Евразийский национальный университет им. Л.Н. Гумилёва

Факультет информационных технологий

Кафедра “Информационные системы”

**Отчет по учебной практике по теме**

**“Компьютерное зрение”**

**Подготовил:** Гасанов Агит, Кан Николай

**Группа:** ИС-12

**Проверил:** профессор .

**Астана 2023**

**Тема:** Распознавание текста на изображениях.

**Цель работы:** Проектирование и разработка Python-модуля для распознавания текста на изображениях и конвертирование его в текстовый файл, с возможностью включения в системы сканирования и обработки рукописного текста, надписей, документов и др.

Данная цель обусловлена потребностью в автоматическом переводе в символьный формат текста, данного на изображении. Проблема сложности перевода текста из изображения в текстовые документы актуальна для бытового применения для быстрого считывания не редактируемых текстов на изображениях.

В ходе работы были выработаны требования к разрабатываемому модулю, спроектирована архитектура и разработана первая рабочая версия модуля. Определено направление дальнейшего развития модуля.

**Определения**

В данной работе используется следующие термины:

- Библиотека: стороннее расширение для программной среды, подключаемая отдельно, с возможностями импорта функций методов и значений.

- Метод: Функция, принадлежащая какому-либо классу или объекту в ООП.

- OCR: Optical Characters Recognition – оптическое распознавание символов, раздел программной инженерии, который специализируется на распознавании символов на изображениях или видео.

- Виджет: небольшое графическое приложение, которое выводит информацию на рабочий стол компьютера.

- Нейросеть: Тип машинного обучение, когда на основе множества примеров выстраивается цепь ассоциаций, которая позволяет принимать компьютеру верные решения, подобно нейронам человеческого мозга.

**Введение**

В современном мире большую роль в автоматизации играет возможность машин понимать реальный мир без участия человека. Поиск дефектов, биометрическая аутентификация, поиск по изображению, машинное зрение, считывание текста, автомобильный автопилот полаются на то, что программное обеспечение способно обрабатывать информацию в виде изображений подобно человеческому зрению и лучше. Использование современных вычислительных возможностей и достижениями в области машинного обучения позволяет создавать программы машинное зрения со впечатляющими возможностями для практического использования.

В нашем случае было разработано python приложение, для считывания и перевода текста с экрана компьютера и возможностями копирования, редактирования и сохранения в текстовый файл полученного результата.

Данное приложение, благодаря использованию готовой библиотеки с шаблонами символов на разных языках легко модифицируется под нужды пользователя.

История оптического распознавания символов уходит в конец тридцатых годов прошлого века, когда немец Густав Таушек первым зарегистрировал патент на технологию оптического распознавания текста. Спустя двадцать лет, американский криптоаналитик Дэвид Шепард создал машину, которая решала подобные задачи. Впоследствии Шепард стал основателем компании, которая первая в мире выпустила собственные системы OCR. Одним из первых покупателей систем был журнал Readerʼs Digest, ставший одним из самых активных её пользователей.

Следующий виток развития OCR случился в 1974 году с созданием компании «Kurzweil Computer Products». Рэй Курцвейль, основатель, сделал акцент на развитие системы Optical Character Recognition, способной распознавать текст независимо от шрифта. Задумка заключалась в создании особой машины для незрячих людей, которая распознавала бы тексты и читала их вслух. Итоговый продукт был представлен в 1976 году, а в 1978 Курцвейль продал компанию Xerox Corporation.

В 1993 году на рынок вышел программный продукт для распознавания текста от ABBYY, получивший широкое распространение. В частности, на её основе была создана ABBYY FineReader. Технологию лицензируют крупнейшие мировые компании, среди которых EMC, Samsung и др.

В современном мире также используются алгоритмы распознавания на основе нейросетей, которые весьма ресурсозатратны, но позволяют выводить текст с очень высокой точностью.

**Основная часть**

1. **Литературный обзор**
   1. **Оптическое распознавание символов.**

Человек получает от 70 до 90 процентов информации через зрение и крайне важно научить компьютер подобным способностям, для дальнейшей автоматизации процессов. Разработчики долгое время работали в этой сфере и сейчас машинное зрение не является чем-то фантастичным, а скорее это обыденность, хоть и ей есть куда развиваться. Оптическое распознавание символов также не новая технология, информация о которой содержит готовые и отработанные методы и алгоритмы обнаружения, идентификации и сегментации символов.

Для быстрой цифровизации текста с изображения существует программы по автоматическому распознаванию символов. В зависимости от метода обработки и стиля текста точность может составлять до 99% (четко различимый печатный текст).

Оптическое распознавание символов – это механический или электронный перевод изображений рукописного, машинописного или печатного текста в текстовые данные, Оптическое распознавание символов – механический или электронный перевод изображений рукописного, машинописного или печатного текста в текстовые данные, использующиеся для представления символов в компьютере. Обычно данная технология применяется для сканирования печатного текста или при опознавании иных надписей, кодов и идентификационных номеров. Все это может структурироваться в формат электронного текстового документа или в случае идентификаторов сверяется с базой данных. Это значительно экономит время при копировании текста в электронный формат для его последующей обработки.

Базовый алгоритм представляет из себя:

1. Предобработку изображения для повышения видимости контрастных элементов.
2. Обнаружение символов на изображении, выделение данных областей.
3. Сегментация по буквам, словам и возможно абзацам.
4. Классификация символов.
5. Обобщение полученных символов в текст.
   1. **Обзор методов ОРС.**

Существуют различные алгоритмы для ОРС, они различаются по точности, универсальности, скорости работы и объему необходимой памяти.

Распознавание при помощи метрик:

Данный метод лучше всего работает с машинописным текстом, но при обработке новых шрифтов точность распознавания падает. Метрика является признаком символа, поэтому иногда в контексте данного способа говорят о процессе выявления признаков. В качестве метрики используют расстояние Хэмминга, которое показывает, на сколько пикселей различаются изображения. Если признаки двух символов максимально похожи, то разность между их метриками (то есть расстояние между ними) стремится к нулю. Дальнейшая классификация символа происходит по методу ближайшего соседа. Однако, одной метрики недостаточно для распознавания символа, так как некоторые очень похожи между собой, (например, “j” и “i”, “Z” и “2”) что может привести к ошибке. Для избегания этого, используют следующие техники:

1) Группировка символов

Некоторые символы (“O”, “H”, “I”) обладают суперсимметрией, (полностью совпадают со своими отражениями, значимые пиксели распределены равномерно по всему изображению) и их можно выделить в отдельный класс. Это значительно сокращает перебор метрик.

2) Контекстное распознавание

В качестве помощи алгоритмам распознавания в систему включают словари. Они предоставляют справки во многих случаях, но быстро отказывают, когда, например, имеют дело с именами собственными, которые не находятся в словаре.

Распознавание с применением нейронных сетей:

Нейронные сети – это структура связанных элементов, на которых заданы функции преобразования сигнала, а также коэффициенты, которые могут быть настроены на определенный характер работы.

Сигнал, проходящий через нейронную сеть, преобразуется согласно формулам на элементах сети, на выходе формируется ответ. Так как все нейроны поименованы значениями букв, следовательно, среагировавший нейрон и несет ответ распознавания.

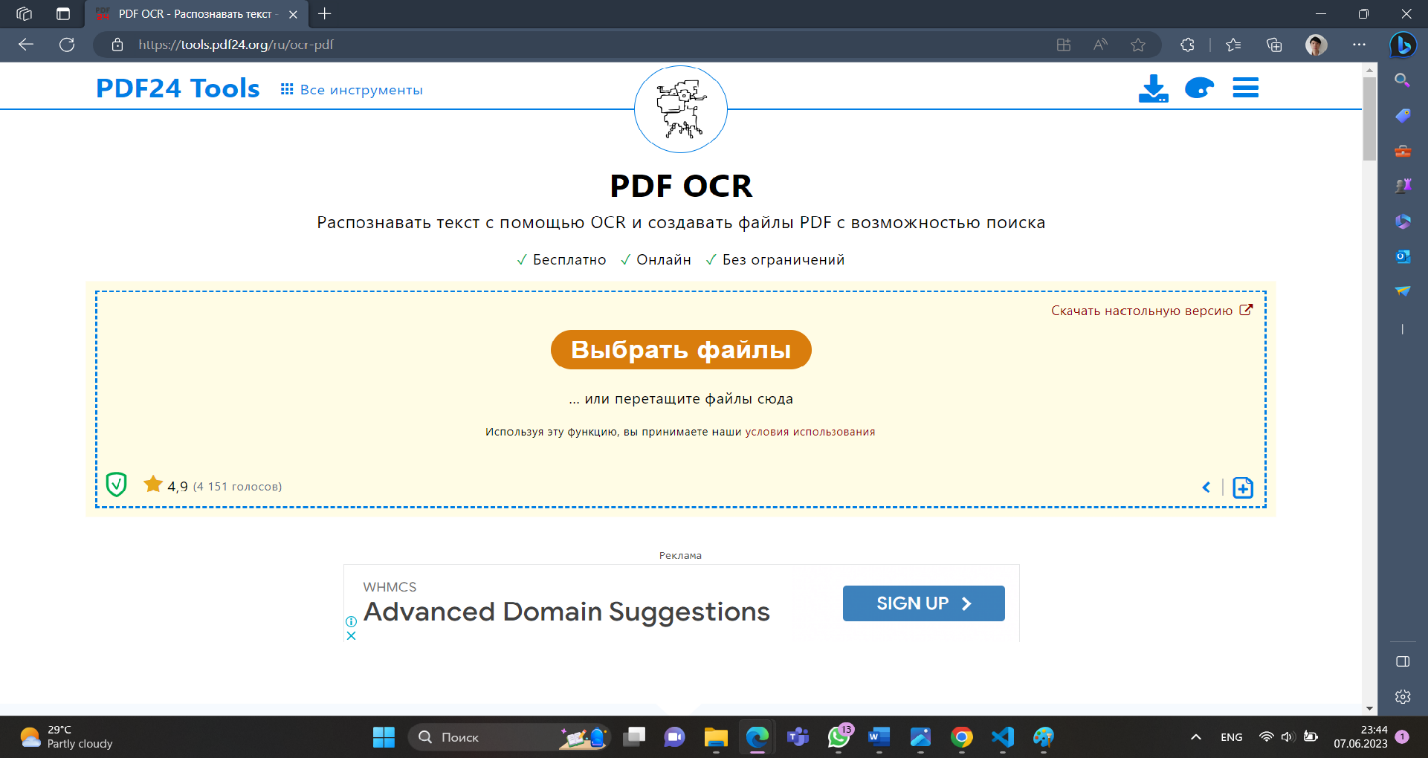
Нейронная сеть может быть использована в системе распознавания текста в качестве классификатора. При обучении, сеть получает на вход изображения, анализирует все позиции черных пикселей и выравнивает коэффициенты, минимизируя ошибку. Таким образом, достигается лучший результат распознавания.

Для обработки текста используется библиотека EasyOCR для python которая использует метод нейронных сетей типа ResNet, LSTM, CTC и выдает впечатляющую точность распознавания символов даже при наличии помех.

**1.3 Обзор аналогов**

Существует множество программ по распознанию текста, как встроенные, так и отдельные приложения и сайты.

Типичный ОРС сайт вроде *tools.pdf24.org* рассчитан на распознавание печатного текста, он бесплатен и без ограничений, только рассчитан на онлайн использование. Интерфейс предлагает загрузку файла и последующее сохранение в формате редактируемого текста. Аналогичные сайты могут требовать регистрации или взимать плату за использование ресурса



*Рисунок 1 – сканер изображений tools.pdf24*

Также существуют решения, встроенные в переводчики, например мобильное приложение Google translate предоставляет возможность сканирования и перевода в реальной времени или сканирования и перевода импортируемых изображений. Приложение Screen Translate позволяет сканировать экран смартфона и предлагает перевод.



*Рисунок 2 – сканер-переводчик в реальном времени от google*

Настольным аналогом на экранного распознавания текста является наш модуль.

1. **Техническое задание**
   1. **Основные задачи и цели**

Целью данной работы является разработка приложения для считывания текста с экрана компьютера и выводящее результат в текстовом окне для возможности его копирования.

При активации виджета программа должна выполнить следующие задачи:

1. Сделать снимок текущего экрана.
2. Предоставить возможность выбрать область экрана для обработки.
3. Определить символы, строчки, провести сегментацию.
4. Сохранить полученный текст в формате строки.
5. Перевести текст на другой язык и вывести его в новое окно

Для определения сроков проекта, постановки целей и задач использовался программа Jira. Данная программа предоставляет удобную визуализацию «Дорожной карты» проекта, интуитивный интерфейс позволяет с легкостью добавлять задачи, назначать сроки и исполнителей. Jira является самым популярным решением для команд из нескольких человек и во многих отраслях требуется знание данной программы.

Для наших задач было выбрано шесть главных целей:

1. Цели и задачи проекта
2. Изучение компьютерного зрение
3. Написание кода
4. Подготовка отчета
5. Представление проекта
6. Выступление

Были поставлены сроки до 10 июня, последнего дня приема.

При создании схем была использована программа draw.io – многофункциональная программа для построения схем, удобная для создания блоков алгоритмов, и UML схем

**Требования к модулю**

Перед проектированием приложения необходимо сформировать все предъявляемые требования:

1. Приложение должно иметь удобный и интуитивный интерфейс.
2. Точность распознавания символов по крайней мере печатного текста должна быть очень высокой
3. Приложение должно обрабатывать запросы на высокой скорости.
4. Исходный код разработанного приложения должен быть открытым и легким для понимания, для последующей возможной модификации.
5. Приложение должно легко настраиваться на различные под различные мировые языки.
6. Соблюдение относительной простоты и надежности, пресечение багов и ошибок при любых возможных непредвиденных обстоятельствах.
7. Универсальность и высокая практическая ценность.

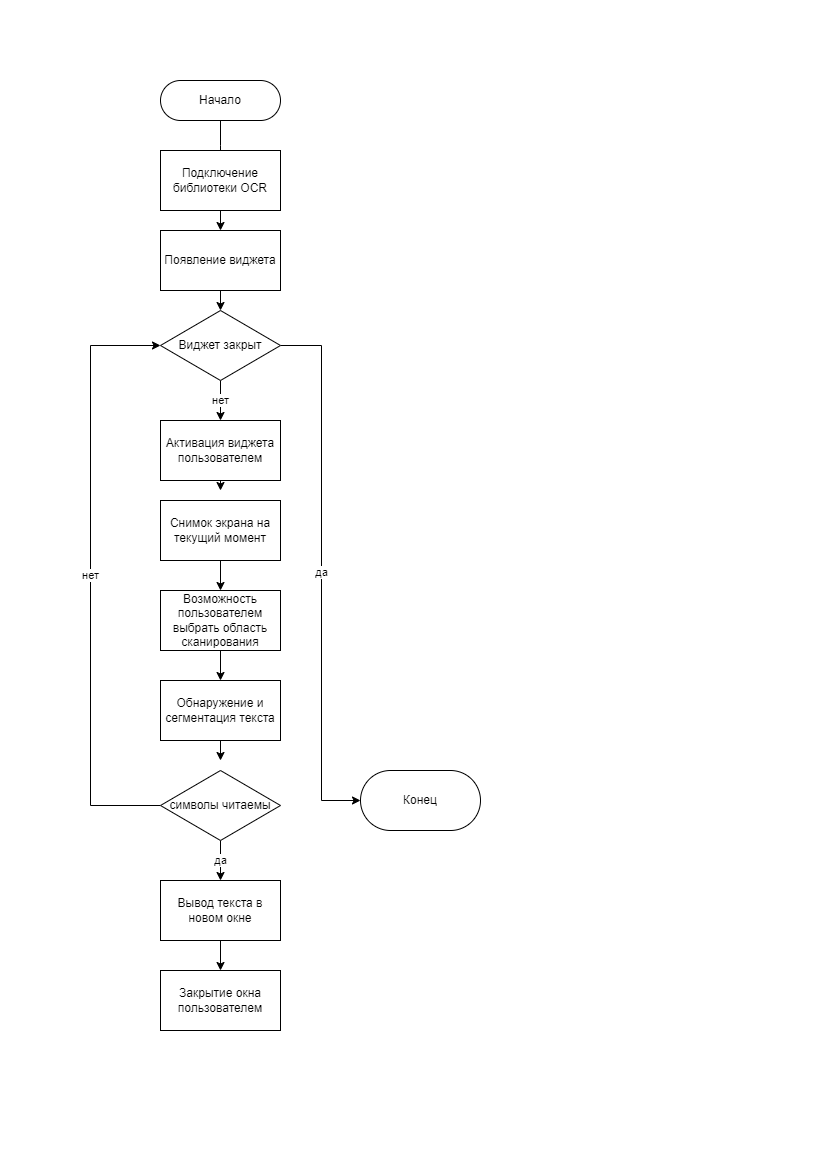
На рисунке 1 с помощью блок-схем представлена работа приложения.

При запуске приложения импортируются все необходимые библиотеки: easyOCR для считывания текста, Tkinter для создания интерфейсов, ImageTk, pyautogui для работы с изображениями. Приложение должно встречать пользователя плавающим небольшим виджетом, который будет отображаться поверх всех окон. На виджете будет располагаться одна кнопка с возможностью размещения еще нескольких кнопок настройки. Выход из приложения осуществляется закрытием виджета, закрытие последующих окон будет возвращать виджет на экран. Все это будет реализовано с помощью библиотеки tkinter

При нажатии на кнопку «сканировать экран» приложение будет ждать 20 миллисекунд для того, чтобы виджет не попал на скриншот и успел закрыться, делает снимок экрана при помощи библиотеки pyautogui и предоставляет пользователю возможность выбрать прямоугольную область экрана для сканирования по средством зажатия и перетаскивания курсора. Координаты выбранной области считываются. За работу с изображением отвечает библиотека ImageTK.

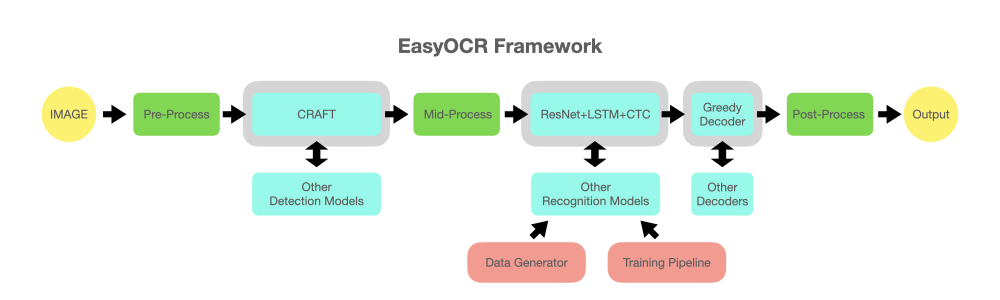
Вырезанная область экрана обрабатывается методом readtext из библиотеки easyOCR, корнем всей программы, который определят, сегментирует и идентифицирует символы, после чего выводит их в виде вектора.

Последующий вектор с текстом должен быть соединен в одну строку и передан переводчику, а после может выводится в новом окне, где он может быть отредактирован или скопирован. При закрытии данного окна будет возможность вернуться к виджету.

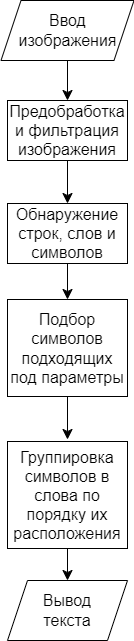


*Рисунок 3 – схема работы приложения*

Если описывать работу алгоритмов EasyOCR (Рисунок 3, 4), то сперва задается язык текста под сканирование, использование двух языков возможно, но это задействует дополнительную память и возможно будет сопровождаться ошибками. Полученное изображение сперва фильтруется, увеличивается контрастность, накладываются фильтры шумоподавления. После изображение подвергается первичному алгоритму обнаружения символов, в данном случае применяется алгоритм CRAFT. Он относительно быстро очерчивает границы предполагаемых символов. Предполагаемые символы проходят идентификацию через нейросеть, которая была обучена определять символы данного языка. Идентифицированные символы группируются и объединяются в слова и словосочетания, если программа решит сохранить пробел.

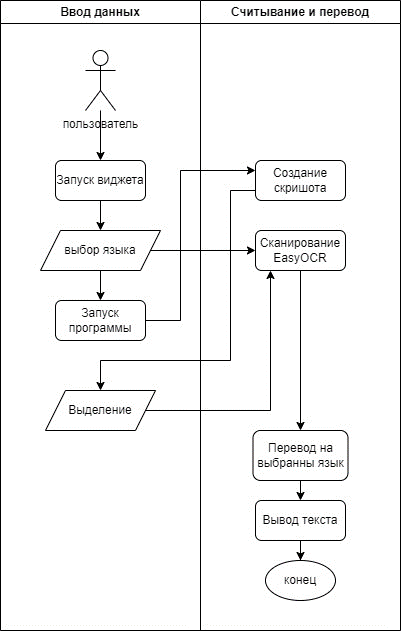


*Рисунок 4 – блок-схема работы алгоритма опознавания.*



*Рисунок 5 – блок-схема интерпретации работы алгоритма опознавания.*

Была создана UML-схема, для интуитивного понимания взаимодействия пользователя с программой. Через интерфейс пользователь выбирает языки сканирования и перевода, и запускает виджет. Виджет в свою очередь активирует снимок экрана на стороне программы. Программа ожидает от пользователя выделение области, и когда область выделена, работа переходит на сканирование текста. EasyOCR ищет символы выбранного языка, а после выводит их в виде текста, который идет на обработку в переводчик. Переводчик выдает результат пользователю в новое окно.



*Рисунок 6 – UML схема работы приложения с пользователем.*

**4 Программная реализация модуля**

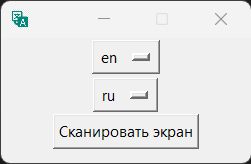
**4.1 Язык программирования**

В качестве инструмента для разработки был выбран язык python. Данный первый в списке когда речь заходит о нейросетях, машинном обучении и в том числе компьютерном зрении. При подключении определенных библиотек можно использовать его для создания приложений, что мы и сделали. Благодоря наличию множества библиотек, которые сильно расширяют возможности стандартного python, этот язык является очень хорошей базой для нашего приложения и позволяют быстро писать компактный и понятный код.

Используемая версия языка – 3.10

**4.2 На экранный виджет**

При создании простого и функционального интерфейса, используем возможности библиотеки для python – Tkinter. Создание окон, полей, текстовых подсказок и так далее. Окно для виджета должно быть небольшим и всегда находится на переднем плане, содержать кнопку для запуска сканера и после запуска закрываться чтобы не попадать на снимок экрана. Для постоянного отображения поверх всего в Tkinter существует параметр для окна wm\_attributes("-topmost", True). Размер виджета выбран через параметр geometry('250x50'), добавление кнопки через функцию tk.Button, которая создает кнопку и исполняет функцию закрытия виджета, выжидания анимации закрытия и старта сканирования. Для любого окна в tkinter в конце стоит добавлять root.mainloop(), данный метод останавливает действие программы, до закрытия окна. Также виджет должен содержать возможности выбора языка для сканирования и перевода, в виде выпадающего меню.



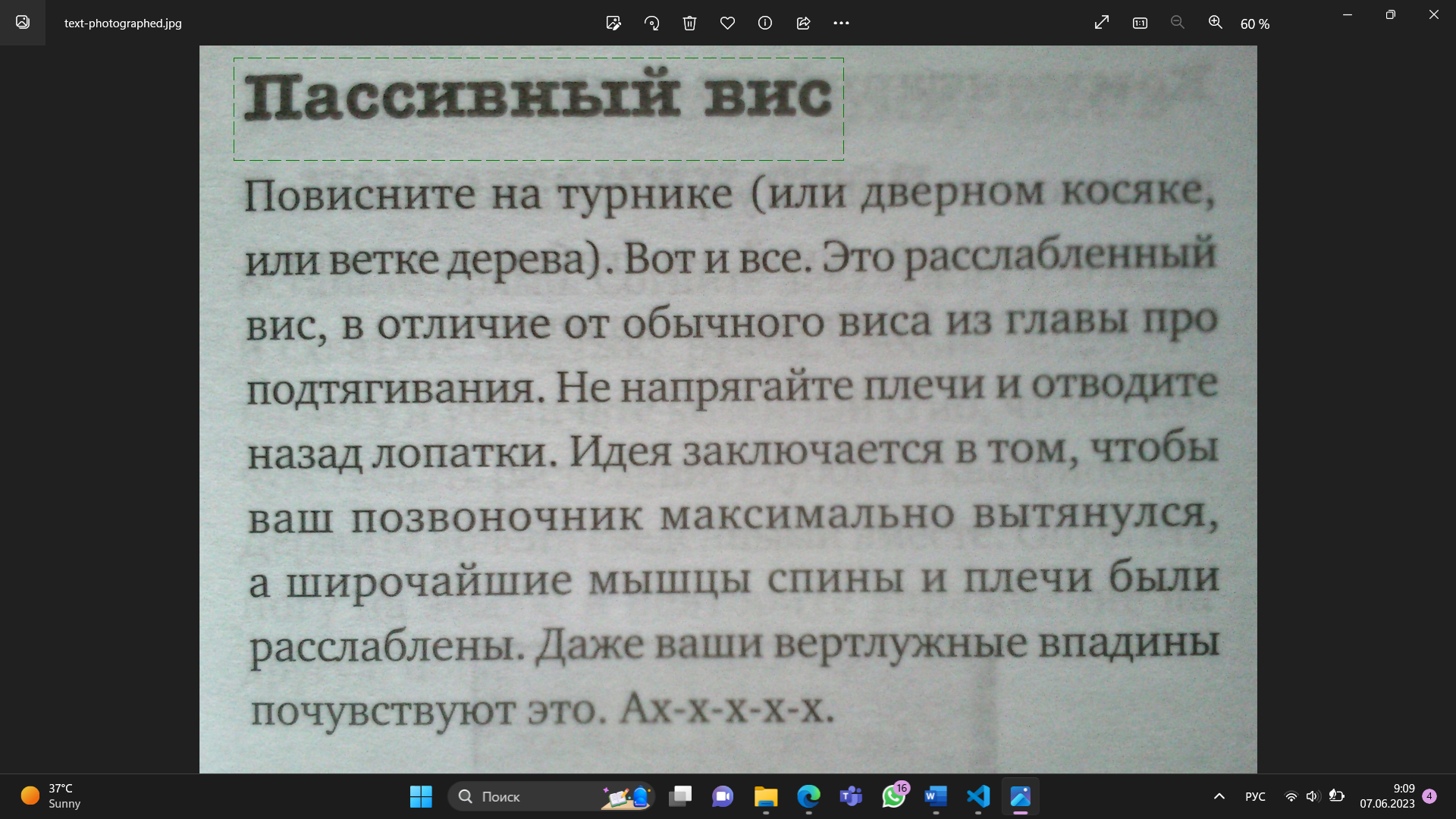
*Рисунок 7 – интерфейс на экранного виджета*

**4.3 Выделение текста**

Для выделения текста были использованы библиотеки Tkinter, pyatongui, PIL. Так после закрытия виджета делается снимок экрана, и открывается полноэкранная копия изображения с возможностью выделения текстого блока. Для создании функции выделения мы спера создаем две вспомогательные функции, которые вычисляют координаты вехней и нижней точки выделения и привязываем их событию зажатия и отпускания ЛКМ: topx, topy = event.x, event.y; botx, boty = event.x, event.y

Для визуализации выделяемой области создаем прямоугольник зеленого цвета с помощью метода из Tkinter: create\_rectangle(topx, topy, topx, topy, dash=(10,10), fill='', outline='green'),

Полученные координаты поступают для обрезки изображения экрана. Выделенная часть экрана передается для обработки EsayOCR.



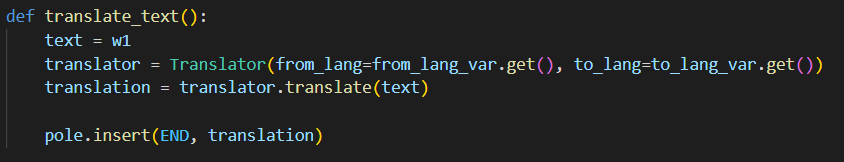
*Рисунок 8 – интерфейс выделения части экрана*

**4.4 Обаботка изображения**

Данная библиотека, а конкретно метод readtext, который считывает текст с помощью уже обученных нейросетей, стали корневыми функциями при считываниии текста с изображений. Метод выводит найденные слова или символы, а также инормацию о расположении текста, и точностью опознования. Также он имеет множество параметров, например вывод только текста, или уменьшение точности в угоду скорости работы или сохранению памяти, переключение между обработкой процессором или видеокартой. Имеется возможность выбрать язык текста, или даже несколько языков. В нашем случае нам понадобиться выводить только текст на английском и русском. Метод readtext принимает на вход путь до изображения, и соотвественно указываем переменную содержащюю выделенную часть экрана и подчеркиваем необходимость выводить только текст: reader.readtext('delete.jpg', detail=0), перед этим указав языки для считывания easyocr.Reader(["en","ru"]). После манипуляций с изображением его следует удалить: os.remove('delete.jpg'). Выведенный текст в последующем сохраняется в новой переменной в виде вектора, но для правильного вывода текста с пробелами выполняем метод insert w = ' '.join(result)

**4.5 Перевод текста**

Для перевода текста была дополнительно использована библиотека translate и содержащаяся в ней функция Translator. Данная библиотека хоть и не идеально, но хорошо справляется с переводом множества языков. К сожалению не все языки поддерживаются в библиотеке, в том числе и казахский язык. Для перевода создается простая функция принимающая на вход отсканированный текст, а также значения изначального и переводимого языка. В последующем функция выведет результат в поле текста. Данная функция исполняется только уже после создания окна для предпологаемого текста и должна стоять в конце.

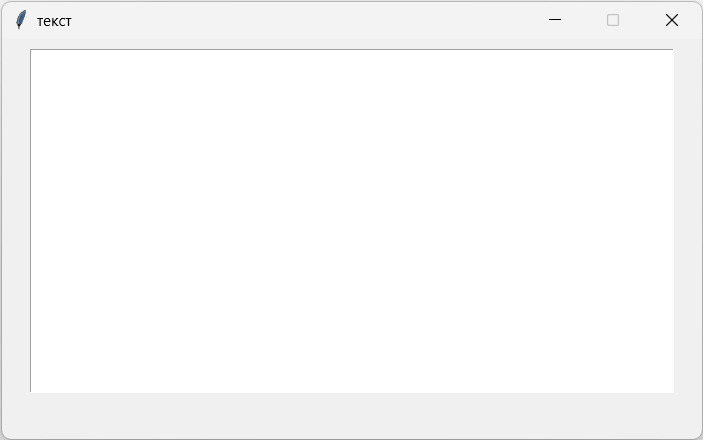


*Рисунок 9 – код функции перевода*

**4.6 Вывод текста**

Осталось лишь вывести текст, для этого создаем еще одно окно с помощью Tk(). В данном случае нам понадобится поле ввода: pole = Text(root, width=80, height=20, font='Arial, 9'), где мы указываем размер в символах и строках, а не в пикселях. В данное поле по средством метода insert('1.0', w) мы вставляем текст полученный в результате опознования символов. В данное окно посредством функции описанной выше вставляется переведенный текст.

Получаем: окно:



*Рисунок 10 – окно вывода текста*

**4.7 Цикличность приложения**

Дополнительно необходимо зациклить окна, чтобы при закрытии окна прогрмма переходила обратно виджет. Для запоминания настроек перевода (выбранного языка) созданны переменные хранащие эти данные и обновляющиеся под конец цикла, и первоночально эти переменные установлены в занчении en >> ru. Дополнительная переменная escape при нулевом значении запускает цикл, при значении единица, которое присваивается на каждом заходе в цикл, и при закрытии окна приложение не запускает щикл. После закрытия окна вновь открывается виджет, который заполнил последий исплользованный язык и готов к повторному использованию, при этом цикл не должен допускать утечек памяти и например изображение должно всегда удалятся в конце программы.

**5. Тестирование приложения**

**5.1 Поставленные тесты**

Проверка отработки приложения проводидся поэтапно, на каждом шаге алгоритма могут возникнуть ошибки и стоит проверить каждую непредвиденную ситуацию, для того чтобы с ней не столкнулся пользователь. В ходе тестирование проверялись:

1. Виджет: появление виджета, его правильное отображение, правильная отработка кнопок и закрытие виджета, возможность его перетаскивать не попадание виджета на скриншот.
2. Открытие полноэкранного режима, правильное отображение снимка экрана, правильное выделение и отправка координат для обрезания изображения.
3. Правильный вывод текста, удобный размер текста и окна, открытие и закрыти окна с текстом.
4. Правильное отображение текста, проверка качества опознования текста, определение предельных возможностей обхода помех.
5. Правильное зацикливание прилжения, возможность покинуть приложение только через виджет.
6. Скорость работы приложения, проверка на утечку памяти.
7. Правильность перевода на различные языки

**5.2 Возникшие ошибки и исправления**

Во время проектирования приложения естественно возникало множество трудностей с которыми иы успешно справились. Основные сложности вызвали:

1. Во время проектирования видежета необходимо было его отображение на переднем фоне, для его удобного использования. Необходимо было изучить атрибуты окон в библиотеке Tkinter, где необходимый параметр был найден и изменен на Topmost
2. Возможность выхода из приложения только из виджета, и зацикливание при повторном использовании. Данная проблема была основным препятсвием для повторного использования приложения. Для решения этой проблеммы был создан цикл открывания окон, который обравался при выходе из виджета
3. Решением для интуитивного выделения части экрана было сделать снимок экрана и открыть его в полном окне, благо необходимые отрибуты для окон в Tkinter существуют.
4. Картинка открывалась немного съехавшей по диагонали, и необходимо было передвинуть ее по координатом вверх и влево.
5. При выделение по координатам от большего к меньшему TKimage выдвала ошибку, для правильной обрезки изображения была создана простое проверочное условие, которое в случае ошибки меняло местами координаты.
6. Переводчик работал некоректно, были сильные ошибки в переводе, необходимо было установить обновленнную версию библиотеки.
7. Для слияния в одну группу векттора текста выдаваемаого Readtext была использована команда str(' '.join(result)).
8. Для эстетического вида была добавлена иконка переводчика в левый верхний угол через атрибут: root.iconphoto(False, mainphoto)

Все вышеописанные проблемы были решены в ходе тестирования и отладки проекта.

**6. результаты**

**6.1 Итоги проекта**

В итоге проекта

**6.2 Выводы**

Данный проект являлся правктической проверкой навыков полученных во время обучения на первом курсе факультета «информационные технологии», а также полученных непосредственно во время проведения исследовательских работ. Были получены и применены множество полезных навыков по организации труда, распределения опязаннастей, постановки целей в команде, по созданию самодостаточных программ и иследованию их практической пользы. Была выработана модель организации проектов, изучен вспомогательный софт: jira, Github. Draw.io и другие. Опыт полученный при проведении учебной практики может быть использован в дальнейшем при разработке новвых проектов, уже более серьезных проектов.

6.2 выводы

заключение