ГУАП

КАФЕДРА № 12

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Доцент, канд. техн. наук		А.С. Костин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

Основы работы с цифровыми изображениями

по курсу: Интеллектуальные системы

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 1346	25.02.2026	Д. И. Мигранов
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2026

1 Цель работы

Освоить базовую обработку цифровых изображений в OpenCV, подтвердить понимание представления изображения как массива (shape/dtype/каналы), выполнить типовые преобразования и корректно оформить результаты в виде отчёта по ГОСТ.

2 Теоретические сведения

2.1. Пиксель и представление изображения

Цифровое изображение представляет собой двумерный или трёхмерный массив целых чисел. Каждый элемент массива называется пиксель (picture element). Значение пикселя определяет его цвет или интенсивность света в данной позиции. Для цветного изображения каждый пиксель содержит несколько компонентов (каналов), обычно три для RGB или BGR.

2.2. Shape и dtype

Shape (форма) массива определяет его размеры. Для цветного изображения shape имеет вид (H, W, C), где H — высота в пикселях, W — ширина в пикселях, C — количество каналов. Для изображения в оттенках серого shape равен (H, W) — двумерный массив. Dtype (тип данных) определяет тип элементов массива. Для изображений обычно используется uint8 (целое число без знака от 0 до 255).

2.3. Цветовое пространство BGR

OpenCV использует цветовое пространство BGR (Blue, Green, Red) вместо привычного RGB. Это означает, что первый канал содержит значения синего, второй — зелёного, третий — красного. При работе с OpenCV необходимо помнить об этом порядке для корректного отображения цветов.

2.4. Преобразования изображений

Основные преобразования включают: масштабирование (изменение размера), поворот (геометрическое преобразование), преобразование цветовых пространств, работу с каналами, фильтрацию (размытие, резкость) и регулировку яркости и контраста.

3 Ход работы и результаты

3.1 Параметры выполнения

Вариант: 15

OpenCV[3] версия: 4.13.0

Расчет параметров по формулам:

- Коэффициенты уменьшения: fx_small = 0.30 + 0.01 * 15 = 0.45; fy_small = 0.30 + 0.005 * 15 = 0.375
- Коэффициенты увеличения: fx_big = 1.20 + 0.01 * 15 = 1.35; fy_big = 1.20 + 0.005 * 15 = 1.275

- Поворот: 15 mod 3 = 0, следовательно, поворот на 90° по часовой стрелке.
- Параметр яркости: beta = (15 mod 5) * 20 = 0
- Размеры холста (canvas): H = 240 + 5 * 15 = 315; W = 320 + 3 * 15 = 365

Полный код представлен в приложении 1.

Названия изображений:

- Исходное изображение – `'image_a6cf7d.jpg'`
- Оттенки серого – out_gray_15.png
- Уменьшенное изображение – out_resize_small_15.png
- Увеличенное изображение – out_resize_big_15.png
- Повернутое изображение – out_rotate_15.png
- Сгенерированное изображение – out_canvas_15.png
- Смена каналов – out_swap_channels_7.png
- Изменение яркости – out_brightness_15.png
- Изменение контрастности – out_contrast_15.png

3.2 Подготовка среды

```
# 5.1 Подготовка среды
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
print("cv2 version:", cv2.__version__)
```

3.3 Загрузка изображения

Был введён путь к изображению `image_a6cf7d`.jpg. Исходное изображение показано на рисунке 1.

```
# 5.2 Загрузка изображения
fname = 'image_a6cf7d.jpg'
img = cv2.imread(fname, cv2.IMREAD_COLOR)
if img is None:
    raise FileNotFoundError(f"Не удалось прочитать файл: {fname}")

# 5.4 Отображение исходного изображения
print("Исходное изображение:")
cv2_imshow(img)
```

Рисунок 1 – Исходное изображение

3.4 Контроль свойств

```
# 5.3 Контроль свойств
print("shape:", img.shape)
print("dtype:", img.dtype)

# Вывод пикселя (координаты выбраны произвольно для примера)
y, x = 200, 150
px = img[y, x]
print(f"pixel(B,G,R) at ({y},{x}) =", px)
# Порядок каналов в OpenCV — BGR.
)
```

Порядок каналов BGR: исторически OpenCV использует BGR вместо более привычного RGB из-за особенностей ранних версий библиотеки.

3.5 Оттенки серого

```
# 5.5 Оттенки серого
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
print("\nОттенки серого (shape):", gray.shape)
cv2_imshow(gray)
cv2.imwrite(f"out_gray_{V}.png", gray)
```

Изображение было преобразовано в градации серого функцией cv2.cvtColor. Атрибут shape изменился, так как цветное изображение имеет 3 канала, а grayscale — только 1 канал

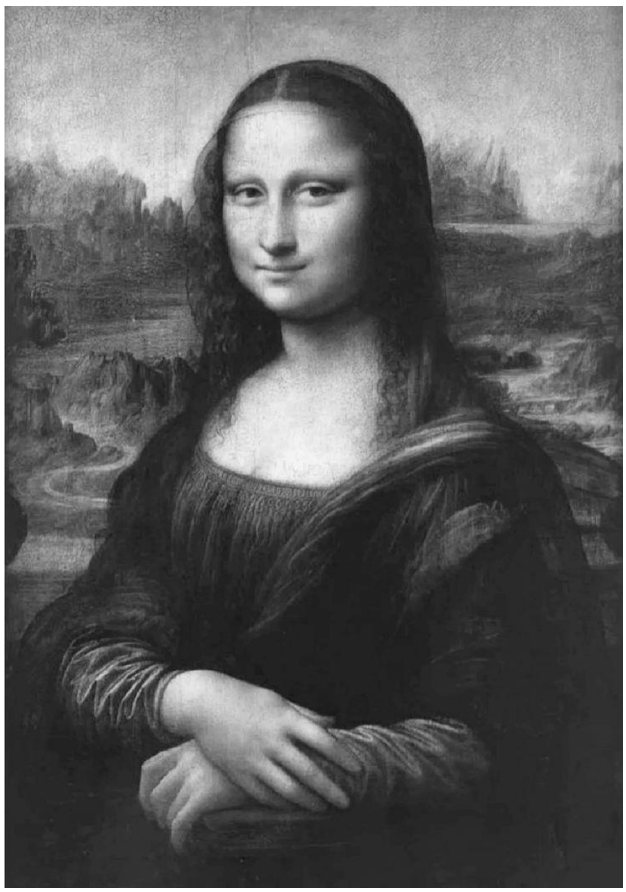

Рисунок 2 – Оттенки серого

3.6 Масштабирование

```
# 5.6 Масштабирование
fx_small = 0.30 + 0.01 * V
fy_small = 0.30 + 0.005 * V
fx_big = 1.20 + 0.01 * V
fy_big = 1.20 + 0.005 * V

small = cv2.resize(img, (0, 0), fx=fx_small, fy=fy_small)
big = cv2.resize(img, (0, 0), fx=fx_big, fy=fy_big)

print("\nУменьшенное изображение:")
cv2_imshow(small)
cv2.imwrite(f"out_resize_small_{V}.png", small)

print("\nУвеличенное изображение:")
cv2_imshow(big)
cv2.imwrite(f"out_resize_big_{V}.png", big)
```

Выполнено уменьшение и увеличение изображения согласно рассчитанным коэффициентам. При уменьшении (Рисунок 3) наблюдается потеря детализации, так как несколько исходных пикселей сливаются воедино. Это необратимая операция: увеличив потом маленькое изображение, мы не восстановим утраченные детали.

При увеличении (Рисунок 4) заметно появление артефактов интерполяции из-за программного вычисления новых пикселей. Это делается с помощью интерполяции – математического алгоритма вычисления промежуточных значений. Результат – размытие или ступенчатость (артефакты интерполяции).

Рисунок 3 – Уменьшенное изображение (45 % от исходного)

Рисунок 4 – Увеличенное изображение (135% от исходного размера)

3.7 Поворот

```
# 5.7 Поворот (V mod 3 = 15 % 3 = 0 -> 90 градусов по часовой стрелке)
rot = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
print("\nПовернутое изображение (shape):", rot.shape)
cv2_imshow(rot)
cv2.imwrite(f"out_rotate_{V}.png", rot)
```

Согласно варианту, выполнен поворот на 90° по часовой стрелке с помощью cv2.rotate. При повороте на 90° атрибут shape изменяется: ширина и высота меняются местами.

Рисунок 5 – Повернутое изображение

3.8 Разделение каналов и сборка

```
# 5.8 Работа с каналами
b, g, r = cv2.split(img)
img_swap = cv2.merge([r, g, b])
print("\nИзображение с переставленными каналами B и R:")
cv2_imshow(img_swap)
cv2.imwrite(f"out_swap_channels_{V}.png", img_swap)
```

Изображение было разделено на каналы с помощью функции split, после чего собрано обратно функцией merge с перестановкой каналов Red и Blue (собрано в порядке RGB вместо стандартного BGR). В результате перестановки цветов синие оттенки стали красными, а красные — синими

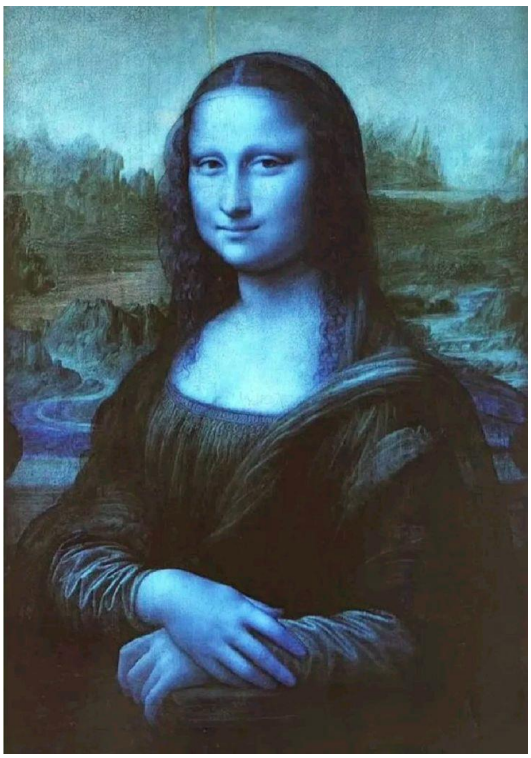

Рисунок 6 – Изображение с переставленными каналами

3.9 Генерация уникального изображения (Canvas)

```
# 6 Генерация уникального цветного изображения (canvas)
H = 240 + 5 * V
W = 320 + 3 * V
canvas = np.full((H, W, 3), 255, dtype=np.uint8)

# Рисуем примитивы
cv2.line(canvas, (20, 20), (W-20, H-20), (255, 0, 0), 3) # Синяя линия
cv2.rectangle(canvas, (50, 50), (200, 150), (0, 255, 0), -1) # Зеленый
залитый прямоугольник
cv2.circle(canvas, (W//2, H//2), 40, (0, 0, 255), 2) # Красная
окружность
cv2.putText(canvas, f"Variant {V}", (10, H-30),
cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 0), 2) # Черный текст

print("\nСгенерированное изображение (Canvas):")
cv2_imshow(canvas)
cv2.imwrite(f"out_canvas_{V}.png", canvas)
```

Создан холст размером 315x365 пикселей с белым фоном. На нем нарисованы: синяя линия (толщина 3), зеленый прямоугольник с заливкой (толщина -1), красная окружность (толщина 2) и нанесен текст черного цвета "Variant 15"

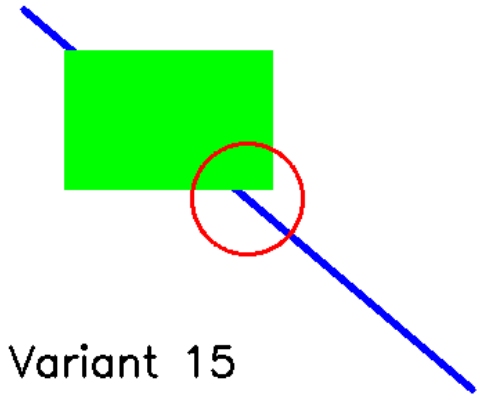

Рисунок 7 - Canvas

3.10 Расширенная часть: регулировка яркости и контраста

```
# 7 Расширенная часть: Вариант A (Яркость и контраст)
beta = (V % 5) * 20 # 15 % 5 = 0, поэтому beta = 0

brighter = cv2.convertScaleAbs(img, alpha=1.0, beta=beta)
contrast = cv2.convertScaleAbs(img, alpha=1.5, beta=0)

print(f"\nИзменение яркости (beta={beta}):")
cv2_imshow(brighter)
cv2.imwrite(f"out_brightness_{V}.png", brighter)

print("\nИзменение контраста (alpha=1.5):")
cv2_imshow(contrast)
cv2.imwrite(f"out_contrast_{V}.png", contrast)
```

Выполнена часть A (рекомендуемая вариант): регулировка яркости и контраста через convertScaleAbs.

- Увеличение яркости:

  Использована формула dst = |src × 1.0 + β|, где β = 0. Параметр яркости вычислен как (V mod 5) × 20 = (15 mod 5) × 20 = 0. В данном варианте яркость не изменяется. Функция cv2.convertScaleAbs() выполняет следующие операции: умножает каждый пиксель на коэффициент, добавляет смещение, вычисляет абсолютное значение и насыщает результат в диапазон [0, 255].

- Изменение контраста:

  Использована формула dst = |src × 1.5 + 0|. Коэффициент 1.5 увеличивает контраст: светлые области становятся светлее, тёмные — темнее. Функция convertScaleAbs() обеспечивает безопасность от переполнения целого типа, автоматически насыщая значения в диапазоне [0, 255]. Это предпочтительнее, чем простое сложение img + beta, которое может привести к переполнению

- Суть проблемы и насыщение диапазона 0..255:

  По умолчанию изображения в OpenCV хранятся как массивы с типом данных `uint8` (8-битные целые числа без знака). Это означает, что физически один канал пикселя может хранить только значения строго от 0 (черный) до 255 (белый)

  Насыщение (saturation) – это математический механизм ограничения. Если вычисленное значение пикселя становится меньше 0, оно принудительно приравнивается к 0, а если превышает 255 – обрезается до 255.

  Если к массиву uint8 напрямую прибавить число (например, img + 40), может произойти переполнение памяти (overflow)

  cv2.convertScaleAbs - эта встроенная функция OpenCV специально создана для безопасного изменения пикселей.

Рисунок 8 – Изображение с увеличенной яркостью

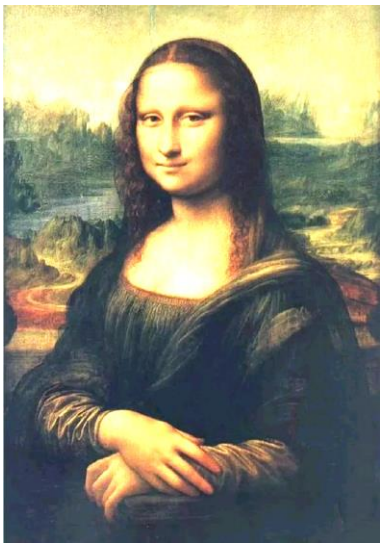

Рисунок 9 – Изображение с увеличенным контрастом

4 Выводы

В ходе выполнения лабораторной работы были успешно освоены фундаментальные методы обработки цифровых изображений с использованием библиотеки OpenCV.

Практически подтверждено, что в компьютерном зрении растровое изображение интерпретируется как многомерный массив данных (numpy.ndarray). Анализ базовых атрибутов shape и dtype позволил наглядно изучить структуру картинки: геометрические размеры, количество цветовых каналов и 8-битный беззнаковый целочисленный формат хранения интенсивностей (uint8).

При выполнении геометрических преобразований установлено, что уменьшение масштаба приводит к безвозвратной потере высокочастотных деталей изображения из-за объединения смежных пикселей. Увеличение масштаба, напротив, требует применения математической интерполяции для генерации новых пикселей на основе соседних. Операция поворота изображения на 90 градусов по часовой стрелке продемонстрировала изменение размерности матрицы: первые две оси (высота и ширина) закономерно меняются местами.

Работа с цветовой моделью путем разделения изображения на каналы (split) и их последующей сборки (merge с перестановкой) подтвердила специфику библиотеки OpenCV — использование формата BGR по умолчанию. Это наглядно доказало, что цвет каждого пикселя формируется из трех независимых слоев интенсивности, которыми можно манипулировать отдельно.

Программная генерация пустого холста (canvas) и отрисовка на нем геометрических примитивов закрепили понимание базового принципа компьютерной графики: любое цифровое изображение является числовой матрицей, а процесс рисования сводится к перезаписи значений конкретных элементов этого массива.

В ходе изучения фотометрических преобразований были проанализированы ограничения целочисленного формата uint8. Доказана критическая необходимость использования специализированных функций (таких как cv2.convertScaleAbs) при изменении контраста и яркости. В отличие от прямого математического сложения, вызывающего переполнение памяти и графические артефакты, данная функция корректно применяет механизм насыщения (saturation). В результате значения, выходящие за границы допустимого диапазона [0, 255], безопасно ограничиваются чистым черным (0) или чистым белым (255) цветом, сохраняя общую визуальную целостность изображения ценой потери детализации в экстремальных тенях и светах (clipping).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Костин, А. С. Лекция 1. Введение в компьютерное зрение и основы цифровых изображений. ГУАП, 2026.

2. Костин, А. С. Лабораторная работа №1: Основы работы с цифровыми изображениями. ГУАП, 2026.

3. OpenCV: Image Processing [Электронный ресурс]. — Режим доступа: https://docs.opencv.org/4.x/d7/da8/tutorial_py_image_processing.html, свободный.

4. NumPy Reference [Электронный ресурс]. — Режим доступа: https://numpy.org/doc/stable/reference/, свободный.

ПРИЛОЖЕНИЕ 1

Код программы

```
# 5.1 Подготовка среды
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

print("cv2 version:", cv2.__version__)

# 5.2 Загрузка изображения
fname = 'image_a6cf7d.jpg'
img = cv2.imread(fname, cv2.IMREAD_COLOR)

if img is None:
    raise FileNotFoundError(f"Не удалось прочитать файл: {fname}")

# 5.3 Контроль свойств
print("shape:", img.shape)
print("dtype:", img.dtype)

# Вывод пикселя (координаты выбраны произвольно для примера)
y, x = 200, 150
px = img[y, x]
print(f"pixel(B,G,R) at ({y},{x}) =", px)
# Порядок каналов в OpenCV — BGR.

# 5.4 Отображение исходного изображения
print("Исходное изображение:")
cv2_imshow(img)

# Параметры для Варианта 15
V = 15

# 5.5 Оттенки серого
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
print("\nОттенки серого (shape):", gray.shape)
cv2_imshow(gray)
cv2.imwrite(f"out_gray_{V}.png", gray)

# 5.6 Масштабирование
fx_small = 0.30 + 0.01 * V
fy_small = 0.30 + 0.005 * V
fx_big = 1.20 + 0.01 * V
fy_big = 1.20 + 0.005 * V

small = cv2.resize(img, (0, 0), fx=fx_small, fy=fy_small)
big = cv2.resize(img, (0, 0), fx=fx_big, fy=fy_big)

print("\nУменьшенное изображение:")
```

```
cv2_imshow(small)
cv2.imwrite(f"out_resize_small_{V}.png", small)

print("\nУвеличенное изображение:")
cv2_imshow(big)
cv2.imwrite(f"out_resize_big_{V}.png", big)

# 5.7 Поворот (V mod 3 = 15 % 3 = 0 -> 90 градусов по часовой стрелке)
rot = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
print("\nПовернутое изображение (shape):", rot.shape)
cv2_imshow(rot)
cv2.imwrite(f"out_rotate_{V}.png", rot)

# 5.8 Работа с каналами
b, g, r = cv2.split(img)
img_swap = cv2.merge([r, g, b])
print("\nИзображение с переставленными каналами B и R:")
cv2_imshow(img_swap)
cv2.imwrite(f"out_swap_channels_{V}.png", img_swap)

# 6 Генерация уникального цветного изображения (canvas)
H = 240 + 5 * V
W = 320 + 3 * V
canvas = np.full((H, W, 3), 255, dtype=np.uint8)

# Рисуем примитивы
cv2.line(canvas, (20, 20), (W-20, H-20), (255, 0, 0), 3) # Синяя линия
cv2.rectangle(canvas, (50, 50), (200, 150), (0, 255, 0), -1) # Зеленый
залитый прямоугольник
cv2.circle(canvas, (W//2, H//2), 40, (0, 0, 255), 2) # Красная окружность
cv2.putText(canvas, f"Variant {V}", (10, H-30), cv2.FONT_HERSHEY_SIMPLEX,
1.0, (0, 0, 0), 2) # Черный текст

print("\nСгенерированное изображение (Canvas):")
cv2_imshow(canvas)
cv2.imwrite(f"out_canvas_{V}.png", canvas)

# 7 Расширенная часть: Вариант А (Яркость и контраст)
beta = (V % 5) * 20 # 15 % 5 = 0, поэтому beta = 0

brighter = cv2.convertScaleAbs(img, alpha=1.0, beta=beta)
contrast = cv2.convertScaleAbs(img, alpha=1.5, beta=0)

print(f"\nИзменение яркости (beta={beta}):")
cv2_imshow(brighter)
cv2.imwrite(f"out_brightness_{V}.png", brighter)

print("\nИзменение контраста (alpha=1.5):")
cv2_imshow(contrast)
cv2.imwrite(f"out_contrast_{V}.png", contrast)
```