



**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 . . . . .	8
1.1 Пользовательские сценарии . . . . .	8
1.1.1 Авторизация пользователя . . . . .	8
1.1.2 Преобразование фотографии . . . . .	9
1.1.3 Преобразование скриншота . . . . .	10
1.2 Метрики . . . . .	10
2 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА . . . . .	11
2.1 Высокоуровневая архитектура нейронной сети . . . . .	11
2.2 Коррекция перспективы . . . . .	12
2.2.1 Неправильное распознавание . . . . .	19
2.3 Сегментация . . . . .	21
2.3.1 Сегментация на абзацы . . . . .	21
2.3.2 Выделение формул . . . . .	22
2.3.3 Обучающие данные . . . . .	22
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ . . . . .	23

## **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  $\text{\LaTeX}$  — мощного инструмента для верстки и оформления математических формул и научных текстов, который позволяет создавать качественные и профессионально оформленные статьи. Также с помощью  $\text{\LaTeX}$  можно готовить конспекты к предметам, причем это может делать как преподаватель, так и студент.

Однако, набор даже простых формул в  $\text{LaTeX}$  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  $\text{\LaTeX}$  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  $\text{\LaTeX}$ -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 прототипа платформы, выполняющего распознавание научного текста и генерацию готового  $\text{\LaTeX}$  кода.

Для достижения поставленной цели в работе были решены следующие задачи:

- 1) Определить требования к платформе
- 2) Спроектировать архитектуру платформы
- 3) Разработать:
  - 1) Сервис для распознавания научного текста
  - 2) Сервис для авторизации пользователя и синхронизации папок в *Google Drive*
  - 3) Приложение для взаимодействия пользователя с сервером
- 4) Протестировать прототип платформы на тестовом изображении

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* - библиотека для запуска и обучения моделей
- в) *YOLO* - модель, созданная для классификации объектов на изображении
- г) *openCV* - библиотека для обработки изображения
- д) *tkinter* - библиотека для работы с *GUI*
- е) *grpc* - фреймворк для удаленного вызова процедур
- ж) Что-то еще

Результатом работы является:

- а) Сервис для коррекции перспективы изображения

- б) Сервис для нахождения формул на изображении
- в) Дообученная модель *YOLO*, позволяющая находить формулы на изображении
- г) Сервис для преобразования найденных формул в  $\text{\LaTeX}$ -код
- д) *GUI* приложение с авторизацией в *Google Drive* и взаимодействием с удаленным сервером с помощью *grpc*

# 1 ПОСТАНОВКА ЗАДАЧИ И ТЕОРЕТИЧЕСКИЕ ПРЕДПОСЫЛКИ

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

## 1.1 Пользовательские сценарии

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

Для начала определим основные роли целевых пользователей:

- a) Пользователь приложения
- b) Программист, использующий *API* сервиса распознавания

Под пользователем приложения подразумевается пользователь одного из типов приложения:

- *WEB* приложение
- *Desktop* приложение
- мобильное приложение

### 1.1.1 Авторизация пользователя

Участники: пользователь любого из типов приложения

Предусловие: пользователь запускает приложение впервые или хочет зайти с другого аккаунта

Постусловие: пользователь авторизован через *Google* и предоставлен доступ к *Google Drive*

Сценарий: Для авторизации с помощью *Google* необходимо реализовать следующий сценарий, изображенный на рисунке 2:

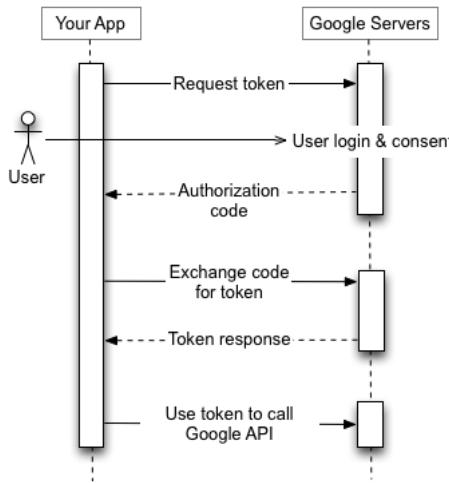


Рисунок 2 – Сценарий сетевого взаимодействия при авторизации пользователя

По сути *Google* делает всё сам на этапе создания сервиса внутри кода. Если приложение не видит токена для авторизации, то происходит автоматическое перенаправление на сайт авторизации. Дальнейший токен сохраняется локально.

Для замены аккаунта достаточно удалить локальный токен пользователя и перезапустить сервис *Google*. Стоит отметить, что лучше сохранять токен для дальнейшего быстрого входа в сервис. Но так как мы разрабатываем прототип, то и этого решения будет вполне достаточно.

### 1.1.2 Преобразование фотографии

Участники: пользователь приложения

Предусловие: пользователь сделал фото научного текста и открыл его в приложении

Постусловие: пользователь получает готовый *LATEX* код текста, полученного на изображении в виде архива из исходного кода и *pdf* файла

Сценарий:

- Производится автоматическая коррекция перспективы фотографии
- Производится автоматическое сегментация текста на абзацы
- С приложения на сервер отправляется изображение, а также таблица с координатами начала и конца абзацев
- С сервера на приложение отправляется таблица с найденными формулами

- д) Пользователь проверяет правильность распознавания формул и вносит корректиды
- е) Приложение отправляет на сервер таблицу с финальными формулами
- ж) Сервер загружает архив с L<sup>A</sup>T<sub>E</sub>X кодом и *pdf* файлом в *Google Drive*, а также отсылает его пользователю
- и) *Desktop* приложение обновляет папку *Google Drive* и загружает в локальное хранилище последний архив

При необходимости пользователь может корректировать точки перспективы, а также точки абзацев на изображении.

### **1.1.3 Преобразование скриншота**

Участники: пользователь приложения

Предусловие: пользователь сделал скриншот научного текста и открыл приложение

Постусловие: пользователь получает готовый L<sup>A</sup>T<sub>E</sub>X код текста, полученного на изображении в виде архива из исходного кода и *pdf* файла

Сценарий: да вроде такой же сценарий, ничем не отличается...

## **1.2 Метрики**

Ну тут стандартные, когда-нибудь будет что-то и про них...

## **2 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА**

Конечный продукт является целостной системой, и необходимо рассказать про каждую ее часть. Пожалуй, стоит начать с самой главной ее части - нейросетевой.

### **2.1 Высокоуровневая архитектура нейронной сети**

Поскольку преобразование изображения в текст нетривиальная задача, было решено разбить её на несколько модулей. Каждый из модулей можно охарактеризовать входными данными и результатом работы модуля (выходными данными). Такой подход обладает рядом преимуществ:

- а) Гибкость и масштабируемость: Модульная структура позволяет легко добавлять новые компоненты или модифицировать существующие без необходимости переписывать всю модель.
- б) Ускорение процесса обучения: Благодаря возможности параллельной обработки данных, модульные нейронные сети обучаются быстрее, чем монолитные модели.
- в) Улучшение качества модели: Разделение модели на модули позволяет специалистам сосредоточиться на оптимизации каждого компонента, что в итоге приводит к улучшению общей производительности модели.
- г) Простота внедрения новых технологий: Модульная архитектура облегчает внедрение новых технологий и подходов, таких как трансферное обучение или диффузионные модели.
- д) Улучшение производительности: некоторые модули могут исполняться в препроцессинге на клиентской машине, что ослабляет нагрузку на сервер.

Однако, имеются и недостатки:

- а) Проблемы с совместимостью: Разные модули могут использовать различные архитектуры, форматы данных и методы обучения, что может привести к проблемам совместимости.
- б) Риск ухудшения производительности: Неправильно спроектированные или плохо интегрированные модули могут снизить общую производительность модели.
- в) Необходимость в дополнительных ресурсах: Для обучения и

развертывания модульных моделей часто требуются дополнительные ресурсы, такие как GPU или TPU.

Несмотря на недостатки, в современных реалиях важно уметь быстро и без проблем масштабироваться и заменять при необходимости один компонент другим, поэтому было принято решение использовать модульную архитектуру. Такая представлена на рисунке 3

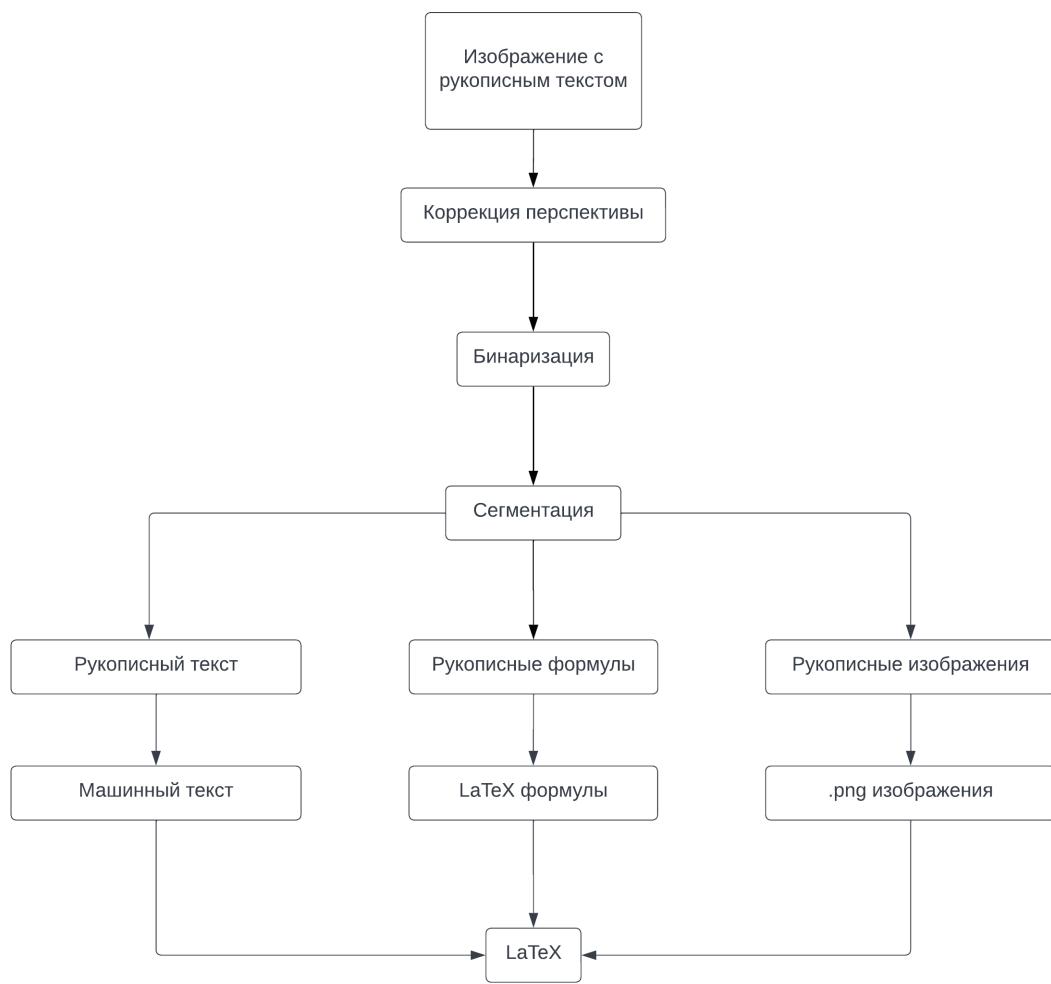


Рисунок 3 – Общая архитектура модели

Подробнее разберем каждый этап данной схемы.

## 2.2 Коррекция перспективы

Коррекция перспективы необходима для устранения шума на изображении и получения лучшего результата. Она состоит из нескольких этапов, представленных на рисунке 4. Также на рисунке представлены результаты, получаемые на каждом из этапов обработки.

Стоит отметить, что коррекция перспективы осуществляется с использованием только алгоритмов обработки изображения без использования каких-либо нейросетей. Поэтому в целях экономии ресурсов сервера, а в следствии улучшения производительности было принято решение выполнять данный этап на машине клиента. Схема данного алгоритма представлена на рисунке 4.

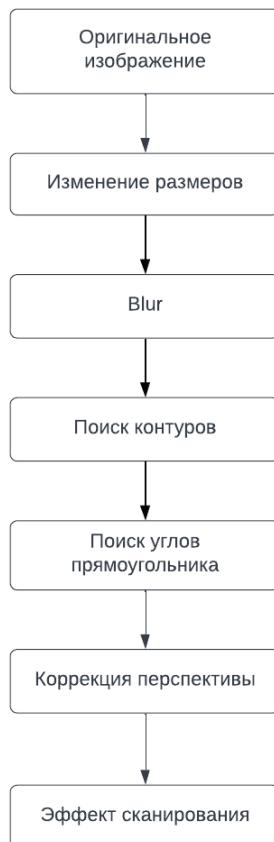


Рисунок 4 – Этапы коррекции перспективы изображения

На начальном этапе мы имеем изображение, показанное на рисунке 5

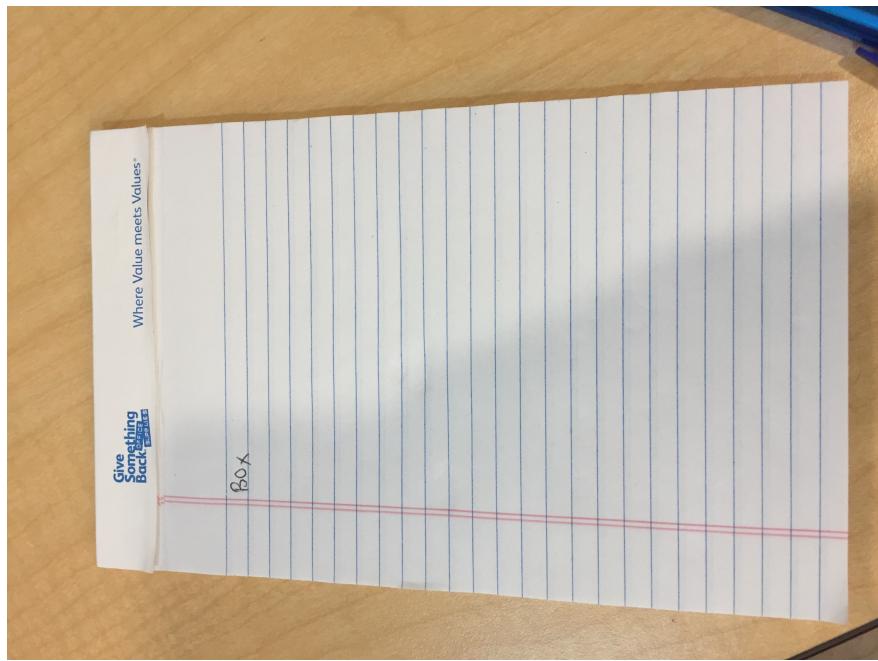


Рисунок 5 – Начальное изображение

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 6

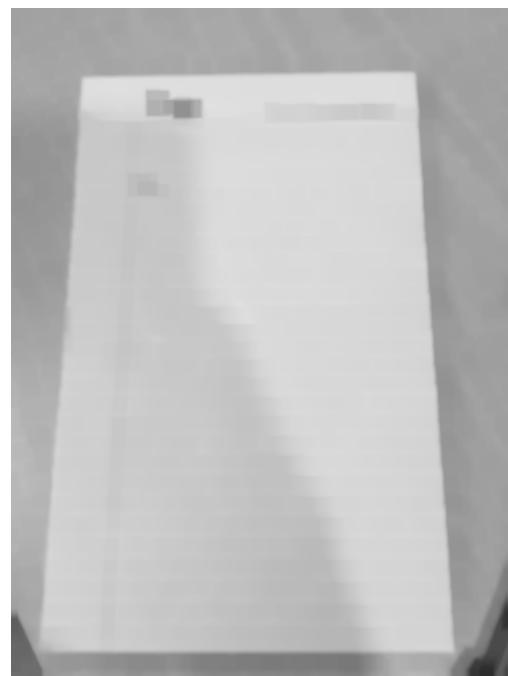


Рисунок 6 – Изображение после размытия Гаусса

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

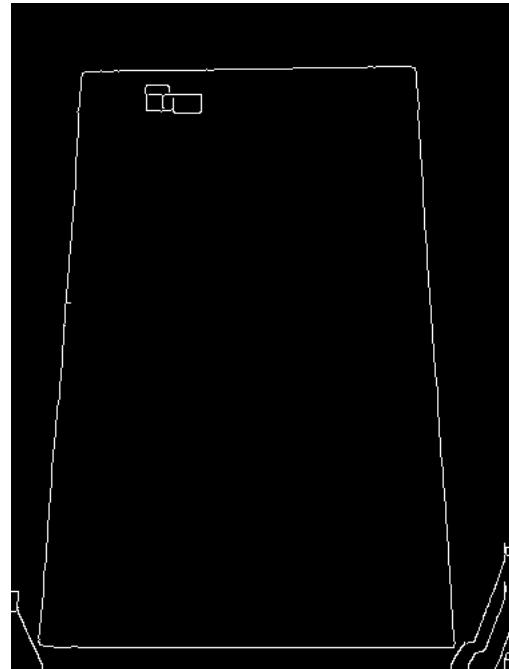


Рисунок 7 – Ребра, найденные на изображении

После нахождения ребер поиск контуров осуществляется двумя способами:

- a) 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 8

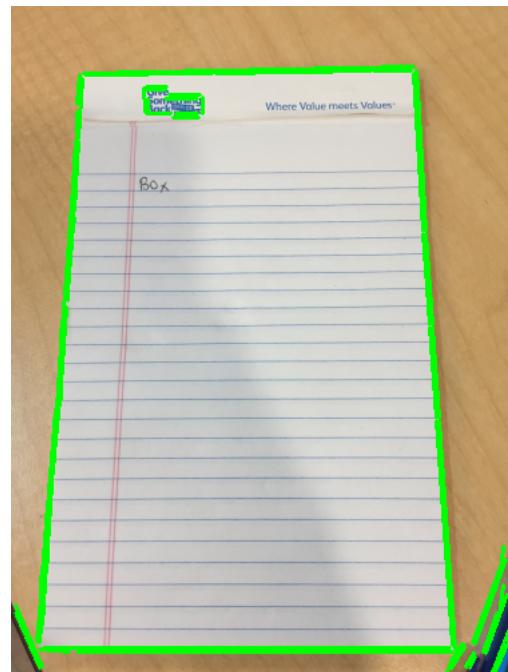


Рисунок 8 – Найденные на изображении линии с помощью алгоритма *LSD*

Контур определяется как пересечение горизонтальных и вертикальных линий. На выходе имеем найденные углы контура, показанные на рисунке 9

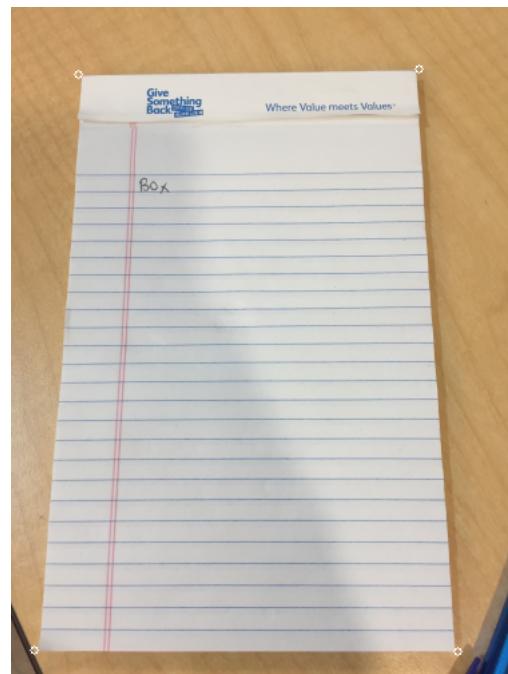


Рисунок 9 – Найденные на изображении контуры на основе линий

С помощью библиотеки *openCV* контуры находятся следующим образом:

- a) 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 10

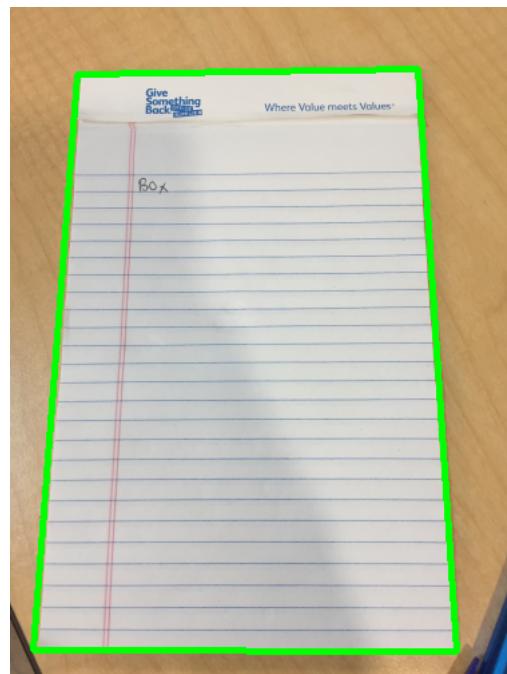


Рисунок 10 – Найденные алгоритмом контуры

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 11

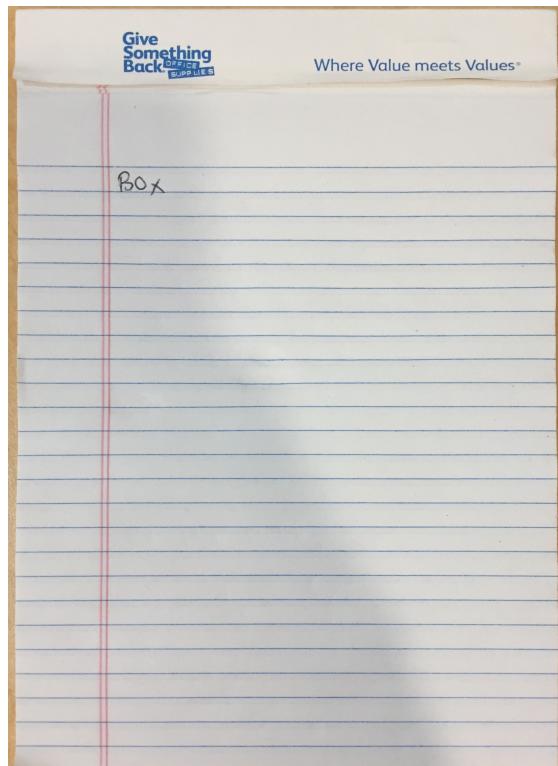


Рисунок 11 – Изображение с коррекцией перспективы

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 12

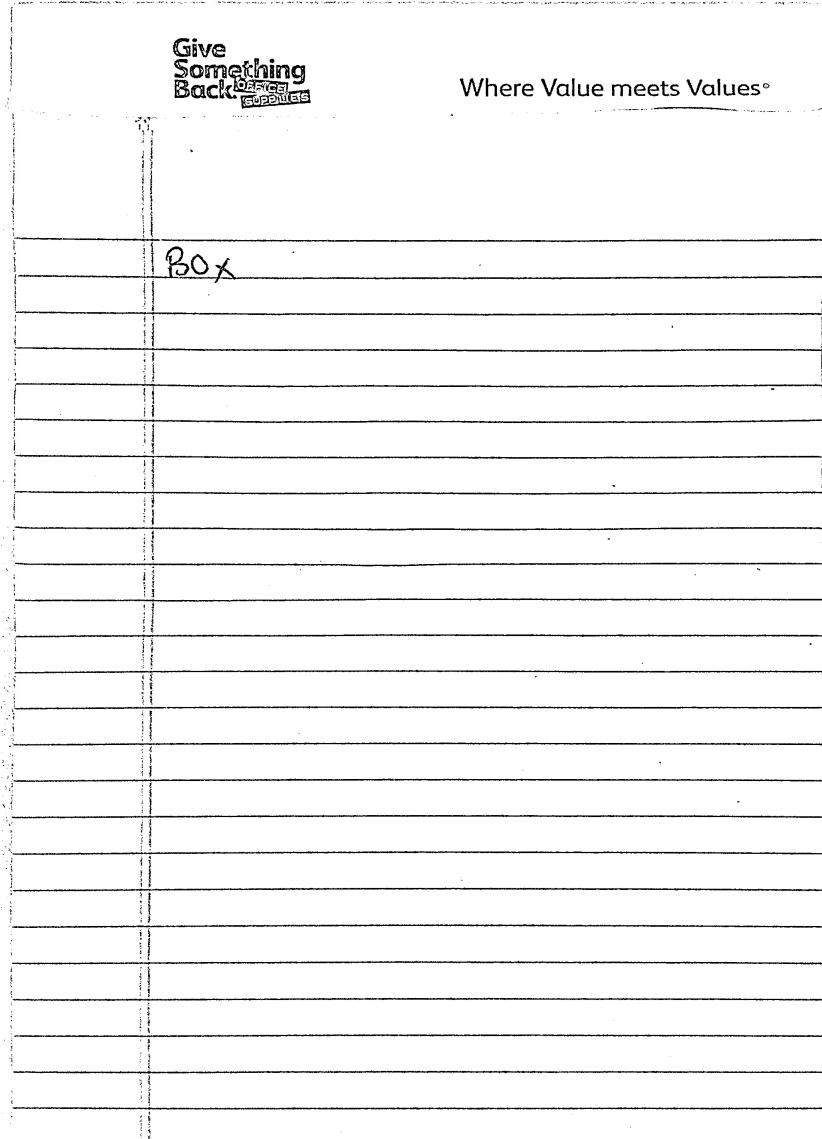


Рисунок 12 – Результирующее изображение

### 2.2.1 Неправильное распознавание

Стоит отметить, что данный алгоритм не является ультимативным. Например, если на изображении находится посторонний шум (например, палец на бумаге, экран монитора, большое здание в виде прямоугольника), то алгоритм не сможет найти или найдет неверный контур. Именно с целью защиты от таких случаев в конечном продукте пользователь должен удостовериться в правильности найденного контура и отредактировать границы контура при необходимости. Пример входного изображения, дающего неверный результат, и найденные на нем контуры приведены на рисунках 13 и 14 соответственно.

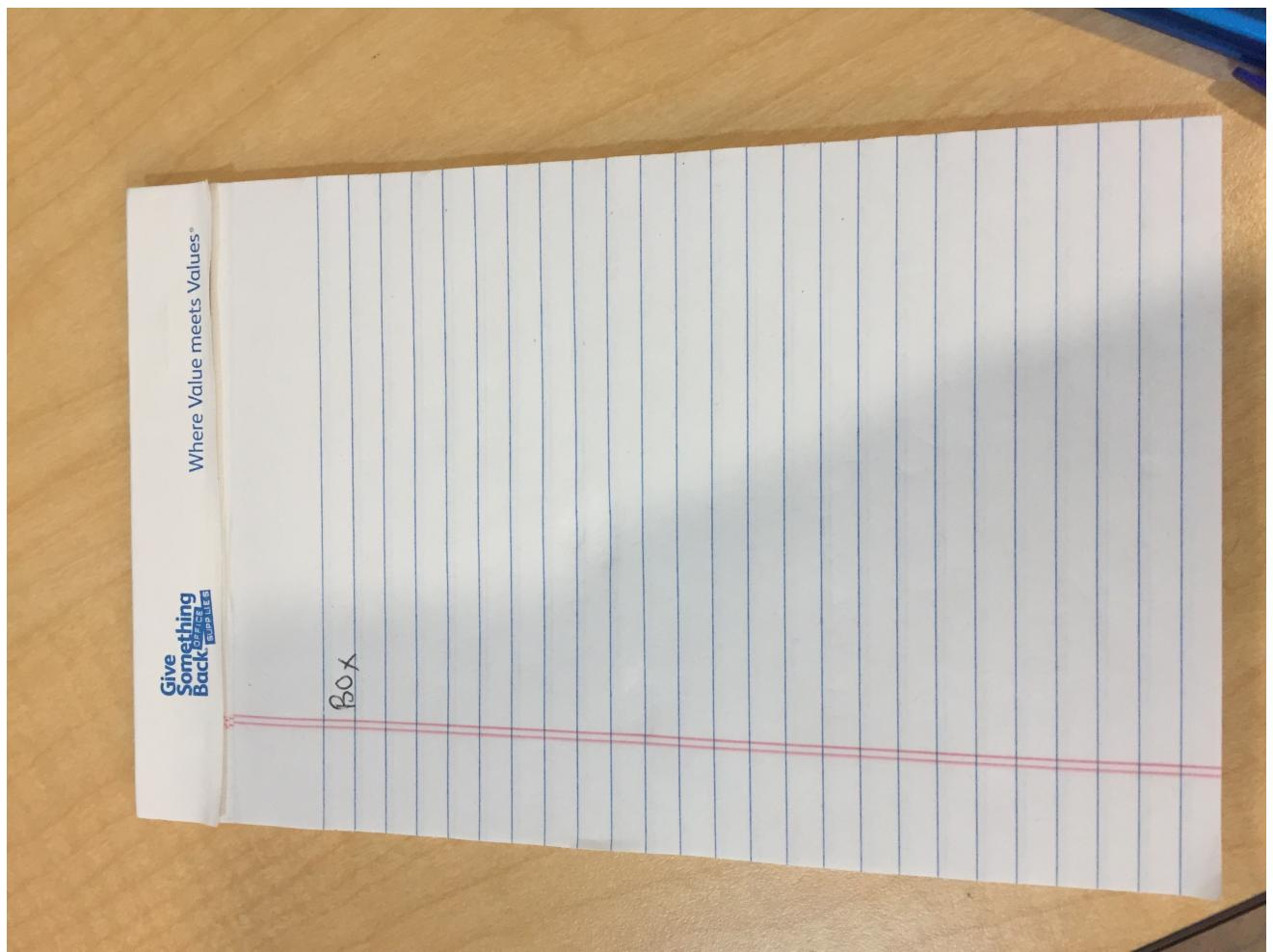


Рисунок 13 – Пример неправильно распознанного изображения:  
входное изображение

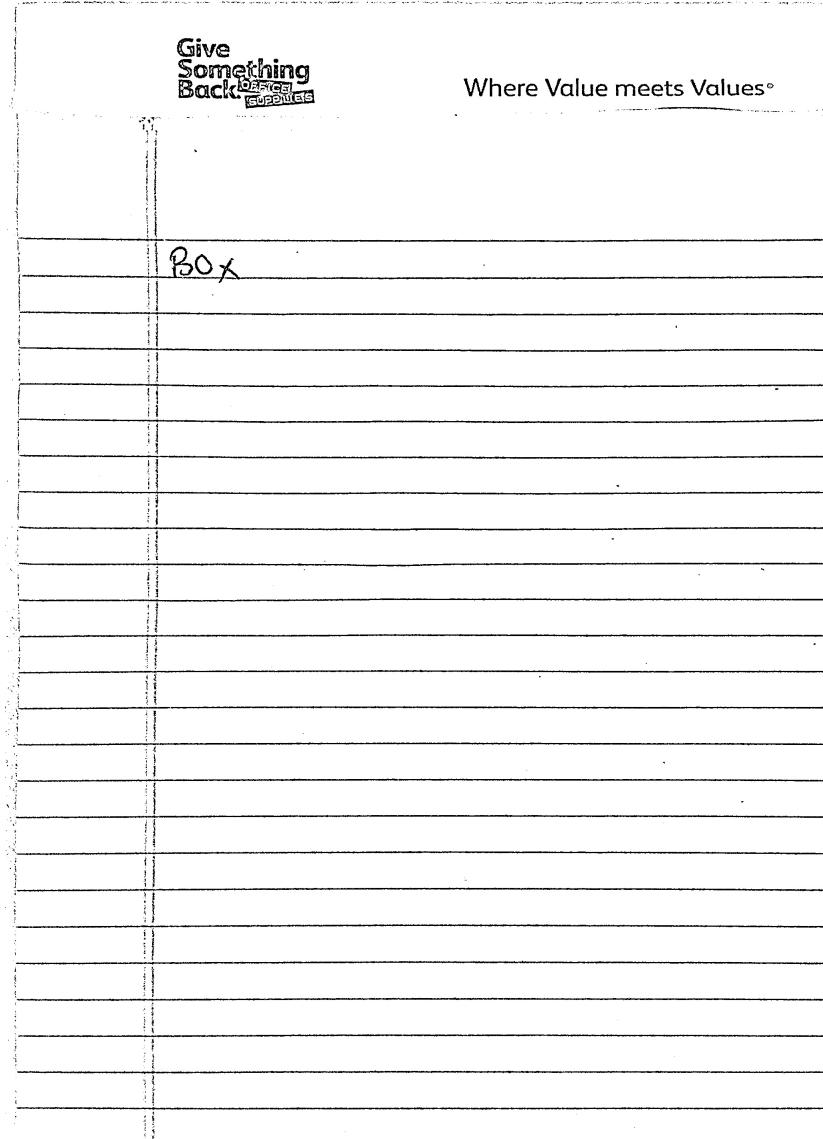


Рисунок 14 – Пример неправильно распознанного изображения:  
найденные контуры

## 2.3 Сегментация

Сегментация состоит из двух отдельных этапов:

- Сегментация на абзацы
- Отделение текста от формул, рисунков и пр.

### 2.3.1 Сегментация на абзацы

Выделение абзацев требуется для их дальнейшего сохранения в L<sup>A</sup>T<sub>E</sub>X коде. Данный вид сегментации, как и коррекция перспективы, осуществляется исключительно алгоритмически. Поэтому данный этап можно также выполнять на машине пользователя.

### **2.3.2 Выделение формул**

Для выделения формул недостаточно одних алгоритмов обработки изображений. Существует множество моделей, выполняющих распознавание объектов на изображении. Одной из таких моделей является *YOLO*. Данная модель обладает следующими преимуществами:

- Быстродействие. Нейросеть работает в реальном времени, поэтому её используют для распознавания объектов на фото и видео ”здесь и сейчас”.
- Точность. Нейросеть *YOLO* умеет распознавать объекты разных размеров в пределах одного кадра.
- Универсальность. Нейросеть *YOLO* способна определять как хорошо знакомые ей объекты, так и те, с которыми она ещё не сталкивалась.
- Простота. Модель *YOLO* можно запросто запускать и дообучать с помощью *Tensorflow*.

Однако, данная модель не специализирована на какой-то одной задаче, поэтому будет проигрывать специализированным моделям.

На основании плюсов данной модели, а также на основании самой архитектуры системы, позволяющую при необходимости легко заменить выбранную модель на другую, было принято решение использовать для распознавания формул модель *YOLO* последней версии *v8*.

### **2.3.3 Обучающие данные**

## **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](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](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](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](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](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](https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html) (дата обращения 25.04.2024).