

Автоматизация обработки изображений с использованием Python и библиотеки Pillow: разработка приложения для массовой обработки изображений

Акентьев Данила Денисович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье представлена разработка программного приложения для автоматизации обработки изображений с использованием языка программирования Python и библиотеки Pillow. Основные функции программы включают изменение размера, поворот, добавление водяных знаков, настройку яркости, контраста, резкости, а также применение фильтров (размытие, инверсия цветов). Рассматриваются основные этапы разработки, включая создание пользовательского интерфейса с использованием библиотеки Tkinter, обработку изображений и алгоритмы для выполнения различных манипуляций с графическими файлами. Представленные методы позволяют значительно ускорить и упростить процесс обработки изображений, что полезно для графических дизайнеров, фотографов и других специалистов в области визуальных искусств.

Ключевые слова: Автоматизация обработки изображений, Python, Pillow, Tkinter, Водяные знаки, Манипуляции с изображениями, Графическая обработка, Массовая обработка изображений, Интерфейс пользователя, Фильтры изображений.

Automating image processing using Python and the Pillow library: Developing an application for mass image processing

Akentev Danila Denisovich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article presents the development of a software application for automating image processing using the Python programming language and the Pillow library. The main functions of the program include resizing, rotating, adding watermarks, adjusting brightness, contrast, sharpness, as well as applying filters (blurring, color inversion). The main stages of development are considered, including the creation of a user interface using the Tkinter library, image processing and algorithms for performing various manipulations with graphic files. The presented methods can significantly speed up and simplify the image processing process, which is useful for graphic designers, photographers and other specialists in the field of visual arts.

Keywords: Automation of image processing, Python, Pillow, Tkinter, Watermarks, Image Manipulation, Graphic Processing, Mass Image Processing, User Interface, Image Filters.

1 Введение

1.1 Актуальность

Современные технологии обработки изображений играют важную роль в различных сферах, таких как цифровая фотография, графический дизайн, мультимедиа и интернет-маркетинг. В условиях большого потока изображений, особенно в социальных сетях и на веб-сайтах, необходимо обеспечить эффективные и автоматизированные методы обработки. Программное обеспечение для массовой обработки изображений, которое сочетает простоту использования и гибкость, является актуальным инструментом для специалистов, работающих с изображениями. Использование Python и библиотеки Pillow позволяет создать мощное решение для автоматизации различных операций с изображениями, что значительно экономит время и ресурсы.

1.2 Обзор исследований

Е.А. Терновой исследует возможности библиотек Python для обработки изображений, таких как Pillow, Scikit-image, OpenCV и Keras, с особым акцентом на использование Pillow. Описание методов сжатия изображений, включая SVD-разложение, и создание графического интерфейса с помощью Tkinter, делает работу актуальной для разработчиков, работающих с мультимедийными данными и заинтересованных в оптимизации хранения и обработки изображений [1].

А.Т.А. Аль-Балдавия исследует использование библиотек Python для обработки изображений. В статье рассматриваются библиотеки, которые предоставляют мощные инструменты для обработки и анализа изображений, включая фильтрацию и работу с массивами данных. Исследование полезно для специалистов в области анализа изображений и всех, кто использует Python для научных и прикладных задач, связанных с визуализацией данных и обработкой изображений [2].

Ш.Т. Касимова фокусируется на цифровой обработке изображений и звука с использованием Python, делая акцент на различиях между цифровой обработкой изображений и компьютерным зрением. Работа полезна для специалистов, работающих с аудиофайлами и изображениями, а также для тех, кто разрабатывает аналитические инструменты для мультимедийных данных [3].

Е.В. Покидько и Ю.Е. Карякин представляют в своей статье решение для автоматизации обработки изображений в фотосалонах с помощью цифрового помощника. Основная цель разработки — ускорить рутинные операции, такие как кадрирование и восстановление дефектных фотографий, с использованием библиотек Python (Pillow для работы с изображениями и rembg для удаления фона). Приложение значительно ускоряет процесс восстановления старых

фотографий и кадрирования изображений, что позволяет специалистам в фотосалонах эффективно выполнять задачи и концентрироваться на более сложных операциях. Это исследование полезно для специалистов, работающих в сфере обработки фотографий и изображений, а также для тех, кто ищет способы оптимизации рутинных процессов в фотосалонах [4].

И.Р. Самандаров, Ш.Т. Маншуров и Н.Т. Душатов исследуют работу с изображениями с использованием языка программирования C++ и библиотеки OpenCV. В статье подробно рассматриваются популярные форматы изображений (JPEG, PNG, GIF, TIFF), их особенности и области применения. Также авторы акцентируют внимание на том, как выбрать подходящий формат и инструменты для обработки изображений в зависимости от конкретных задач. Это исследование является полезным для разработчиков, в сфере обработки изображений и разработке решений в области компьютерного зрения, а также для тех, кто ищет способы работы с изображениями в программных проектах [5].

1.3 Цель исследования

Целью данной работы является разработка программного приложения для автоматизированной обработки изображений с использованием Python и библиотеки Pillow. Это приложение должно позволить пользователям эффективно изменять параметры изображений, такие как размер, угол поворота, яркость, контрастность, а также накладывать водяные знаки и применять различные фильтры для улучшения качества изображений.

2 Материалы и методы

Для разработки программы был использован язык программирования Python и библиотека Pillow, которая предоставляет удобные методы для работы с изображениями. Для создания графического пользовательского интерфейса использована библиотека Tkinter. В качестве методов обработки изображений использованы алгоритмы масштабирования (resize), поворота (rotate), наложения текста (watermark), изменения яркости, контраста, резкости и применения различных фильтров (размытие, инверсия цветов). Для эффективной работы с изображениями применяется хранение истории изменений с возможностью отмены действий. Также используется создание полупрозрачных водяных знаков и корректировка изображений по заданным параметрам.

3 Результаты и обсуждения

1. Импорт библиотек (рис.1).

```
import os
from tkinter import *
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk, ImageDraw, ImageFont, ImageEnhance, ImageFilter
import tkinter.font as tkFont
```

Рисунок 1. Импорт библиотек

Здесь импортируются все необходимые библиотеки:

- os — для работы с файлами (например, проверка расширений изображений).
- tkinter — для создания графического интерфейса.
- PIL (Pillow) — для работы с изображениями (изменение размера, добавление водяного знака, фильтры и т.д.).
- tkinter.font — для работы с шрифтами.

2. Функции для обработки изображений.

2.1. Проверка формата изображения (рис.2).

```
def is_image_format_supported(image_path): 1 usage
# Проверка расширения файла
supported_formats = ['.jpg', '.jpeg', '.png', '.gif', '.bmp']
_, ext = os.path.splitext(image_path)
return ext.lower() in supported_formats
```

Рисунок 2. Код

Эта функция проверяет, поддерживает ли программа данный формат изображения. Для этого получаем расширение файла с помощью `os.path.splitext()` и проверяем его в списке поддерживаемых форматов.

2.2. Функции для обработки изображений

Изменение размера изображения (рис.3).

```
def resize_image(image, width, height): 1
    return image.resize((width, height))
```

Рисунок 3. Код изменения размера изображения

Эта функция изменяет размеры изображения на заданные `width` и `height`. Поворот изображения (рис.4).

```
def rotate_image(image, angle): 1 usage
    return image.rotate(angle, expand=True)
```

Рисунок 4. Код поворота изображения

Поворачивает изображение на угол `angle`. Параметр `expand=True` позволяет избежать обрезки изображения при повороте.

Добавление водяного знака (рис.5):

```
def add_watermark(image, watermark_text, font_path, opacity): 1 usage
    # Создаем копию изображения
    watermark_image = image.copy()
    draw = ImageDraw.Draw(watermark_image)
    width, height = watermark_image.size

    # Настроим шрифт
    font = ImageFont.truetype(font_path, size=36)

    # Получаем размер текста с помощью textbbox
    bbox = draw.textbbox(xy=(0, 0), watermark_text, font=font)
    text_width = bbox[2] - bbox[0]
    text_height = bbox[3] - bbox[1]

    # Позиция текста (внизу справа)
    position = (width - text_width - 10, height - text_height - 10)

    # Создание полупрозрачного слоя для текста
    text_image = Image.new(mode='RGBA', watermark_image.size, color=(0, 0, 0, 0))
    text_draw = ImageDraw.Draw(text_image)
    text_draw.text(position, watermark_text, font=font, fill=(255, 255, 255, int(255 * opacity)))

    # Наложение текста на изображение
    watermark_image = Image.alpha_composite(watermark_image.convert('RGBA'), text_image)

    return watermark_image.convert('RGB') # Возвращаем в RGB для дальнейшей обработки
```

Рисунок 5. Код водяного знака

Эта функция добавляет водяной знак на изображение. Текст водяного знака будет внизу справа. Прозрачность текста регулируется параметром `opacity`.

Коррекция яркости, контрастности и других эффектов (рис.6).

```
def adjust_brightness(image, factor): 1 usage
    enhancer = ImageEnhance.Brightness(image)
    return enhancer.enhance(factor)

def adjust_contrast(image, factor): 1 usage
    enhancer = ImageEnhance.Contrast(image)
    return enhancer.enhance(factor)

def apply_blur(image, radius): 1 usage
    return image.filter(ImageFilter.GaussianBlur(radius))

def invert_colors(image): 1 usage
    return Image.eval(image, *args: lambda x: 255 - x)

def adjust_sharpness(image, factor): 1 usage
    enhancer = ImageEnhance.Sharpness(image)
    return enhancer.enhance(factor)
```

Рисунок 6. Код коррекции яркости, контрастности, размытия, инверсии цвета, резкости

Эти функции позволяют изменять яркость, контрастность, применить размытие, инвертировать цвета и регулировать резкость изображения.

3. Функции для работы с изображениями (открытие и сохранение).

3.1. Открытие изображения (рис.7).

```
def open_image(): 1 usage
    global img_path, img, img_display
    img_path = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg;*.jpeg;*.png;*.gif;*.bmp")])

    if img_path and is_image_format_supported(img_path):
        img = Image.open(img_path)
        history.append(img.copy()) # Сохраняем состояние для отмены
        img_display = ImageTk.PhotoImage(img)
        update_image_display()
    else:
        messagebox.showerror(title="Ошибка", message="Неподдерживаемый формат изображения!")
```

Рисунок 7. Код открытия изображения

Функция открывает диалоговое окно для выбора файла изображения. После того как файл выбран, программа проверяет его формат и, если он поддерживается, загружает изображение и сохраняет его в историю для возможной отмены.

3.2. Сохранение изображения (рис.8).

```
def save_image(): 1 usage
    global img
    save_path = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("Image files", "*.png;*.jpg;*.jpeg")])
    if save_path:
        img.save(save_path)
        messagebox.showinfo(title="Сохранение", message="Изображение успешно сохранено!")
```

Рисунок 8. Код сохранения изображения

Функция для сохранения изображения в файл. Открывает диалоговое окно для выбора места и имени файла.

4. Функции для применения эффектов

Каждая из этих функций применяется к изображению и изменяет его в соответствии с выбранными пользователем параметрами.

Применение изменение размера изображения (рис.9).

```
def apply_resize(): 1 usage
    global img, img_display
    try:
        width = int(entry_width.get())
        height = int(entry_height.get())
        history.append(img.copy()) # Сохраняем состояние для отмены
        img = resize_image(img, width, height)
        update_image_display()
    except ValueError:
        messagebox.showerror(title="Ошибка", message="Введите корректные значения ширины и высоты!")
```

Рисунок 9. Код применения изменения размера изображения

Применение поворота изображения (рис.10).

```
def apply_rotate(): 1 usage
    global img, img_display
    try:
        angle = int(entry_angle.get())
        history.append(img.copy()) # Сохраняем состояние для отмены
        img = rotate_image(img, angle)
        update_image_display()
    except ValueError:
        messagebox.showerror(title="Ошибка", message="Введите корректное значение угла!")
```

Рисунок 10. Код применения поворота изображения

Применение добавления водяного знака (рис.11).

```
def apply_watermark(): 1 usage
    global img, img_display
    watermark_text = entry_watermark.get()
    if watermark_text:
        font_path = font_path_var.get()
        opacity = float(opacity_var.get())
        history.append(img.copy()) # Сохраняем состояние для отмены
        img = add_watermark(img, watermark_text, font_path, opacity)
        update_image_display()
    else:
        messagebox.showerror(title="Ошибка", message="Введите текст для водяного знака!")
```

Рисунок 11. Код применения добавления водяного знака

Применение коррекции яркости, контрастности и другие эффекты (рис.12.1-12.5).

```
def apply_brightness(): 1 usage
    global img, img_display
    try:
        factor = float(entry_brightness.get())
        history.append(img.copy()) # Сохраняем состояние для отмены
        img = adjust_brightness(img, factor)
        update_image_display()
    except ValueError:
        messagebox.showerror( title: "Ошибка", message: "Введите корректное значение яркости!")
```

Рисунок 12.1. Код применения яркости

```
def apply_contrast(): 1 usage
    global img, img_display
    try:
        factor = float(entry_contrast.get())
        history.append(img.copy()) # Сохраняем состояние для отмены
        img = adjust_contrast(img, factor)
        update_image_display()
    except ValueError:
        messagebox.showerror( title: "Ошибка", message: "Введите корректное значение контрастности!")
```

Рисунок 12.2. Код применения контраста

```
def apply_blur_action(): 1 usage
    global img, img_display
    try:
        radius = float(entry_blur.get())
        history.append(img.copy()) # Сохраняем состояние для отмены
        img = apply_blur(img, radius)
        update_image_display()
    except ValueError:
        messagebox.showerror( title: "Ошибка", message: "Введите корректное значение радиуса размытия!")
```

Рисунок 12.3. Код применения размытия

```
def invert_colors_action(): 1 u
    global img, img_display
    history.append(img.copy())
    img = invert_colors(img)
    update_image_display()
```

Рисунок 12.4. Код применения инверсии цвета

```
def adjust_sharpness_action(): 1 usage
    global img, img_display
    try:
        factor = float(entry_sharpness.get())
        history.append(img.copy()) # Сохраняем состояние для отмены
        img = adjust_sharpness(img, factor)
        update_image_display()
    except ValueError:
        messagebox.showerror( title: "Ошибка", message: "Введите корректное значение резкости!")
```

Рисунок 12.5. Код применения резкости

5. Функция для отмены изменений (рис.13).

```
def undo_action(): 1 usage
    global img, img_display
    if len(history) > 1:
        history.pop() # Удаляем текущее состояние
        img = history[-1].copy() # Восстанавливаем последнее состояние
        img_display = ImageTk.PhotoImage(img)
        update_image_display()
    else:
        messagebox.showwarning( title: "Отмена", message: "Нет доступных действий для отмены!")
```

Рисунок 13. Код функции отмены действия

Функция для отмены последнего изменения. Если в истории есть хотя бы одно состояние, она восстанавливает последнее изображение и обновляет отображение.

6. Обновление отображения изображения (рис.14).

```
def update_image_display(): 10 usages
    global img, img_display
    # Подгонка изображения под размеры окна
    img_width, img_height = img.size
    window_width, window_height = root.winfo_width(), root.winfo_height()

    # Убираем отступы для окна
    max_width = window_width - 50 # Отступы от краев
    max_height = window_height - 150 # Отступы от верхней панели с кнопками

    # Расчет пропорций для масштабирования
    ratio = min(max_width / img_width, max_height / img_height)
    new_width = int(img_width * ratio)
    new_height = int(img_height * ratio)

    # Масштабируем изображение с использованием нового метода
    img_resized = img.resize( size: (new_width, new_height), Image.Resampling.LANCZOS)
    img_display = ImageTk.PhotoImage(img_resized)

    # Обновляем изображение на холсте
    canvas.delete("all")
    canvas.create_image( *args: 0, 0, anchor=NW, image=img_display)
    canvas.config(scrollregion=canvas.bbox(ALL)) # Обновляем область прокрутки, если есть
```

Рисунок 14. Код обновления отображения изображения

Эта функция отвечает за обновление отображаемого изображения в окне. Изображение масштабируется, чтобы поместиться в окно, при этом сохраняются пропорции.

7. Интерфейс и запуск программы (рис.15.1-15.)

```
# Инициализация главного окна
root = Tk()
root.title("Автоматизация обработки изображений с Pillow")
root.geometry("800x600")

# Холст для отображения изображения
canvas = Canvas(root, width=600, height=400)
canvas.pack()

# Кнопки и поля ввода
frame_controls = Frame(root)
frame_controls.pack(pady=10)

btn_open = Button(frame_controls, text="Открыть изображение", command=open_image)
btn_open.grid(row=0, column=0, padx=10)

btn_save = Button(frame_controls, text="Сохранить изображение", command=save_image)
btn_save.grid(row=0, column=1, padx=10)

# Поля для изменения размера
label_width = Label(frame_controls, text="Ширина:")
label_width.grid(row=1, column=0, padx=5)
entry_width = Entry(frame_controls)
entry_width.grid(row=1, column=1)

label_height = Label(frame_controls, text="Высота:")
label_height.grid(row=1, column=2, padx=5)
entry_height = Entry(frame_controls)
entry_height.grid(row=1, column=3)

btn_resize = Button(frame_controls, text="Изменить размер", command=apply_resize)
btn_resize.grid(row=1, column=4, padx=10)

# Поле для поворота
label_angle = Label(frame_controls, text="Угол поворота:")
label_angle.grid(row=2, column=0, padx=5)
entry_angle = Entry(frame_controls)
entry_angle.grid(row=2, column=1)

btn_rotate = Button(frame_controls, text="Повернуть", command=apply_rotate)
btn_rotate.grid(row=2, column=2, padx=10)

# Поле для водяного знака
label_watermark = Label(frame_controls, text="Текст водяного знака:")
label_watermark.grid(row=3, column=0, padx=5)
entry_watermark = Entry(frame_controls)
entry_watermark.grid(row=3, column=1)

# Шрифт и прозрачность водяного знака
font_path_var = StringVar(value="arial.ttf")
opacity_var = StringVar(value="0.5")

btn_watermark = Button(frame_controls, text="Добавить водяной знак", command=apply_watermark)
btn_watermark.grid(row=3, column=2, padx=10)
```

Рисунок 15.1. Код интерфейса

```
# Настройка яркости и контрастности
label_brightness = Label(frame_controls, text="Яркость:")
label_brightness.grid(row=4, column=0, padx=5)
entry_brightness = Entry(frame_controls)
entry_brightness.grid(row=4, column=1)

btn_brightness = Button(frame_controls, text="Применить яркость", command=apply_brightness)
btn_brightness.grid(row=4, column=2, padx=10)

label_contrast = Label(frame_controls, text="Контрастность:")
label_contrast.grid(row=5, column=0, padx=5)
entry_contrast = Entry(frame_controls)
entry_contrast.grid(row=5, column=1)

btn_contrast = Button(frame_controls, text="Применить контрастность", command=apply_contrast)
btn_contrast.grid(row=5, column=2, padx=10)

# Гайссово размытие
label_blur = Label(frame_controls, text="Радиус размытия:")
label_blur.grid(row=6, column=0, padx=5)
entry_blur = Entry(frame_controls)
entry_blur.grid(row=6, column=1)

btn_blur = Button(frame_controls, text="Применить размытие", command=apply_blur_action)
btn_blur.grid(row=6, column=2, padx=10)

# Инверсия цветов
btn_invert = Button(frame_controls, text="Инвертировать цвета", command=invert_colors_action)
btn_invert.grid(row=7, column=0, padx=10)

# Резкость
label_sharpness = Label(frame_controls, text="Резкость:")
label_sharpness.grid(row=8, column=0, padx=5)
entry_sharpness = Entry(frame_controls)
entry_sharpness.grid(row=8, column=1)

btn_sharpness = Button(frame_controls, text="Применить резкость", command=adjust_sharpness_action)
btn_sharpness.grid(row=8, column=2, padx=10)

# Кнопка отмены действия
btn_undo = Button(frame_controls, text="Отменить действие", command=undo_action)
btn_undo.grid(row=9, column=0, columnspan=5, pady=10)

# История изображений (для отмены)
history = []

# Запуск приложения
root.mainloop()
```

Рисунок 15.2. Код интерфейса и запуска

Создается основное окно программы, на котором располагаются все элементы интерфейса. После чего запускается.

Внешний вид приложения (рис.16).

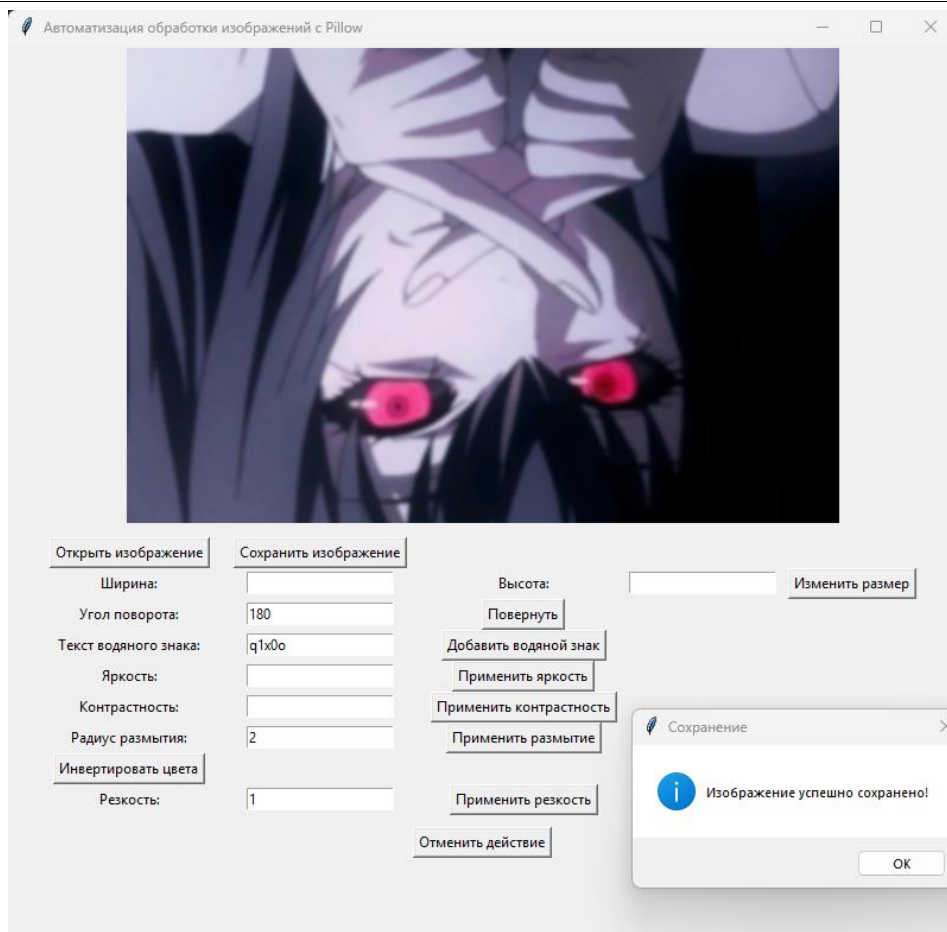


Рисунок 16. Внешний вид приложения

Выводы

Разработанное приложение для автоматической обработки изображений с помощью Python и библиотеки Pillow успешно решает задачу массовой обработки изображений. Оно предоставляет пользователю удобный интерфейс для выполнения операций, таких как изменение размера, поворот, настройка яркости и контрастности, а также добавление водяных знаков. Применение данной программы позволяет значительно ускорить процесс обработки изображений и повысить производительность работы специалистов, занимающихся графическим дизайном, фотографией и другими областями, связанными с визуальными материалами. Работа продемонстрировала потенциал Python и Pillow для создания эффективных инструментов для обработки изображений с учетом современных требований и нужд пользователей.

Библиографический список

1. Терновой Е.А. Библиотеки python для обработки изображений. примеры использования // Всероссийская студенческая конференция «студенческая научная весна», посвященная 85-летию Ю.А. Гагарина. 2019. С. 314.
2. Аль-Балдави А.Т.А. Инструменты python для работы с изображениями // Академическая наука на службе обществу. 2022. С. 144-152.

3. Касимова Ш.Т. Цифровая обработка изображения и звука на python // Innovations in Science and Technologies. 2024. Т. 1. №. 1. С. 136-150.
4. Покидько Е.В., Карякин Ю.Е. Цифровой помощник обработки изображений // Математическое и информационное моделирование. 2021. С. 425-429.
5. Самандаров И.Р., Маншуров Ш.Т., Душатов Н.Т. Обработка изображений в с++ с помощью библиотеки opencv // Universum: технические науки. 2023. №. 5-2 (110). С. 19-28.