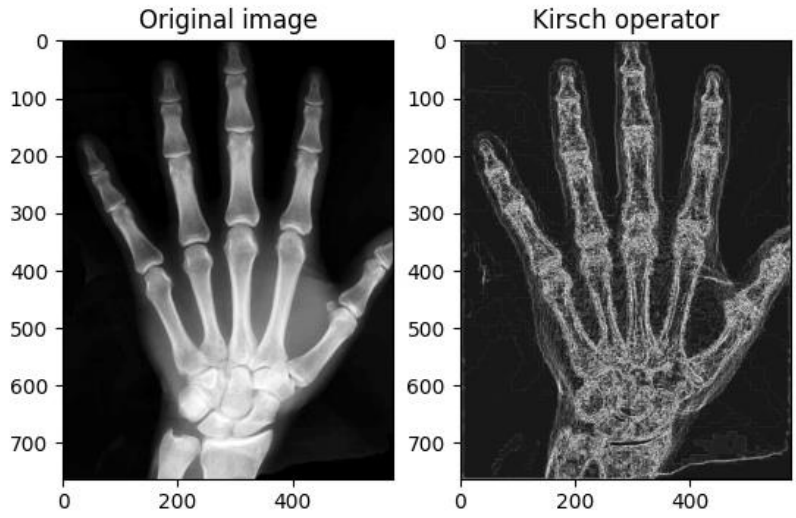


Рис. 21 Оператор Кирша

Оператор Робинсона по времени работает столько же, сколько и оператор Кирша. Был эффективен в XX веке, но с появлением оператора Кэнни стал малоэффективным, т.к время работы намного дольше, результат при этом хуже.

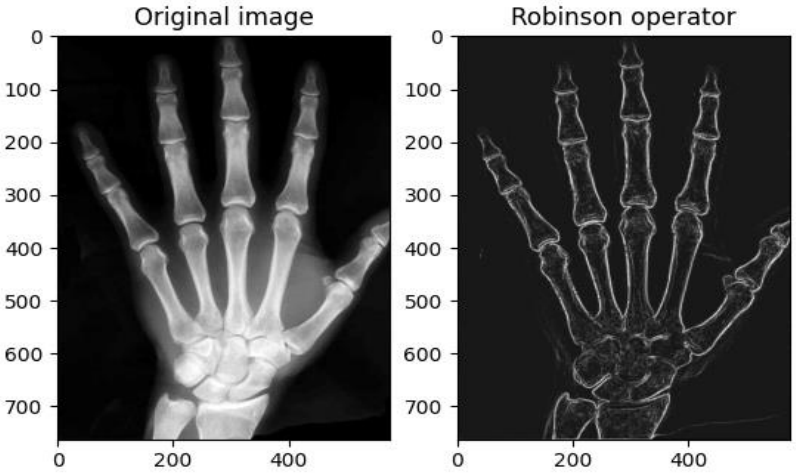


Рис. 22 Оператор Робинсона

Оператор Кэнни эффективен в выполнении задачи выделения контуров, т. к. время работы чуть больше, чем в операторе Собеля, но результирующее изображение намного лучше.

Из-за двойной пороговой фильтрации границы выделены ярко, при этом они тонкие. К шуму на медицинских изображениях практически не чувствителен из-за применения фильтра Гаусса.

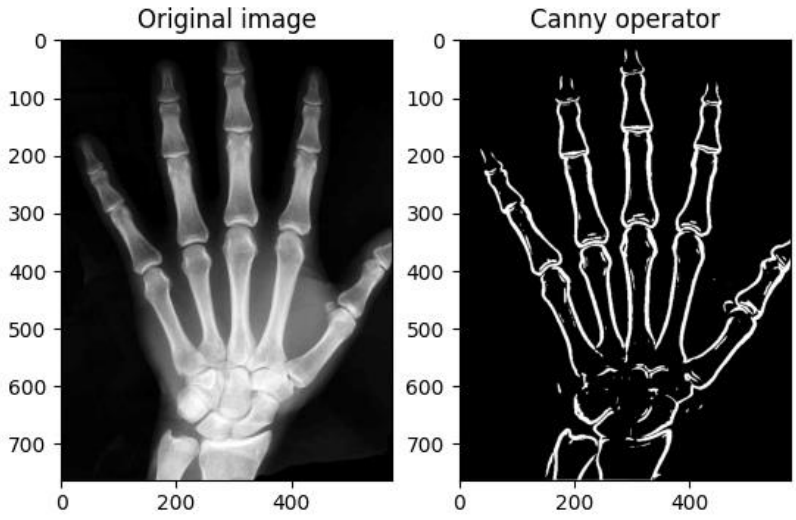


Рис. 23 Оператор Кэнни

ЗАКЛЮЧЕНИЕ

В данной курсовой работе была принята попытка изучить на практике и описать методы выделения контуров на медицинских изображениях.

В процессе работы над курсовой были изучены библиотеки OpenCV, PIL и Pyplot.

Было реализовано восемь операторов для выделения контуров. В результате были получены изображения с выделенными границами костей кисти.

В дальнейшем в Python лучше использовать готовые реализации операторов, т.к. код написан на C, использованы различные оптимизации (OpenCV уже реализованы оператор Кэнни, оператор Собеля и т.д).

Обработка изображений – быстро развивающаяся область в дисциплине компьютерного зрения.

Подводя итоги, можно сказать, что на данный момент нет универсального метода, который был бы эффективен и во времени, и в расходуемой памяти, и в результирующем изображении.

Например, если конечной целью является выделение контуров, то тогда оправдано использовать оператор Кэнни. Т.к., например, сделав рентгеновский снимок, можно подождать еще пару секунд, для выделения границ. А если же, к примеру, нам быстро нужно выделить контуры и затем передать изображение в качестве входного параметра в нейросеть (например, для постановления диагноза), то тогда лучше использовать перекрестный оператор Робертса, т. к. он выполняется намного быстрее, а дефекты в изображении на хорошо обученной нейросети не так критичны.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Н.Н. Красильников Цифровая обработка 2D- и 3D-изображений: учебное пособие. БХВ-Петербург, 2014. -С. 353-365.
2. Marr D., Hildreth E. Theory of edge detection // Proc. Royal Society of London. 1980. – P. 187-217
3. Документация к языку Python. – Режим доступа: https://docs.python.org/3.8/
4. Документация библиотеки OpenCV. – Режим доступа: https://docs.opencv.org/
5. Документация библиотеки PIL. – Режим доступа: https://pillow.readthedocs.io/en/stable/
6. М. М. Ибрагимов [Репозиторий]. – Режим доступа: https://github.com/Curryrasul/outlineSelection