МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ
ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
Национальный исследовательский ядерный университет «МИФИ»
Институт Лазерных и Плазменных Технологий
Кафедра № 97 «Суперкомпьютерное моделирование
инженерно-физических процессов»

# ОТЧЁТ
# Учебная практика (научно-исследовательская работа) на тему:

Подготовка базы свойств материалов на основе открытых источников

**Работу выполнил:**
А. Е. Диков
Группа: Б20-221
**Научный руководитель:**
Д. П. Макаревич

Москва
2023

# Содержание

# 1. Введение

## 1.1. Актуальность проблемы

В современном информационном обществе существует огромное количество данных, которые хранятся в различных форматах, включая оцифрованные изображения документов. Вместе с тем, большое количество информации представлено в виде таблиц, которые являются основным средством структурирования и организации данных. Однако, извлечение данных из изображений таблиц является нетривиальной задачей, которая требует автоматизации и использования специализированных технологий.

Проблема извлечения данных из изображений таблиц имеет высокую актуальность в различных сферах деятельности, таких как научные исследования, финансовый анализ, медицина, юридические и бухгалтерские отчеты и многие другие. Сфокусируем внимание на применении в научной сфере, а именно, для сбора данных о параметрах материалов, используемых в матемаческом моделировании. Автоматическое распознавание и извлечение данных из изображений таблиц позволяет значительно ускорить и упростить процесс обработки информации.

## 1.2. Трудности реализации задачи

Задача извлечения данных из изображений таблиц является сложной и требует решения нескольких проблем. Во-первых, требуется точное обнаружение и сегментация таблицы на изображении. Это включает в себя предобработку изображений, выделение контуров, определение границ и разделение на ячейки.

Во-вторых, необходимо правильно распознать текст и числа внутри каждой ячейки таблицы. Распознавание текста визуально представленного на изображении является сложной задачей из-за различных шрифтов, размеров, стилей и размещения текста в ячейках таблицы.

В-третьих, требуется структурировать распознанные данные в виде таблицы с соответствующими заголовками столбцов и строками. Иными словами, сохранить общий вид исходной таблицы.

## 1.3. Используемые технологии

Для решения задачи извлечения данных из изображений таблиц используются различные технологии и методы. Ниже приведены некоторые из них:

### 1.3.1. Обработка изображений с использованием OpenCV

OpenCV (Open Source Computer Vision Library) является библиотекой с открытым исходным кодом на языке C++, предназначенной для обработки изображений и компьютерного зрения. Она предоставляет мощные инструменты для обработки и анализа изображений, включая выделение контуров, сегментацию, фильтрацию и морфологические операции над изображениями. [1]

### 1.3.2. Распознавание текста с помощью библиотеки Tesseract

Tesseract - это библиотека с открытым исходным кодом для распознавания текста на изображениях. Она использует алгоритмы машинного обучения для распознавания текста и может работать с различными языками. Tesseract обладает хорошей точностью распознавания и широкими возможностями настройки.

Ядро Tesseract было разработано в лаборатории Hewlett Packard в середине 1980-х и поддерживалось до середины 1990-х, а затем 10 лет никакой поддержки не было. С 2006 г. Google купил Tesseract и открыл исходные тексты под лицензией Apache 2.0 для продолжения разработки. В настоящий момент программа поддерживает кодировку UTF-8 и множество языков, включая русский.[**2**]

# 2. Постановка задачи

В качестве набора данных были предоставлены ГОСТ, технические условия и справочники в различных форматах - rtr, djvu, pdf. В дополнению к этому был сгенерирован еще один набор данных, имеющий меньшую вариативность наполнения, но имеющий метки - файлы в более удобном формате (был выбран csv) для оценки качества работы алгоритма (размеченные данные). Задача состояла в выделении таблицы из корпуса текста в виде изображений. Затем нужно было распознать таблицу с помощью алгоритмов компьютерного зрения так, чтобы сохранилась общая сруктура таблицы. Необходимо было создать универсальную и масштабируемую систему, точно работающую для всех входных файлов.

**Механические свойства фасонного проката**

| Наименование стали | Толщина полки, мм | Механические характеристики | | | Изгиб до параллельности сторон ($a$—толщина образца, $d$—диаметр оправки) | Ударная вязкость KCU, Дж/см² (кгс · м/см²) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Предел текучести $\sigma_т$, Н/мм² (кгс/мм²) | Временное сопротивление $\sigma_в$, Н/мм² (кгс/мм²) | Относительное удлинение $\delta_5$, % | | при температуре, °С | | | после механического старения |
| | | | | | | —20 | —40 | —70 | |
| | | не менее | | | | не менее | | | |
| С235 | От 4 до 20 включ. | 235(24) | 360(37) | 26 | $d = a$ | — | — | — | — |
| | Св. 20 » 40 » | 225(23) | 360(37) | 25 | $d = 2a$ | — | — | — | — |
| С245 | От 4 до 20 включ. | 245(25) | 370(38) | 25 | $d = a$ | — | — | — | 29(3)* |
| | Св. 20 » 25 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | 29(3) |
| | » 25 » 30 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | — |
| С255 | От 4 до 10 включ. | 255(26) | 380(39) | 25 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 245(25) | 370(38) | 25 | $d = a$ | 29(3) | — | — | 29(3) |
| | » 20 » 40 » | 235(24) | 370(38) | 24 | $d = 2a$ | 29(3) | — | — | 29(3) |
| С275 | От 4 до 10 включ. | 275(28) | 390(40) | 24 | $d = a$ | — | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 380(39) | 23 | $d = a$ | — | — | — | 29(3) |
| С285 | От 4 до 10 включ. | 285(29) | 400(41) | 24 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 390(40) | 23 | $d = a$ | 29(3) | — | — | 29(3) |

Рисунок 2.1. Пример таблицы

# 3. Алгоритмы, технологии и программный код

## 3.1. Обработка входных данных

Для решения задачи напишем класс на языке Python, полями которого будут подаваемый на вход файл, из которого нужно извлечь таблицы, и базы данных (pandas.dataFrame), в которых будет храниться результат работы.

```
class TableExtraction:
    def __init__(self):
        self.input_data = []
        self.list_of_np_arrays = []
        self.threshold_images = []
        self.parts_boxes = []
        self.parts_image = []
        self.result = []

    def convert_file_to_array(self, file_path):...

    def binarization(self):...

    def split_pages(self):...

    def extract(self):...
```

Для простоты будем считать, что входной файл всегда имеет формат pdf.

```
def convert_file_to_array(self, file_path):
    file_extension = os.path.splitext(file_path)[1]
    if file_extension == '.pdf':
        doc = fitz.open(file_path)
        for n in range(doc.page_count):
            page = doc.load_page(n)
            pix = page.get_pixmap()
            image = np.frombuffer(pix.samples, dtype=np.uint8).reshape(pix.h,
                pix.w, pix.n)
            image = np.ascontiguousarray(image[..., [2, 1, 0]])
            self.list_of_np_arrays.append(image)
        doc.close()
        else:
            raise ValueError('Unsupported file format')
```

Отсканированные изображения страниц документов в общем случае могут быть цветными, что затормозит работу алгоритмов поиска контуров и распозования букв в дальнейшем, поэтому опустим информацию о цвете. Более того, отбросим и оттенки серого, тем самым бинаризовав изображение. [**3**]

```
def binarization(self):
    for image in self.list_of_np_arrays:
        gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        _, threshold_image = cv2.threshold(gray_image, 0, 255,
            cv2.THRESH_BINARY)
            self.threshold_images.append(threshold_image)
```

**Механические свойства фасонного проката**

| Наименование стали | Толщина полки, мм | Механические характеристики: Предел текучести $\sigma_т$, Н/мм² (кгс/мм²) | Временное сопротивление $\sigma_в$, Н/мм² (кгс/мм²) | Относительное удлинение $\delta_5$, % | Изгиб до параллельности сторон (a—толщина образца, d—диаметр оправки) | Ударная вязкость KCU, Дж/см² (кгс · м/см²) при температуре, °С: −20 | −40 | −70 | после механического старения |
|---|---|---|---|---|---|---|---|---|---|
| | | не менее | | | | не менее | | | |
| С235 | От 4 до 20 включ. | 235(24) | 360(37) | 26 | $d = a$ | — | — | — | — |
| | Св. 20 » 40 » | 225(23) | 360(37) | 25 | $d = 2a$ | — | — | — | — |
| С245 | От 4 до 20 включ. | 245(25) | 370(38) | 25 | $d = a$ | — | — | — | 29(3)* |
| | Св. 20 » 25 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | 29(3) |
| | » 25 » 30 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | — |
| С255 | От 4 до 10 включ. | 255(26) | 380(39) | 25 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 245(25) | 370(38) | 25 | $d = a$ | 29(3) | — | — | 29(3) |
| | » 20 » 40 » | 235(24) | 370(38) | 24 | $d = 2a$ | 29(3) | — | — | 29(3) |
| С275 | От 4 до 10 включ. | 275(28) | 390(40) | 24 | $d = a$ | — | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 380(39) | 23 | $d = a$ | — | — | — | 29(3) |
| С285 | От 4 до 10 включ. | 285(29) | 400(41) | 24 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 390(40) | 23 | $d = a$ | 29(3) | — | — | 29(3) |

Рисунок 3.1. Черно-белое изображение

**Механические свойства фасонного проката**

| Наименование стали | Толщина полки, мм | Механические характеристики: Предел текучести $\sigma_т$, Н/мм² (кгс/мм²) | Временное сопротивление $\sigma_в$, Н/мм² (кгс/мм²) | Относительное удлинение $\delta_5$, % | Изгиб до параллельности сторон (a—толщина образца, d—диаметр оправки) | Ударная вязкость KCU, Дж/см² (кгс · м/см²) при температуре, °С: −20 | −40 | −70 | после механического старения |
|---|---|---|---|---|---|---|---|---|---|
| | | не менее | | | | не менее | | | |
| С235 | От 4 до 20 включ. | 235(24) | 360(37) | 26 | $d = a$ | — | — | — | — |
| | Св. 20 » 40 » | 225(23) | 360(37) | 25 | $d = 2a$ | — | — | — | — |
| С245 | От 4 до 20 включ. | 245(25) | 370(38) | 25 | $d = a$ | — | — | — | 29(3)* |
| | Св. 20 » 25 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | 29(3) |
| | » 25 » 30 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | — |
| С255 | От 4 до 10 включ. | 255(26) | 380(39) | 25 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 245(25) | 370(38) | 25 | $d = a$ | 29(3) | — | — | 29(3) |
| | » 20 » 40 » | 235(24) | 370(38) | 24 | $d = 2a$ | 29(3) | — | — | 29(3) |
| С275 | От 4 до 10 включ. | 275(28) | 390(40) | 24 | $d = a$ | — | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 380(39) | 23 | $d = a$ | — | — | — | 29(3) |
| С285 | От 4 до 10 включ. | 285(29) | 400(41) | 24 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 390(40) | 23 | $d = a$ | 29(3) | — | — | 29(3) |

Рисунок 3.2. Бинаризованное изображение

## 3.2. Поиск таблиц в документе

Теперь будем рассматривать документ как набор черно-белых изображений и все сказанное ниже применять для каждой страницы документа. Для нахождения таблицы на странице, сделаем копию изображения и применим к ней инвертированную бинаризацию и морфологическое сверточное преобразование. Оно размоет границы букв, замкнет линии и уменьшит возможные шумы, возникающие в результате сканирования. [4]

```
1 kernel = np.ones((5, 5), np.uint8)
2 morph_image = cv2.morphologyEx(threshold_image, cv2.MORPH_CLOSE, kernel)
```

С. 4 ГОСТ 27772—88

2.13.1. Допускается химический анализ стали на содержание хрома, никеля (кроме стали С345К и С590К), меди (кроме стали С345К, С345Д, С375Д, С390Д, С390К и С440Д), мышьяка и азота (кроме стали С390, С390К, С440 и С590), алюминия (кроме стали С345К и С590К), а в стали С235 также хрома и в стали С590К титана изготовителю не проводить. Требуемый химический состав гарантируется изготовителем. В стали, выплавляемой из керченских руд, определение мышьяка обязательно.

2.13.2. Допускается химический анализ готового проката изготовителю не проводить. Установленные нормы гарантируются изготовителем.

2.14. Прокат изготовляют в горячекатаном состоянии. Для обеспечения требуемых свойств допускается применение термической обработки.

Листы из стали С390, С390К и С440 изготовляют в нормализованном или улучшенном состоянии, листы из стали С590 и С590К — в улучшенном состоянии.

2.15. Состояние поверхности и кромок для листового и широкополосного универсального проката должно соответствовать требованиям ГОСТ 14637 и ГОСТ 16523, фасонного проката — ГОСТ 535, подгруппы 1. Зачистка поверхности проката допускается на глубину, не выводящую за пределы минусовых отклонений.

2.16. Плоскостность листового проката должна соответствовать требованиям ГОСТ 19903. Вид плоскостности оговаривается в заказе. Для листового проката из стали С590, С590К толщиной до 20 мм включительно отклонения от плоскостности должны быть не более 15 мм на 1 м длины, толщиной свыше 20 мм — не более 12 мм на 1 м длины.

2.17. Расслоение проката не допускается.

По сплошности при проведении ультразвукового контроля прокат должен соответствовать классам 0, 1, 2, 3 ГОСТ 22727.

Необходимость проведения УЗК и класс сплошности указывают в заказе.

2.18. Свариваемость стали гарантируется изготовителем.

По требованию потребителя углеродный эквивалент ($C_э$) должен быть для стали С390 и С390К не более 0,49 %, стали С440 — не более 0,51 %.

2.19. Механические свойства при растяжении, ударная вязкость, а также условия испытаний на изгиб должны соответствовать для фасонного проката требованиям табл. 3, листового и широкополосного универсального — табл. 4.

Таблица 3

Механические свойства фасонного проката

| Наименование стали | Толщина полки, мм | Механические характеристики: Предел текучести $\sigma_т$, Н/мм² (кгс/мм²) | Временное сопротивление $\sigma_в$, Н/мм² (кгс/мм²) | Относительное удлинение $\delta_5$, % | Изгиб до параллельности сторон (a—толщина образца, d—диаметр оправки) | Ударная вязкость KCU, Дж/см² (кгс · м/см²) при температуре, °С: −20 | −40 | −70 | после механического старения |
|---|---|---|---|---|---|---|---|---|---|
| | | не менее | | | | не менее | | | |
| С235 | От 4 до 20 включ. | 235(24) | 360(37) | 26 | $d = a$ | — | — | — | — |
| | Св. 20 » 40 » | 225(23) | 360(37) | 25 | $d = 2a$ | — | — | — | — |
| С245 | От 4 до 20 включ. | 245(25) | 370(38) | 25 | $d = a$ | — | — | — | 29(3)* |
| | Св. 20 » 25 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | 29(3) |
| | » 25 » 30 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | — |
| С255 | От 4 до 10 включ. | 255(26) | 380(39) | 25 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 245(25) | 370(38) | 25 | $d = a$ | 29(3) | — | — | 29(3) |
| | » 20 » 40 » | 235(24) | 370(38) | 24 | $d = 2a$ | 29(3) | — | — | 29(3) |
| С275 | От 4 до 10 включ. | 275(28) | 390(40) | 24 | $d = a$ | — | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 380(39) | 23 | $d = a$ | — | — | — | 29(3) |
| С285 | От 4 до 10 включ. | 285(29) | 400(41) | 24 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 390(40) | 23 | $d = a$ | 29(3) | — | — | 29(3) |

Рисунок 3.3. Инвертированно бинаризованное изображение

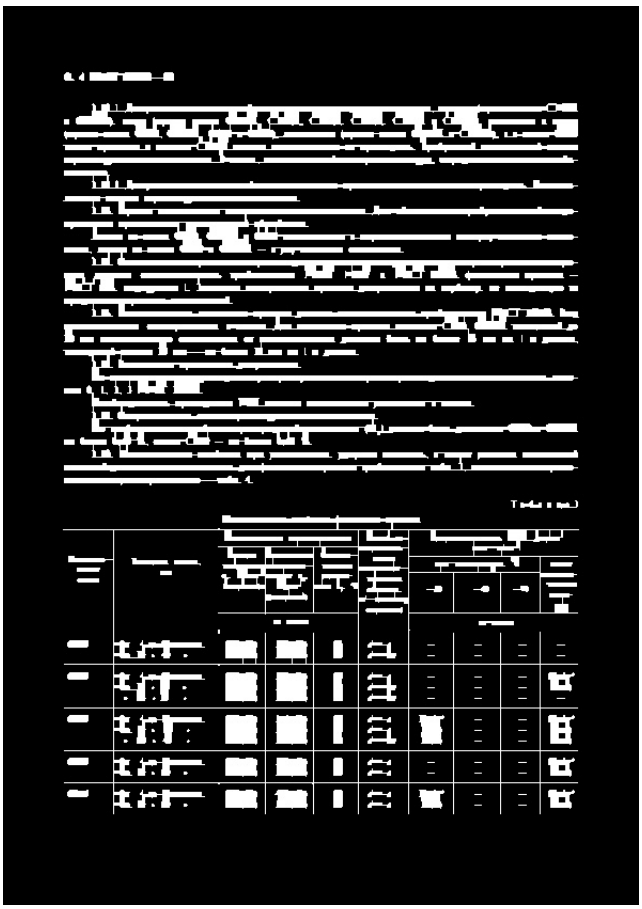


Рисунок 3.4. Изображение, после применения морфологического преобразования

Применим к предобработанному изображению метод findContours(). Внутри метода findContours() в OpenCV используется алгоритм обхода границы объектов, известный как алгоритм Сузуки и Абе.[**5**]

Алгоритм работает следующим образом:

1. Имеется бинарное изображение, где объекты обозначаются белым цветом, а фон - черным цветом. Каждый пиксель бинарного изображения имеет значение "0" или "1".

2. Сканируем изображение и находим первую непосещенную точку на границе объекта (белый пиксель) и присваивает ей имя.

3. Следует по границе объекта, перемещаясь от текущей точки к следующей, пока не вернется в исходную точку. Каждая точка границы добавляется в список контура.

4. Маркируем посещенные точки на границе объекта, чтобы не посещать их повторно.

5. Процесс повторяется для остальных непосещенных объектов на изображении, пока не будут обнаружены все контуры.

Особенности алгоритма, связанные с восстановлением иерархии границ, при решении задачи не понадобятся, поэтому их опустим.

```
1 contours, hierarchy = cv2.findContours(morph_image, cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE)
```

Здесь режим `cv2.RETR_EXTERNAL` позволяет извлечь только внешние контуры, игнорируя внутренние, а `cv2.CHAIN_APPROX_SIMPLE` возвращает координаты ограничивающих эти контуры прямоугольников.

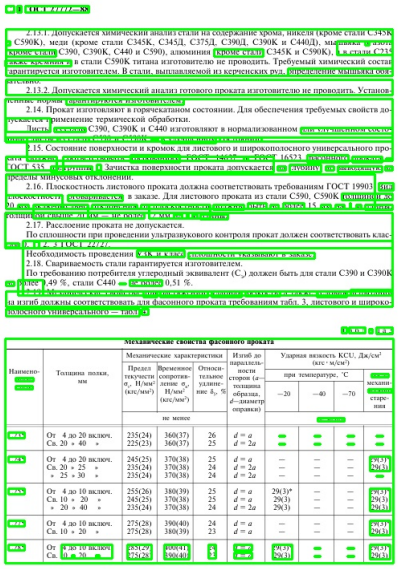


Рисунок 3.5. Результат работы cv2.findContours

Далее надо провести фильтрацию контуров. Заметим, что таблица занимает большее пространство, чем каждое из слов и введем эмперический критерий по площади контура. Сохраним как координаты предполагаемой таблицы, так и вырезанное со страницы ее изображение.

```
1 for contour in contours:
2     area = cv2.contourArea(contour)
3     if area > 3000:
4         x, y, w, h = cv2.boundingRect(contour)
5         self.parts_boxes.append((x, y, x + w, y + h))
6         x1, y1, x2, y2 = self.parts_boxes[-1]
7         self.parts_image.append(image[y1:y2, x1:x2])
```

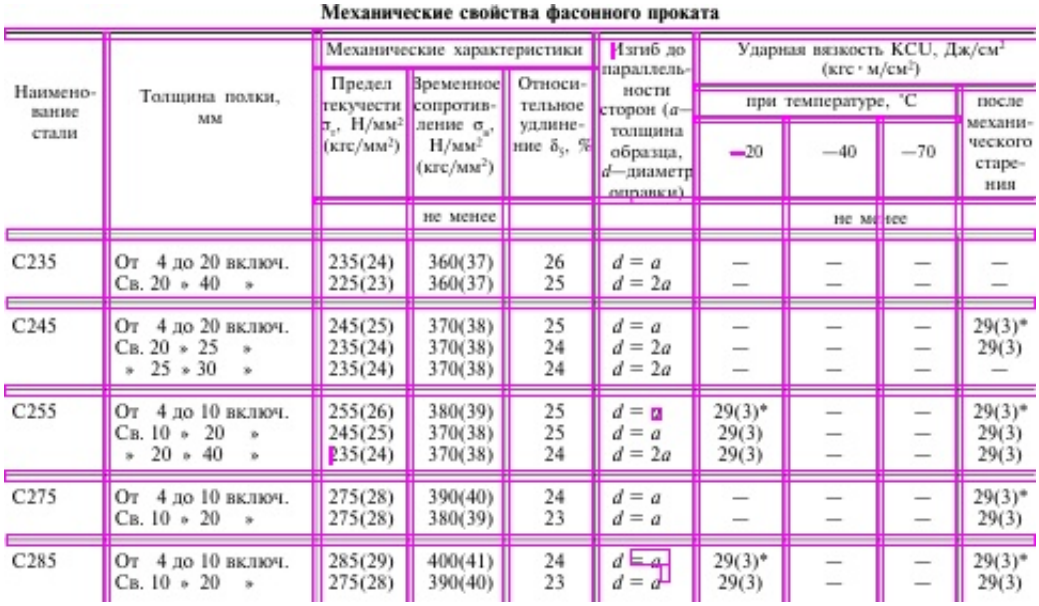
## 3.3. Работа с таблицей

Далее будем работать в основном с библиотекой Tesseract, поэтому обозначим, что внутри этой библиотеки есть множество методов машинного обучения и специализированных алгоритмов, применяемых для конкретных этапов обработки текста. После сегментации текста алгоритмом, аналогичным описанному выше алгоритму Сузуки и Абе, изображения символов поступают в сверточную рекуррентную нейронную сеть для классификации. Сверточные преобразовании позволяют получить инвариантность относительно преобразований поворота, сдвига и искажений, в то время как рекуррентность дает лучшие результаты исходя из того, что текст это последовательность - ряд - в котором предыдущие элементы влияют на последующие. Архитектура этой сети фиксированная, сеть предобучена на большом наборе данных и дает хорошую точность. [2]

Tesseract позволяет классифицировать найденные объекты, в том числе линии. Поэтому для нахождения ячеек таблицы воспользуемся методом `pytesseract.image_to_boxes`.

```
boxes = pytesseract.image_to_boxes(threshold_image, lang='rus')
box_lines = boxes.strip().split('\n')
boxes_list = [line.split() for line in box_lines]
height, width = gray_image.shape
borders = []
epsilon = 10

for box in boxes_list:
    if box[0] == '~' and ((abs(int(box[1]) - int(box[3])) <= epsilon) or
       (abs(int(box[2]) - int(box[4])) <= epsilon)):
        x, y, w, h = int(box[1]), int(box[2]), int(box[3]), int(box[4])
        borders.append([x, y, w, h])
        cv2.rectangle(image, (x, height - y), (w, height - h), (255, 0, 255), 1)
```



**Механические свойства фасонного проката**

| Наименование стали | Толщина полки, мм | Механические характеристики: Предел текучести $\sigma_т$, Н/мм² (кгс/мм²) | Временное сопротивление $\sigma_в$, Н/мм² (кгс/мм²) | Относительное удлинение $\delta_5$, % | Изгиб до параллельности сторон ($a$—толщина образца, $d$—диаметр оправки) | Ударная вязкость KCU, Дж/см² (кгс · м/см²) при температуре, °С: +20 | −40 | −70 | после механического старения |
|---|---|---|---|---|---|---|---|---|---|
| | | не менее | | | | не менее | | | |
| С235 | От 4 до 20 включ. | 235(24) | 360(37) | 26 | $d = a$ | — | — | — | — |
| | Св. 20 » 40 » | 225(23) | 360(37) | 25 | $d = 2a$ | — | — | — | — |
| С245 | От 4 до 20 включ. | 245(25) | 370(38) | 25 | $d = a$ | — | — | — | 29(3)* |
| | Св. 20 » 25 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | 29(3) |
| | » 25 » 30 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | — |
| С255 | От 4 до 10 включ. | 255(26) | 380(39) | 25 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 245(25) | 370(38) | 25 | $d = a$ | 29(3) | — | — | 29(3) |
| | » 20 » 40 » | 235(24) | 370(38) | 24 | $d = 2a$ | 29(3) | — | — | 29(3) |
| С275 | От 4 до 10 включ. | 275(28) | 390(40) | 24 | $d = a$ | — | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 380(39) | 23 | $d = a$ | — | — | — | 29(3) |
| С285 | От 4 до 10 включ. | 285(29) | 400(41) | 24 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 390(40) | 23 | $d = a$ | 29(3) | — | — | 29(3) |

Рисунок 3.6. Результат работы `pytesseract.image_to_boxes`

Далее, зная координаты всех линий образующих таблицу, разделим их на две группы - вертикальные и горизонтальные. Затем найдем точки, в окрестности которых эти линии пересекаются. Отдельно обработаем граничные точки таблицы, так как в них пересечений нет.

В случае, когда в таблице разграничивающие ячейки линии, например, двойные, точки могут находится некорректно. Поэтому полезно будет провести обработку координат с их объединением, в случае если ячейки лежат в окрестности друг друга.

**Механические свойства фасонного проката**

| Наименование стали | Толщина полки, мм | Механические характеристики: Предел текучести $\sigma_т$, Н/мм² (кгс/мм²) | Временное сопротивление $\sigma_в$, Н/мм² (кгс/мм²) | Относительное удлинение $\delta_5$, % | Изгиб до параллельности сторон ($a$ — толщина образца, $d$ — диаметр оправки) | Ударная вязкость KCU, Дж/см² (кгс · м/см²): при температуре, °С: −20 | −40 | −70 | после механического старения |
|---|---|---|---|---|---|---|---|---|---|
| | | не менее | | | | не менее | | | |
| С235 | От 4 до 20 включ. | 235(24) | 360(37) | 26 | $d = a$ | — | — | — | — |
| | Св. 20 » 40 » | 225(23) | 360(37) | 25 | $d = 2a$ | — | — | — | — |
| С245 | От 4 до 20 включ. | 245(25) | 370(38) | 25 | $d = a$ | — | — | — | 29(3)* |
| | Св. 20 » 25 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | 29(3) |
| | » 25 » 30 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | — |
| С255 | От 4 до 10 включ. | 255(26) | 380(39) | 25 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 245(25) | 370(38) | 25 | $d = a$ | 29(3) | — | — | 29(3) |
| | » 20 » 40 » | 235(24) | 370(38) | 24 | $d = 2a$ | 29(3) | — | — | 29(3) |
| С275 | От 4 до 10 включ. | 275(28) | 390(40) | 24 | $d = a$ | — | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 380(39) | 23 | $d = a$ | — | — | — | 29(3) |
| С285 | От 4 до 10 включ. | 285(29) | 400(41) | 24 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 390(40) | 23 | $d = a$ | 29(3) | — | — | 29(3) |

Рисунок 3.7. Результат нахождения узлов таблицы

Теперь по узлам таблицы можно найти координаты прямоугольника, совпадающего с ячейкой.

**Механические свойства фасонного проката**

| Наименование стали | Толщина полки, мм | Механические характеристики: Предел текучести $\sigma_т$, Н/мм² (кгс/мм²) | Временное сопротивление $\sigma_в$, Н/мм² (кгс/мм²) | Относительное удлинение $\delta_5$, % | Изгиб до параллельности сторон ($a$ — толщина образца, $d$ — диаметр оправки) | Ударная вязкость KCU, Дж/см² (кгс · м/см²): при температуре, °С: −20 | −40 | −70 | после механического старения |
|---|---|---|---|---|---|---|---|---|---|
| | | не менее | | | | не менее | | | |
| С235 | От 4 до 20 включ. | 235(24) | 360(37) | 26 | $d = a$ | — | — | — | — |
| | Св. 20 » 40 » | 225(23) | 360(37) | 25 | $d = 2a$ | — | — | — | — |
| С245 | От 4 до 20 включ. | 245(25) | 370(38) | 25 | $d = a$ | — | — | — | 29(3)* |
| | Св. 20 » 25 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | 29(3) |
| | » 25 » 30 » | 235(24) | 370(38) | 24 | $d = 2a$ | — | — | — | — |
| С255 | От 4 до 10 включ. | 255(26) | 380(39) | 25 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 245(25) | 370(38) | 25 | $d = a$ | 29(3) | — | — | 29(3) |
| | » 20 » 40 » | 235(24) | 370(38) | 24 | $d = 2a$ | 29(3) | — | — | 29(3) |
| С275 | От 4 до 10 включ. | 275(28) | 390(40) | 24 | $d = a$ | — | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 380(39) | 23 | $d = a$ | — | — | — | 29(3) |
| С285 | От 4 до 10 включ. | 285(29) | 400(41) | 24 | $d = a$ | 29(3)* | — | — | 29(3)* |
| | Св. 10 » 20 » | 275(28) | 390(40) | 23 | $d = a$ | 29(3) | — | — | 29(3) |

Рисунок 3.8. Результат нахождения ячеек таблицы

Получившиеся координаты удобно отсортировать по координатам верхней левой границы прямоугольников, учитывая тот факт, что начало координат находится в нижнем левом углу, а таблицу удобнее читать и заполнять с верхнего левого. Теперь по координатам этих прямоугольников можно проводить обрезку изображения таблицы так, что получится новое изображение, на котором лишь одна ячейка. Ее можно подать на вход методу `pytesseract.image_to_string` и получить строковое представление содержимого ячейки. На данный момент реализовано сохранение этих строковых значений в двумерный массив, однако при таком подходе частично теряется структура таблицы.

```
1 for i, rectangle in enumerate(rectangles):
```

```
    x1, y1 = rectangle[0]
    x2, y2 = rectangle[1]
    cell_image = image[max(0, (height - y2) - 2):min(height, (height - y1) +
        2), max(0, x1 - 2):min(width, x2 + 2)]

    custom_config = r'--oem 1 --psm 1 -l rus'
    if cell_image.shape[0] > 10 and cell_image.shape[1] > 10:
        gray_image = cv2.cvtColor(cell_image, cv2.COLOR_BGR2GRAY)
        text = pytesseract.image_to_string(gray_image,
            config=custom_config).replace('\n', ' ')
        row = i // num_columns
        column = i % num_columns
        if row < num_rows and column < num_columns:
            table_data[row][column] = text
```

# 4. Результаты

В результате выполнения алгоритма создается файл с расширением csv, в котором хранятся данные из таблиц, встречающихся в документе. Итоговый результат сильно зависит от качества входного файла, от структуры таблицы (наличие линий разметки и их замкнутость подразумеваются при обработке) и от того, какие данные хранятся в ячейках (степени и индексы и, иногда, числа не распознаются) Так, например, при корректном распознавании координат ячеек, выходной файл для рассматриваемой выше таблицы имеет пропуски и неправильно распознанные ячейки.

| Толщина полки, мм | Механические характеристики | Изгиб до параллель- ности [сторон (а— толщина образца, 4—диаметр ити | Предел "текучести \|. Н/мм? (кгс/мм?) | Временное сопротив- ление д, `НИмме (кгс/мм2) | Относи - тельное удлине - ние 5, ® | при температуре, °С | ‘после механи - ческого старе- ния | —20 | |
|---|---|---|---|---|---|---|---|---|---|
| —0 | | не менее | | | | не ме | нее | | C23 5 |
| От 4до 20 включ. Св.20 » 40 › | 235(24) 225(23) | 360(37) 360(37) | 26 25 | | | | | | C24 5 |
| От 4до 20 включ. Св.20 » 25 ь » 25»30 » | 245(25) 235(24) 235(24) | 370(38) 370638) 370638) | 25 24 | ъьь нии в | | | | 29(3)* 29(3) | C25 5 |
| От 4до 10 включ. Св.10 > 20 » » 20540 › | 255(26) 245425) 235(24) | 380(39) 370638) 370(38) | 25 24 | эза ии эз= | 29(3)* 2963) 2963) | | | 29(3)* 2903) 29(3) | C27 5 |
| От 4до 10 включ. Св.10» 20 » | 275(28) 275(28) | 390(40) 380(39) | 24 23 | | | | | 29(3)* 29(3) | C28 5 |
| От 4до 10 включ. Св.10» 20 › | 285029) 27508) | 400441) 390(40) | 24 23 | | 29(3)* 29(3) | | | 29(3)* 29(3) | |

Рисунок 4.1. Общий результат алгоритма

# 5. Заключение

В рамках данного отчета была рассмотрена задача извлечения данных из таблиц в отсканированных или цифровых документов. Были реализованы различные этапы алгоритма обработки изображений, включая предобработку, сегментацию, распознавание текста и структурирование данных таблицы.

В результате экспериментов было установлено, что использование методов компьютерного зрения, таких как бинаризация, сглаживание и детектирование контуров, позволяет эффективно выделить таблицу на изображении. Далее, применение библиотеки распознавания текста, такой как Tesseract, позволяет извлечь текстовую информацию из ячеек таблицы.

Стоит отметить, что качество распознавания может сильно зависеть от различных факторов, включая качество изображения, размер и шрифт текста, тип таблицы, наличие шумов и искажений. Для улучшения качества распознавания были применены методы предобработки, такие как улучшение контраста, фильтрация шума и увеличение резкости, некоторые из них имели положительный результат и вошли в итоговый алгоритм.

Актуальными остаются вопросы конечного структурирования данных, улучшения качества распознавания текста и, в особенности цифр. Части используемых сейчас инструментов может быть заменена более специализированными под эту задачу.

# Список литературы


[1] Оффициальный сайт OpenCV - https://opencv.org/

[2] An Overview of the Tesseract OCR Engine - https://research.google.com/pubs/archive/33418.pdf

[3] How to OCR with Tesseract, OpenCV and Python - https://nanonets.com/blog/ocr-with-tesseract/

[4] OpenCV шаг за шагом. Обработка изображения — детектор границ Кенни - https://robocraft.ru/computervision/484

[5] Suzuki's Contour tracing algorithm OpenCV-Python - https://theailearner.com/2019/11/19/suzukis-contour-tracing-algorithm-opencv-python/