



**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»**

Институт (Филиал) № 8 «Компьютерные науки и прикладная математика» Кафедра 806
Группа М8О-408Б-20 Направление подготовки 01.03.02 «Прикладная математика и информатика»
Профиль Информатика
Квалификация: бакалавр

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

БАКАЛАВРА

на тему: Создание системы преобразования рукописного математического текста в LaTeX

Автор ВКРБ: Ядров Артем Леонидович ()
Руководитель: Миронов Евгений Сергеевич ()
Консультант: Кондаратцев Вадим Леонидович ()
Консультант: — ()
Рецензент: — ()

К защите допустить

Заведующий кафедрой № 806 Крылов Сергей Сергеевич (_____
мая 2024 года

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ	3
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	4
ВВЕДЕНИЕ	5
1 ПОСТАНОВКА ЗАДАЧИ И ТЕОРЕТИЧЕСКИЕ ПРЕДПОСЫЛКИ	7
1.1 Пользовательские сценарии	7
2 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА	8
2.1 Высокоуровневая архитектура	8
2.2 Коррекция перспективы	8
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	16

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящей выпускной квалификационной работе бакалавра применяют следующие термины с соответствующими определениями:

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей выпускной квалификационной работе бакалавра применяют следующие сокращения и обозначения:

ВВЕДЕНИЕ

В России на постоянной основе проводятся научные исследования во многих областях. Результаты этих исследований публикуются в виде научных статей, которые являются важным инструментом для распространения новых знаний и научных открытий. Только в электронной версии научной библиотеки опубликовано 52573694 [1] статей, и все они написаны с помощью \LaTeX — мощного инструмента для верстки и оформления математических формул и научных текстов, который позволяет создавать качественные и профессионально оформленные статьи. Также с помощью \LaTeX можно готовить конспекты к предметам, причем это может делать как преподаватель, так и студент.

Однако, набор даже простых формул в LaTeX для неподготовленного человека может оказаться достаточно сложным и трудоемким занятием. Для примера возьмем формулу

$$f(x, y, \alpha, \beta) = \frac{\sum_{n=1}^{\infty} A_n \cos\left(\frac{2n\pi x}{\nu}\right)}{\prod \mathcal{F}_g(x, y)} \quad (1)$$

На рисунке 1 показан листинг \LaTeX кода этой формулы:

```
1 f(x,y,\alpha, \beta) = \frac{%
2   \sum \limits_{n=1}^{\infty} A_n \cos%
3     \left( \frac{2 n \pi x}{\nu} \right)%
4   }{%
5     \prod \mathcal{F}_g(x,y)%
6 }
```

Рисунок 1 – Листинг формулы 1

Как мы видим, используется много специальных символов (например, символ суммы, произведения, а также дроби, скобки и пр.), которые необходимо знать или тратить время для их поиска на просторах Интернета. В любом случае, требуется каждый раз компилировать pdf-файл для просмотра и проверки результата, что требуется некоторого количества времени.

В настоящее время появляется все больше различных нейросетей

(например, Гигачат, Yandex GPT, ChatGPT, stable diffusion, Midjonery и тд), в том числе преобразующие рукописный текст на изображении в машинный, а также способных генерировать готовый код. Поэтому появляется мотивация для автоматизации процесса преобразования формул из чистового варианта на бумаге в готовый \LaTeX -код.

Однако, мир не стоит на месте, и компания *Mathpix* придумала свое решение [2] этой задачи, которое распространяется по платной подписке, что не удовлетворяет требованию доступности ПО.

Целью работы является разработка программного обеспечения, выполняющего распознавание научного текста и генерацию готового \LaTeX кода.

Для достижения поставленной цели в работе были решены следующие задачи:

- а) Задачка
- б) Задачка еще одна:
 - 1) Подзадачка для задачки

Для разработки программного обеспечения необходимо изучить технологии и методы, решающие поставленные задачи. Работа основывается на следующих библиотеках, технологиях, алгоритмах:

- а) *Python* является основным языком программирования, который использовался для решения задач;
- б) *Tensorflow*
- в) *openCV* - библиотека для обработки изображения
- г) Что-то ещё

Результатом работы является:

- а) Результат

1 ПОСТАНОВКА ЗАДАЧИ И ТЕОРЕТИЧЕСКИЕ ПРЕДПОСЫЛКИ

Итак, перед нами стоит задача разработки прототипа платформы, позволяющей преобразовывать изображение с математическим текстом в \LaTeX -код.

1.1 Пользовательские сценарии

Для того, чтобы определить требования к нашей платформе, были проработаны пользовательские сценарии.

Для начала определим основные роли целевых пользователей:

- а) Пользователь мобильного приложения
- б) Пользователь *desktop* приложения
- в) Пользователь *web* приложения
- г) Программист

2 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

2.1 Высокоуровневая архитектура

Подводка к архитектуре. На рисунке 2 представлена общая арихеткура продукта.

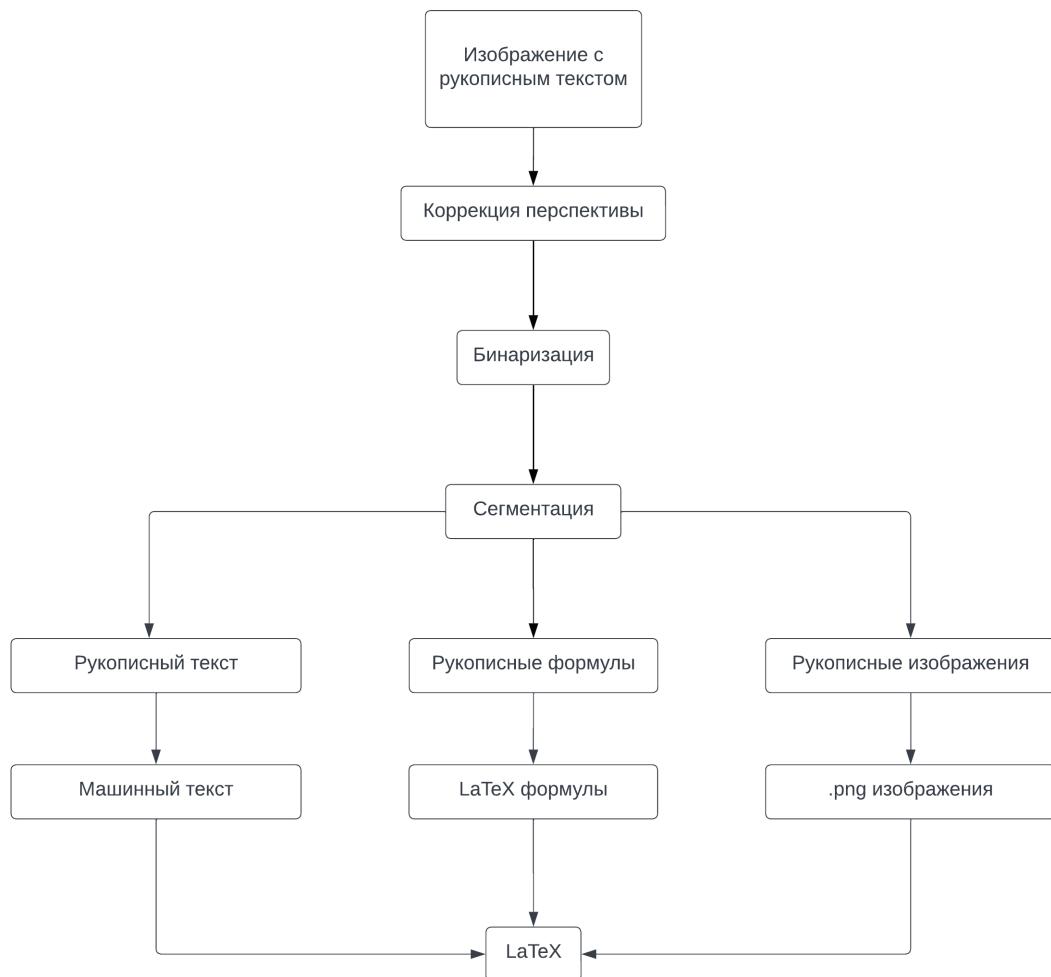


Рисунок 2 – Общая архитектура модели

2.2 Коррекция перспективы

Коррекция перспективы необходима для устранения шума на изображении и получения лучшего результата. Она состоит из нескольких этапов, представленных на рисунке 3. Также на рисунке представлены результаты, получаемые на каждом из этапов обработки.

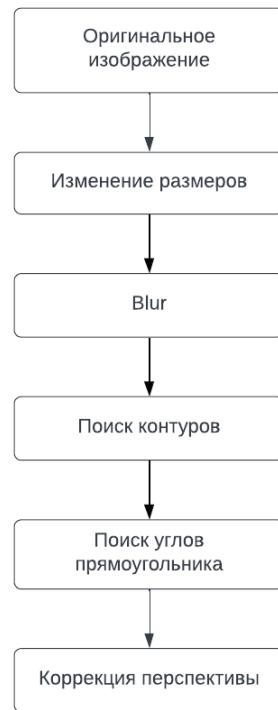


Рисунок 3 – Этапы коррекции перспективы изображения

На начальном этапе мы имеем изображение, показанное на рисунке 4

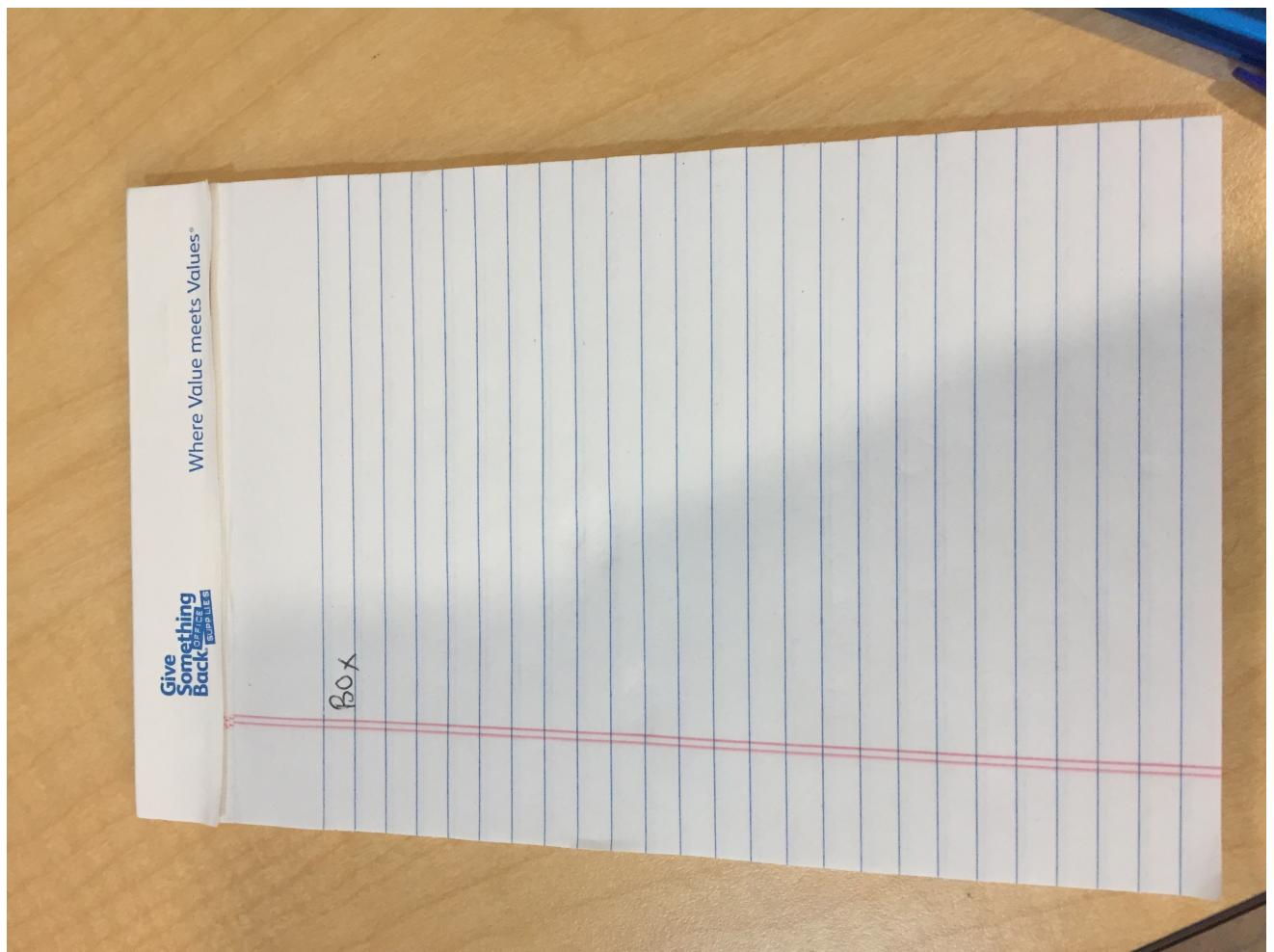


Рисунок 4 – Начальное изображение

Для начала необходимо удалить текст с изображения. Для этого преобразуем изображение в серый цвет и применим к нему размытие Гаусса [3]. На выходе данного этапа имеем изображение, представленное на рисунке 5

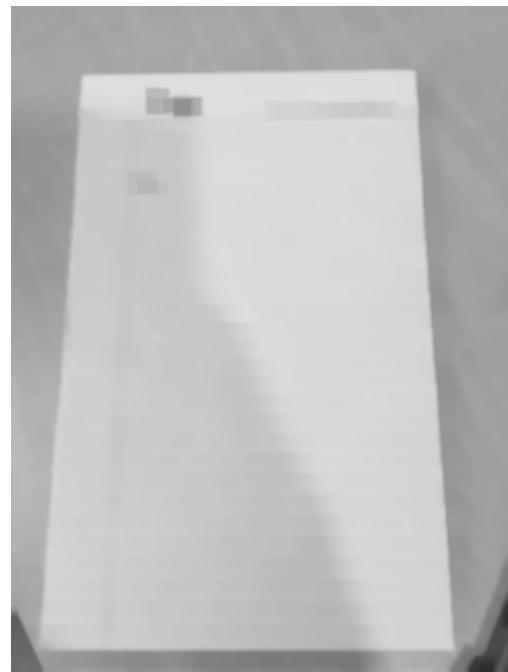


Рисунок 5 – Изображение после размытия Гаусса

Для поиска контуров необходимо выделить ребра. Для этого используется алгоритм Канни [4]. На выходе имеем изображение, представленное на рисунке 6

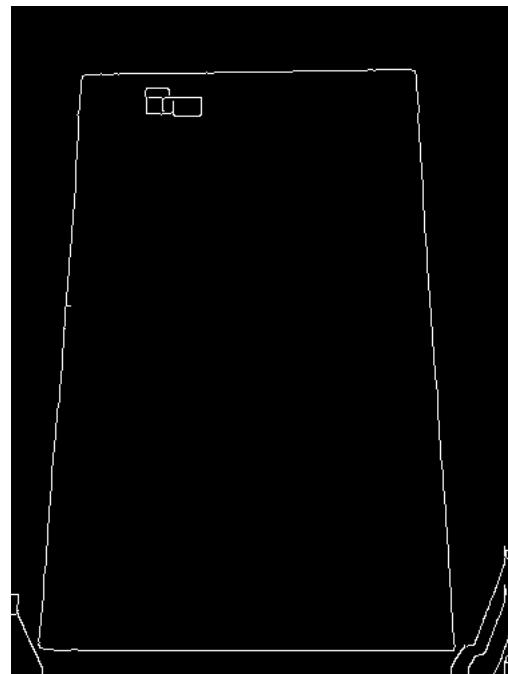


Рисунок 6 – Ребра, найденные на изображении

После нахождения ребер поиск контуров осуществляется двумя способами:

- С помощью алгоритма *LineSegmentDetector* [5]

б) С помощью встроенного в *openCV* алгоритма поиска контуров [6]

Опишем подробнее поиск контуров на основе найденных линий: после прохода алгоритма имеем изображение, представленное на рисунке 7

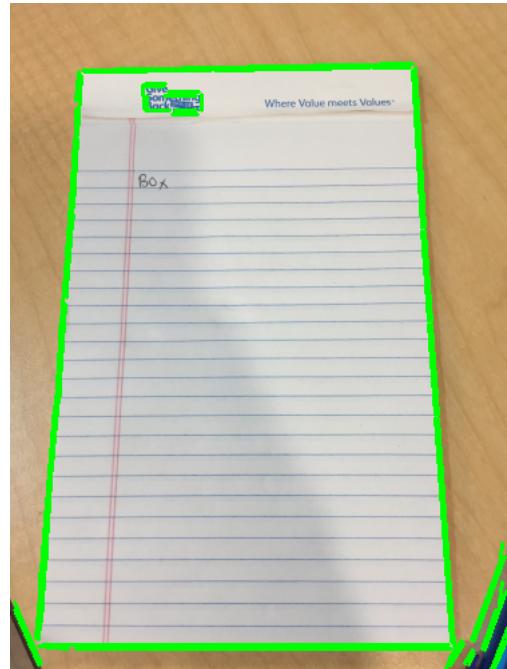


Рисунок 7 – Найденные на изображении линии с помощью алгоритма *LSD*

Контур определяется как пересечение горизонтальных и вертикальных линий. На выходе имеем найденные углы контура, показанные на рисунке 8

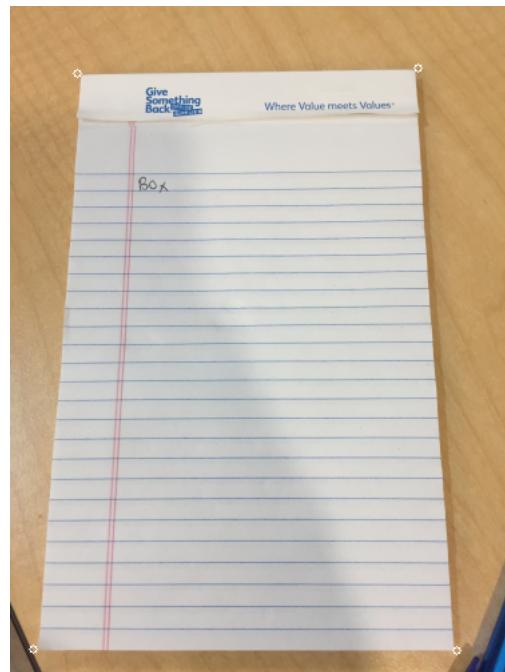


Рисунок 8 – Найденные на изображении контуры на основе линий

С помощью библиотеки *openCV* контуры находятся следующим образом:

- а) Находятся 5 наибольших по площади контуров
- б) Найденные контуры проверяются на количество углов, минимальную площадь контура

Среди всех подходящих контуров выбирается наибольший по площади, как показано на рисунке 9

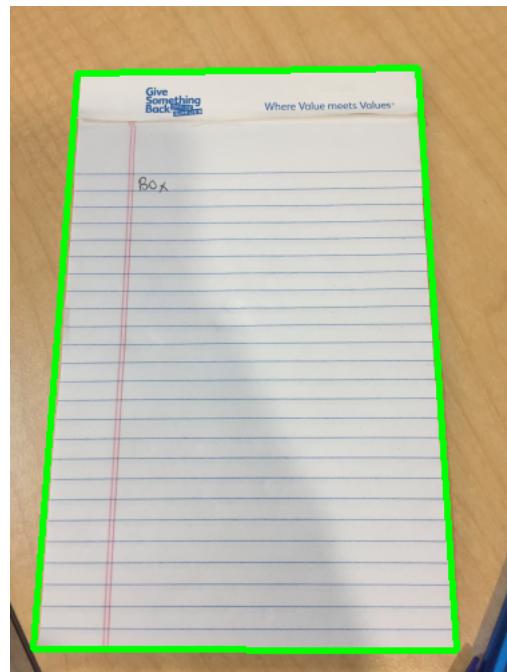


Рисунок 9 – Найденные алгоритмом контуры

На основе найденного контура, представляющего лист бумаги, осуществляем коррекцию перспективы. Для этого находим матрицу коррекции [7] и применим ее к изображению [8]. Получаем результирующее изображение, показанное на рисунке 10

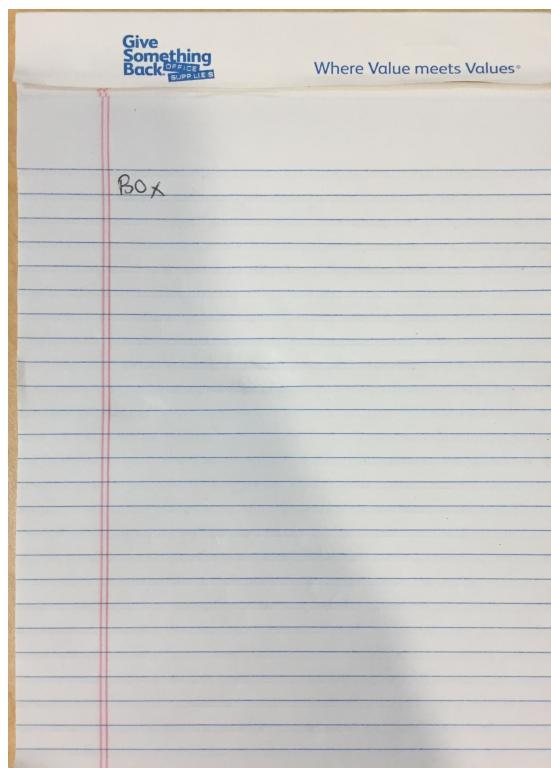


Рисунок 10 – Изображение с коррекцией перспективы

Далее необходимо добавить эффект сканирования. Эффект достигается путем применения к композиции небольшого размытия и серого изображения алгоритма сегментации *AdaptiveThreshold* [9]. В конечном итоге имеем результирующее изображение показанное на рисунке 11

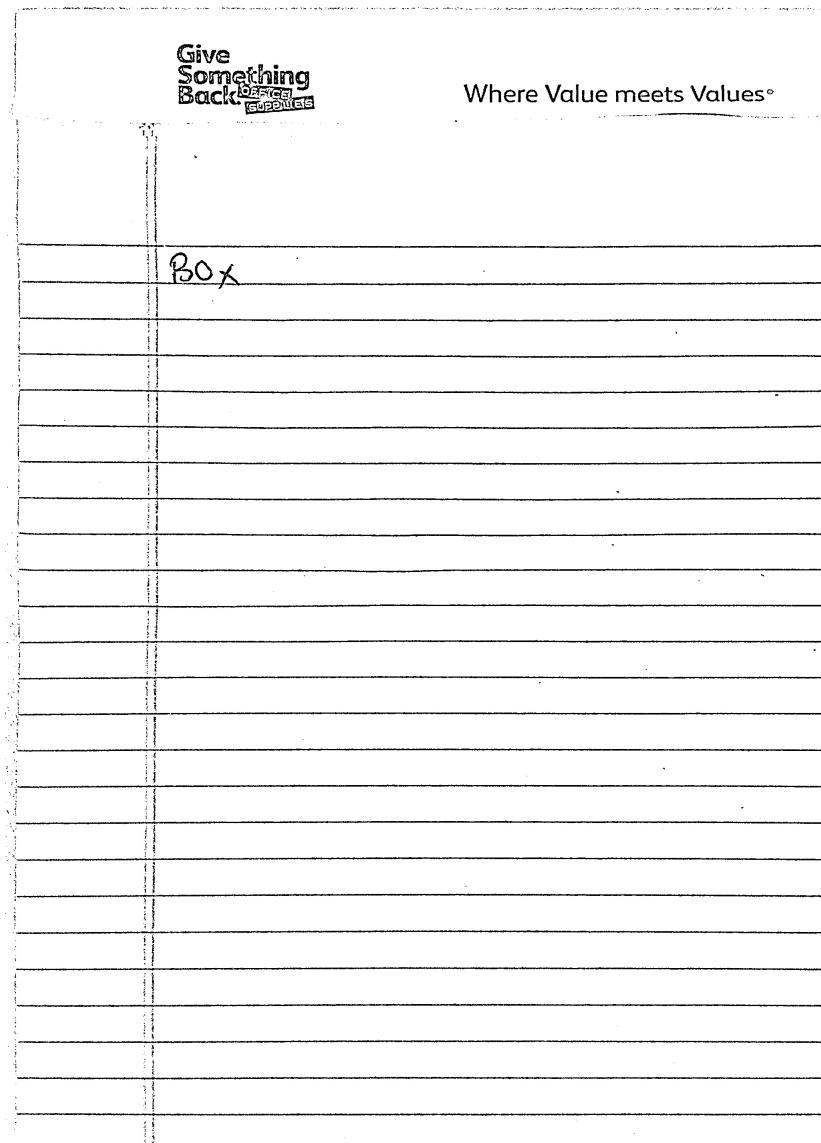


Рисунок 11 – Результирующее изображение

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. eLibrary. Научная электронная библиотека. — 2000. — URL: <https://www.elibrary.ru> (дата обращения 07.03.2024).
2. Mathpix. PDF to LaTeX. — URL: <https://mathpix.com/pdf-to-latex> (дата обращения 09.03.2024).
3. OpenCV. Smoothing Images. — URL: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html (дата обращения 25.04.2024).
4. OpenCV. Canny Edge Detection. — URL: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html (дата обращения 25.04.2024).
5. pylsd. Line Segment Detector. — URL: <https://github.com/primetang/pylsd> (дата обращения 25.04.2024).
6. OpenCV. Contours : Getting Started. — URL: https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html (дата обращения 25.04.2024).
7. OpenCV. Geometric Image Transformations. — URL: https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html (дата обращения 25.04.2024).
8. OpenCV. Geometric Image Transformations. — URL: https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html (дата обращения 25.04.2024).
9. OpenCV. Image Thresholding. — URL: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html (дата обращения 25.04.2024).