Лабораторная работа №1: Основы работы с цифровыми изображениями

Дисциплина: **Интеллектуальные системы**

Место лабораторной работы в структуре курса

Изображение как данные

Понимание цифрового изображения как многомерного массива чисел — фундаментальная концепция компьютерного зрения. Каждый пиксель представлен числовыми значениями, определяющими цвет и интенсивность.

Базовые операции OpenCV

Освоение основных функций библиотеки OpenCV для чтения, обработки и сохранения изображений. Это инструментарий, который станет основой всех последующих алгоритмов.

Подготовка к сложным задачам

Базовые навыки работы с изображениями необходимы для детекции объектов, сегментации, применения нейросетевых моделей и других продвинутых методов машинного зрения.

Связь с последующими темами

Фильтрация, выделение границ, вычисление признаков, распознавание образов — все эти темы опираются на понимание структуры изображения и базовых операций преобразования.

Если вы уверенно работаете с массивами изображений, понимаете каналы и типы данных, все последующие алгоритмы будут восприниматься как «функции над массивами». Сегодня мы закладываем фундамент базовой грамотности в области обработки изображений.

Планируемые результаты обучения

По завершении лабораторной работы вы приобретёте следующие практические компетенции и навыки работы с цифровыми изображениями:

01

Загрузка и сохранение изображений

Умение корректно читать изображения из файлов, обрабатывать возможные ошибки и сохранять результаты в различных форматах с контролем успешности операции.

02

Корректный вывод в Google Colab

Освоение специфики визуализации изображений в среде Colab с использованием cv2_imshow, понимание ограничений оконных функций в облачной среде.

03

Анализ структуры данных

Чтение и интерпретация параметров shape и dtype массивов изображений, понимание связи между размерностью массива и типом изображения (цветное/оттенки серого).

04

Преобразование в оттенки серого

Применение цветового преобразования BGR→GRAY, понимание изменения размерности массива с 3 каналов до 1 канала и принципов вычисления яркости.

05

Геометрические преобразования

Выполнение операций масштабирования (resize), отражения (flip) и поворота (rotate) с пониманием влияния на размеры изображения и качество.

06

Генерация и рисование

Создание изображений программно, использование функций рисования примитивов (линии, прямоугольники, круги, текст) с пониманием системы координат.

07

Понимание формата BGR

Осознание особенностей порядка каналов в OpenCV (BGR вместо RGB), умение работать с отдельными цветовыми каналами и их перестановкой.

08

Подготовка отчёта по ГОСТ

Оформление технического отчёта с соблюдением стандартов: структура разделов, нумерация рисунков, ссылки на источники, приложения с кодом.

Что необходимо сдать: состав deliverables

Основные материалы

Письменный отчёт — главный объект проверки, оформленный по стандартам ГОСТ в формате PDF или DOCX.

Google Colab ноутбук — ссылка на выполненный код или приложение к отчёту для подтверждения воспроизводимости.

Набор результатов — сохранённые изображения-результаты всех ключевых операций с говорящими именами файлов.

Требования к содержанию

- Все изображения «до/после» ключевых операций должны быть представлены в отчёте с подробными подписями
- Краткие, но содержательные пояснения к каждой операции и наблюдаемым эффектам
- Обязательное указание номера варианта и параметров, обеспечивающих индивидуальность работы
- Код должен быть воспроизводимым: другой человек по вашему отчёту должен получить те же результаты
- Фиксация версий используемых библиотек и параметров среды выполнения

Ноутбук служит доказательством того, что код был выполнен вами лично и работает корректно. В отчёте обязательно демонстрируйте понимание процессов: не просто вставляйте скриншоты, а объясняйте, что происходит на каждом этапе обработки.

Среда выполнения: Google Colab + OpenCV

Критически важное правило визуализации

В Google Colab стандартные оконные функции OpenCV работают иначе, чем на локальном компьютере. Функция `cv2.imshow()` не будет работать в облачной среде.

Среда выполнения

Все задания выполняются исключительно в **Google Colab** — облачной среде для Jupyter ноутбуков с предустановленными библиотеками машинного обучения.

Основные библиотеки

Используем **OpenCV (cv2)** для обработки изображений и **NumPy** для работы с многомерными массивами. Это стандартный стек компьютерного зрения.

Исключаем matplotlib

Matplotlib не используется в данной лабораторной работе, чтобы избежать проблем с форматами BGR/RGB и обеспечить единообразие визуализации.

Единственный способ вывода

Вывод изображений осуществляется **только через cv2_imshow** из модуля `google.colab.patches`. Это решает все вопросы совместимости.

Соблюдение этого правила критически важно: использование неподходящих функций визуализации приведёт к ошибкам выполнения или некорректному отображению цветов. Функция cv2_imshow специально адаптирована для работы в среде Google Colab и автоматически обрабатывает формат BGR.

Базовые импорты и проверка версии библиотеки

Каждый ноутбук должен начинаться с импорта необходимых библиотек и проверки их версий. Это обеспечивает воспроизводимость результатов и помогает диагностировать потенциальные проблемы совместимости.

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

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

Импорт OpenCV

Библиотека cv2 предоставляет все функции для работы с изображениями: чтение, запись, преобразования, фильтры.

Импорт NumPy

NumPy необходим для работы с многомерными массивами — основной структурой данных для представления изображений.

Функция вывода

cv2_imshow из google.colab.patches — единственный корректный способ отображения изображений в Colab.

Важно для отчёта: Обязательно зафиксируйте версию OpenCV в вашем отчёте. Обычно это что-то вроде 4.x.x. Эта информация важна для воспроизводимости результатов, так как поведение некоторых функций может меняться между версиями.

Если при выполнении импорта возникает ошибка `ModuleNotFoundError`, необходимо установить OpenCV командой `!pip install opencv-python` в отдельной ячейке перед импортами. В Google Colab обычно OpenCV уже предустановлен.

Два подхода к получению входных данных

В данной лабораторной работе допускается использование двух различных подходов к получению исходных изображений для обработки. Каждый имеет свои преимущества и области применения.

Подход А: Загрузка файла

Преимущества: работа с реальными изображениями, наглядная демонстрация эффектов обработки на «живых» фотографиях, простота получения разнообразного визуального контента.

Применение: удобен для демонстрации эффектов преобразований (resize, flip, rotate, cvtColor) на узнаваемых объектах.

Подход Б: Генерация canvas

Преимущества: полный контроль над каждым пикселем, гарантированная уникальность работы, глубокое понимание структуры данных изображения.

Применение: методически ценен, так как связывает теорию («изображение — массив чисел») с практикой. Идеален для параметризации по вариантам.

Требование к работе: Минимально необходимо продемонстрировать оба подхода — хотя бы один раз прочитать изображение из файла и хотя бы один раз создать собственный canvas с рисованием примитивов. Это обеспечит полноту освоения материала.

Загрузка файла в Google Colab программным способом

Google Colab предоставляет удобный API для загрузки файлов непосредственно из интерфейса браузера. Этот метод не требует настройки путей к файлам или подключения Google Drive.

```
from google.colab import files

uploaded = files.upload()  # откроется диалог выбора файла
fname = list(uploaded.keys())[0]
print("Uploaded:", fname)
```

1

Импорт модуля

Модуль google.colab.files предоставляет функции для работы с файлами в облачной среде.

2

Диалог загрузки

files.upload() открывает диалог выбора файла, возвращает словарь с именами и содержимым загруженных файлов.

3

Получение имени

Файл помещается в текущую рабочую директорию, имя извлекается из ключей словаря.

Альтернативные подходы

Если вам нужно работать с несколькими файлами или большими наборами данных, можно:

- Подключить Google Drive через `drive.mount()`
- Загрузить данные по URL с помощью `wget` или `requests`
- Использовать datasets из библиотек (например, OpenCV samples)

Для отчёта

В отчёте достаточно указать:

- «Изображение загружено через `files.upload()`»
- Название файла и его исходные параметры (размер, формат)
- Если используете несколько файлов — перечислите их имена

Для базовой лабораторной работы метод `files.upload()` является оптимальным — он прост, не требует дополнительной настройки и работает надёжно для файлов размером до нескольких мегабайт.

Чтение изображения: cv2.imread и обработка ошибок

Функция `cv2.imread()` — основной способ чтения изображений с диска. Критически важно понимать её поведение при ошибках и корректно обрабатывать возможные проблемы.

```
img = cv2.imread(fname, cv2.IMREAD_COLOR)

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

1

Флаг чтения

`cv2.IMREAD_COLOR` — читает изображение как цветное (3 канала BGR). Существуют альтернативы: IMREAD_GRAYSCALE (1 канал), IMREAD_UNCHANGED (сохраняет альфа-канал если есть).

2

Проверка None

imread **не вызывает исключение** при ошибке — она молча возвращает None. Это типичная ловушка: если не проверить, все последующие операции упадут с непонятными ошибками.

3

Возбуждение исключения

Явное возбуждение исключения с информативным сообщением помогает быстро локализовать проблему. Это хорошая практика программирования.

Типичные причины возврата None:

- Файл не существует по указанному пути (опечатка в имени, файл не загружен)
- Формат файла не поддерживается OpenCV (редкие форматы изображений)
- Файл повреждён или имеет некорректную структуру
- Недостаточно прав доступа для чтения файла

В каждом отчёте по лабораторной работе должна присутствовать проверка результата imread. Это демонстрирует понимание особенностей API OpenCV и культуру написания надёжного кода. Grayscale-преобразование лучше делать отдельной операцией через cvtColor, чтобы осознанно контролировать переход от 3 каналов к 1 каналу.

Единственный способ отображения: cv2_imshow

Категорически запрещено

```
# НЕ используйте в этой работе:
# plt.imshow(img)
# cv2.imshow('window', img)
```

Matplotlib требует конвертации BGR→RGB и создаёт путаницу для начинающих. Оконная функция cv2.imshow не работает в Colab.

Единственно правильный способ

```
cv2_imshow(img)
```

Функция `cv2_imshow` специально адаптирована для Google Colab. Она:

- Автоматически обрабатывает формат BGR
- Не требует указания имени окна
- Корректно отображает как цветные, так и grayscale изображения
- Работает внутри ячеек ноутбука без дополнительной настройки

Для отчёта: Каждый вызов cv2_imshow в ноутбуке должен соответствовать рисунку в отчёте. Под каждым рисунком должна быть подпись вида «Рисунок N — [описание]». В тексте отчёта обязательно должна быть ссылка на этот рисунок, например: «Результат представлен на рисунке 3».

Это правило фиксируется для всей лабораторной работы и обеспечивает единообразие отчётов, устранение ошибок с перепутыванием каналов и упрощение проверки работ преподавателем. Даже если вы знакомы с matplotlib и предпочитаете его для визуализации, в контексте данной лабораторной работы использование других методов вывода не допускается.

Изображение как массив NumPy: shape и dtype

Понимание структуры данных изображения — ключевой момент перехода от «картинки на экране» к «массиву чисел в памяти». Это фундамент всей компьютерной обработки изображений.

```
print("shape:", img.shape)
print("dtype:", img.dtype)
```

Атрибут shape

Кортеж вида `(H, W, C)`: высота в пикселях, ширина в пикселях, число каналов. Для цветного изображения C=3 (BGR), для grayscale: `(H, W)` — двумерный массив.

Атрибут dtype

Тип данных элементов массива. Обычно `uint8` — беззнаковое 8-битное целое, диапазон 0..255. Это стандартное представление интенсивности пикселей.

Индексация

Доступ к пикселям: `img[y, x]` или `img[y, x, c]`. Важно: сначала строка (y), потом столбец (x), это математическая индексация матриц.

Порядок каналов

В OpenCV: **BGR**, не RGB. Первый канал (индекс 0) — Blue, второй (индекс 1) — Green, третий (индекс 2) — Red. Это историческая особенность OpenCV.

В отчёте обязательно приводите значения shape и dtype для исходного изображения и для всех ключевых промежуточных результатов. Это демонстрирует понимание того, как преобразования влияют на структуру данных. Например, переход к grayscale меняет shape с (480, 640, 3) на (480, 640), а операции resize меняют первые две размерности.

Доступ к отдельным пикселям и порядок каналов BGR

Изображение в OpenCV — это обычный массив NumPy, и к нему применимы все стандартные операции индексирования. Понимание порядка индексов и каналов критически важно для корректной обработки.

```
y, x = 100, 200
px = img[y, x]
print("pixel(B,G,R) =", px)

# Для цветного изображения:
# px[0] — синий канал
# px[1] — зелёный канал
# px[2] — красный канал
```

Ключевые моменты

- **Порядок индексов:** сначала y (номер строки, координата по вертикали), затем x (номер столбца, координата по горизонтали)
- **Порядок каналов BGR:** исторически OpenCV использует BGR вместо более привычного RGB из-за особенностей ранних версий библиотеки
- **Результат:** для цветного изображения px — массив из трёх чисел [B, G, R], для grayscale — одно число

Канал B (Blue)

`img[y, x, 0]` Значение синего компонента в диапазоне 0..255. Чем больше значение, тем интенсивнее синий цвет в данном пикселе.

Канал G (Green)

`img[y, x, 1]` Значение зелёного компонента. Для изображений с растительностью этот канал обычно имеет высокие значения.

Канал R (Red)

`img[y, x, 2]` Значение красного компонента. Например, для закатного неба или красных объектов этот канал доминирует.

В отчёте достаточно привести один пример обращения к конкретному пикселю и его значения, указав координаты и интерпретацию каналов. Это демонстрирует понимание того, что «изображение — это не магия, а конкретные числа в определённом порядке».

Преобразование в оттенки серого: cvtColor BGR→GRAY

Преобразование в grayscale (оттенки серого) — одна из фундаментальных операций предобработки. Она упрощает данные, снижает вычислительную сложность и часто является первым шагом в алгоритмах выделения границ, детекции признаков и распознавания.

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
print("gray shape:", gray.shape)
cv2_imshow(gray)
```

Исходное изображение

Трёхмерный массив shape=(H, W, 3), каждый пиксель представлен тремя значениями интенсивности в каналах B, G, R.

1

2

Преобразование

Функция cvtColor применяет взвешенную формулу: Gray = 0.299·R + 0.587·G + 0.114·B (приблизительно). Эти коэффициенты учитывают восприятие яркости человеческим глазом.

3

Результат

Двумерный массив shape=(H, W), каждый пиксель — одно число от 0 (чёрный) до 255 (белый), представляющее яркость.

Что происходит с данными

При переходе от цветного изображения к grayscale происходит не просто «удаление цвета». Алгоритм вычисляет яркость каждого пикселя как взвешенную сумму трёх каналов. Веса выбраны так, чтобы имитировать чувствительность человеческого глаза: мы наиболее чувствительны к зелёному (вес ~0.59), менее к красному (~0.30) и ещё менее к синему (~0.11).

Для отчёта

В отчёте обязательно:

- Показать исходное цветное изображение и результат grayscale рядом
- Указать изменение shape: с (H, W, 3) на (H, W)
- Кратко проанализировать: какие области стали светлее/темнее и почему (например, синее небо стало тёмным из-за малого веса синего канала)

Важно понимать: grayscale — это не «чёрно-белая картинка в трёх одинаковых каналах», а именно одноканальное представление. Это принципиально разные структуры данных, что отражается в shape массива.

Сохранение результатов: cv2.imwrite

Сохранение обработанных изображений необходимо для формирования отчёта и обеспечения воспроизводимости результатов. Функция `cv2.imwrite()` записывает массив изображения на диск в различных форматах.

```
ok = cv2.imwrite("out_gray.png", gray)
print("saved:", ok)

# Примеры говорящих имён файлов:
cv2.imwrite("out_resize_05.png", small)
cv2.imwrite("out_flip_horizontal.png", flip_h)
cv2.imwrite("out_rotate_90cw.png", rot)
```

Формат файла

Определяется автоматически по расширению. Поддерживаются PNG, JPEG, TIFF, BMP и другие. PNG рекомендуется для точного сохранения без потерь.

Имена файлов

Используйте говоряшие имена: out_gray, out_resize_05, out_flip_h. Это упрошает организацию результатов и понимание того, что в файле.

Проверка успеха

imwrite возвращает True при успешной записи, False при ошибке. Всегда проверяйте результат — это культура надёжного кода.

Почему важно сохранять

- Результаты станут иллюстрациями в отчёте
- Независимость от среды выполнения: можно закрыть ноутбук и вернуться к файлам
- Возможность сравнения результатов разных экспериментов

Типичные ошибки

- Перезапись файлов с одинаковыми именами
- Некорректный путь (папка не сушествует)
- Нет проверки возвращаемого значения
- Сохранение в форматы с потерями (JPEG) для промежуточных результатов

Для отчёта: Составьте перечень всех сохранённых файлов-результатов с кратким описанием каждого. Например: «out_gray.png — результат преобразования в оттенки серого, shape=(480, 640)». Этот перечень должен быть в разделе «Результаты» или «Приложения».

Масштабирование изображений: операция resize

Изменение размера изображения — базовая операция, используемая для унификации входных данных, экономии памяти или подготовки к обучению нейросетевых моделей. Масштабирование влияет не только на размер, но и на детальность и качество изображения.

```
small = cv2.resize(img, (0,0), fx=0.5, fy=0.5)
big = cv2.resize(img, (0,0), fx=1.5, fy=1.5)

cv2_imshow(small)
cv2_imshow(big)
```

Уменьшение (fx<1, fy<1)

При уменьшении происходит **потеря информации** — мелкие детали просто исчезают, так как несколько пикселей «сжимаются» в один. Это необратимая операция: увеличив потом маленькое изображение, мы не восстановим утраченные детали.

Уменьшение полезно для:

- Ускорения обработки (меньше данных)
- Экономии памяти
- Создания пирамид изображений
- Подготовки к нейросетям с фиксированным входом

Увеличение (fx>1, fy>1)

При увеличении приходится **«придумывать» новые пиксели** между сушествующими. Это делается с помошью интерполяции — математического алгоритма вычисления промежуточных значений. Результат — размытие или ступенчатость (артефакты интерполяции).

Увеличение применяется для:

- Визуализации мелких деталей
- Подготовки данных для печати
- Согласования размеров разных изображений

Для вариативности: Каждый студент должен использовать свои значения коэффициентов fx и fy согласно номеру варианта. Например: fx = 0.3 + 0.01·variant, fy = 0.3 + 0.005·variant. Это гарантирует уникальность результатов при обшем алгоритме обработки.

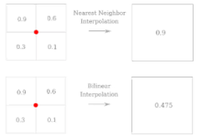
В отчёте необходимо показать два рисунка: уменьшенное и увеличенное изображение, привести их shape и дать краткое сравнение визуального качества. Укажите, где наблюдается потеря деталей (при уменьшении) и где появляются артефакты интерполяции (при увеличении).

Интерполяция при масштабировании: выбор метода

Когда мы изменяем размер изображения, OpenCV должен вычислить значения новых пикселей на основе сушествуюших. Способ такого вычисления называется **интерполяцией**. Выбор метода интерполяции влияет на качество и скорость обработки.

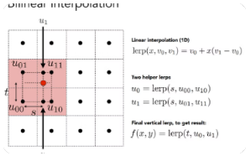
```
small_area = cv2.resize(img, (0,0), fx=0.4, fy=0.4,
                        interpolation=cv2.INTER_AREA)

big_cubic = cv2.resize(img, (0,0), fx=2.0, fy=2.0,
                       interpolation=cv2.INTER_CUBIC)
```



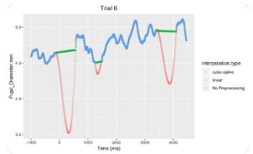
INTER_NEAREST

Ближайший сосед — самый быстрый, но самый низкокачественный метод. Просто копирует значение ближайшего пикселя. Даёт «пиксельный» эффект при увеличении.



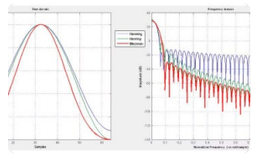
INTER_LINEAR

Билинейная интерполяция — стандартный метод, хороший компромисс между скоростью и качеством. Усредняет 4 ближайших пикселя. Подходит для большинства задач.



INTER_CUBIC

Бикубическая интерполяция — более качественный, но медленный метод. Учитывает 16 соседних пикселей. Даёт более гладкие переходы при увеличении, лучше для высококачественной обработки.



INTER_AREA

Метод усреднения по площади — **рекомендуется для уменьшения**. Усредняет все пиксели, которые попадают в новый пиксель, что даёт наилучшее качество при downsampling.

Рекомендации по выбору

- **Для уменьшения:** используйте INTER_AREA — он даёт наилучшее качество, минимизирует эффект «ступенек» и муара
- **Для увеличения:** INTER_LINEAR для обычных задач, INTER_CUBIC для более гладких результатов
- **Для пиксель-арта:** INTER_NEAREST, чтобы сохранить чёткие границы

Для расширенного задания

Демонстрация разных методов интерполяции — хороший элемент расширенной части работы. Покажите:

- Одно и то же увеличение с LINEAR и CUBIC
- Визуальное сравнение результатов
- Краткий анализ: где заметна разница (обычно на границах объектов и диагональных линиях)

Отражение изображений: операция flip

Отражение (flipping) — геометрическое преобразование, которое зеркально переворачивает изображение относительно горизонтальной или вертикальной оси. Это простая, но полезная операция для аугментации данных и исправления ориентации.

```
flip_h = cv2.flip(img, 1)   # отражение по горизонтали
flip_v = cv2.flip(img, 0)   # отражение по вертикали
flip_hv = cv2.flip(img, -1) # отражение по обеим осям
```

flipCode = 1: горизонтальное

Отражение слева направо, как в зеркале. Левая часть изображения становится правой и наоборот. Координата x инвертируется: new_x = width - 1 - x. Часто используется для аугментации датасетов: удваивает количество обучающих примеров.

flipCode = 0: вертикальное

Отражение сверху вниз. Верхняя часть изображения становится нижней. Координата y инвертируется: new_y = height - 1 - y. Реже используется на практике, но полезно для исправления перевёрнутых изображений.

flipCode = -1: обе оси

Эквивалентно повороту на 180°. Инвертируются обе координаты. Применяется для исправления изображений, снятых «вверх ногами», или как специфический вариант аугментации данных.

Важная особенность: Операция flip не изменяет shape изображения — размеры остаются теми же. Изменяется только порядок пикселей в массиве. Это отличает flip от rotate на произвольный угол, где могут изменяться размеры.

В отчёте достаточно показать один вид отражения согласно вашему варианту. Укажите, какой flipCode использовался, продемонстрируйте результат и подтвердите, что shape не изменился. Кратко опишите визуальный эффект — как изменилась композиция изображения.

Поворот изображения: cv2.rotate для кратных 90°

Для поворота на углы, кратные 90°, OpenCV предоставляет специальную функцию `cv2.rotate()`. Это значительно проще и быстрее, чем использование аффинных преобразований, и не требует вычисления матриц поворота.

```
r90 = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
r180 = cv2.rotate(img, cv2.ROTATE_180)
r270 = cv2.rotate(img, cv2.ROTATE_90_COUNTERCLOCKWISE)
```

ROTATE_90_CLOCKWISE

Поворот на 90° по часовой стрелке. Важно: ширина и высота меняются местами! Если было (480, 640, 3), станет (640, 480, 3).

ROTATE_180

Поворот на 180°. Эквивалентно flip с кодом -1. Shape не изменяется, только порядок пикселей инвертируется по обеим осям.

ROTATE_90_COUNTERCLOCKWISE

Поворот на 90° против часовой стрелки (или на 270° по часовой). Также меняет ширину и высоту местами.

Изменение размерностей

Ключевое отличие от flip: при повороте на 90° или 270° изображение меняет ориентацию с portrait на landscape или наоборот. Это означает, что первые две компоненты shape меняются местами:

- Было: (H, W, 3)
- Стало: (W, H, 3)

При повороте на 180° размеры остаются прежними.

Для отчёта

Выберите один вариант поворота согласно вашему варианту. В отчёте:

- Укажите используемую константу (например, ROTATE_90_CLOCKWISE)
- Покажите результат визуально
- Зафиксируйте изменение shape: «было (480, 640, 3), стало (640, 480, 3)»
- Опишите практическую пользу: исправление ориентации фотографий с камеры

Пример сквозного конвейера обработки

Соединим несколько базовых операций в единый pipeline (конвейер обработки). Это демонстрирует понимание того, что каждая операция возвращает новый массив, который становится входом для следующей операции.

```
img = cv2.imread(fname)
small = cv2.resize(img, (0,0), fx=0.6, fy=0.6)
gray = cv2.cvtColor(small, cv2.COLOR_BGR2GRAY)
rot = cv2.rotate(gray, cv2.ROTATE_90_CLOCKWISE)

cv2.imwrite("out_pipeline.png", rot)
cv2_imshow(rot)
```

Этот минимальный конвейер демонстрирует типичную последовательность предобработки изображения. Давайте разберём, что происходит на каждом шаге:

Шаг 1: Чтение

imread загружает изображение с диска в память как массив NumPy. Предположим, исходный размер 800×1200 пикселей, 3 канала BGR.

Шаг 2: Масштабирование

resize уменьшает изображение на 40% по обеим осям. Новый размер: 480×720 пикселей, всё ещё 3 канала. Цель: экономия памяти и ускорение дальнейшей обработки.

Шаг 3: Grayscale

cvtColor преобразует RGB в оттенки серого. Размер: 480×720, но теперь только 1 канал. Shape меняется с (480, 720, 3) на (480, 720).

Шаг 4: Поворот

rotate поворачивает на 90° по часовой. Размер меняется: 720×480 (ширина и высота поменялись местами). Результат сохраняется в файл.

Такой пример удобно включить в отчёт как отдельный раздел «Сквозной пример обработки». Он демонстрирует понимание композиции операций, отслеживание изменений shape и dtype на каждом этапе, а также навык построения многошагового алгоритма обработки.

Генерация изображений: создание canvas в NumPy

Создание изображения «с нуля» программным способом — важный навык для понимания структуры данных и обеспечения уникальности лабораторных работ. Мы создаём пустой массив нужного размера и заполняем его значениями.

```
canvas = np.full((300, 400, 3), 255, dtype=np.uint8)  # белый фон
cv2_imshow(canvas)
```

Размеры

`(300, 400, 3)` означает: высота 300 пикселей, ширина 400 пикселей, 3 канала (BGR). Размеры выбираются по варианту для обеспечения уникальности.

Заполнение

Значение 255 означает белый цвет во всех каналах. Для чёрного фона используйте 0. Можно создать цветной фон: `np.full((H,W,3), [255,0,0], dtype=np.uint8)` — красный.

Тип данных

`dtype=np.uint8` — критически важно! OpenCV ожидает именно этот тип для изображений. Без указания dtype NumPy может создать массив float или int, что приведёт к ошибкам.

Альтернативные способы создания

```
# Чёрный фон
canvas = np.zeros((H, W, 3), dtype=np.uint8)

# Случайный шум
canvas = np.random.randint(0, 256, (H, W, 3), dtype=np.uint8)

# Градиент
canvas = np.tile(np.arange(0, 256, dtype=np.uint8), (H, W, 1))
```

Связь с теорией

Генерация canvas — прямая реализация концепции «изображение = массив чисел»:

- Мы создаём матрицу конкретного размера
- Заполняем её числовыми значениями 0..255
- OpenCV интерпретирует эти числа как цвета пикселей
- Результат — изображение, которое мы полностью контролируем

Генерация собственного изображения — обязательная часть лабораторной работы. Она демонстрирует понимание низкоуровневой природы изображений и позволяет обеспечить уникальность через параметризацию по варианту.

Рисование примитивов: line, rectangle, circle

OpenCV предоставляет удобные функции для рисования геометрических примитивов непосредственно на массиве изображения. Это полезно для аннотации, визуализации результатов алгоритмов и создания тестовых изображений.

```
cv2.line(canvas, (10,10), (390,290), (0,0,255), 3)
cv2.rectangle(canvas, (50,200), (200,280), (0,255,0), -1)
cv2.circle(canvas, (300,160), 40, (255,0,0), 2)

cv2_imshow(canvas)
```

Линия (line)

Параметры: начальная точка (x1, y1), конечная точка (x2, y2), цвет BGR, толщина в пикселях. Рисует прямую линию между двумя точками.

Прямоугольник (rectangle)

Параметры: верхний левый угол, нижний правый угол, цвет BGR, толщина. Толщина -1 означает заливку. Полезен для выделения областей интереса.

Круг (circle)

Параметры: центр (x, y), радиус, цвет BGR, толщина. Толщина -1 даёт залитый круг. Используется для маркировки точек или создания масок.

Система координат

Важно понимать систему координат OpenCV:

- **Начало координат:** левый верхний угол изображения
- **Ось X:** направлена вправо (увеличение столбца)
- **Ось Y:** направлена вниз (увеличение строки)
- **Точка (0, 0):** самый верхний левый пиксель

Формат цвета BGR

Цвета задаются как кортежи (B, G, R):

- `(0, 0, 255)` — красный
- `(0, 255, 0)` — зелёный
- `(255, 0, 0)` — синий
- `(0, 0, 0)` — чёрный
- `(255, 255, 255)` — белый

В отчёте покажите итоговый canvas с несколькими примитивами и перечислите использованные функции с параметрами. Для обеспечения вариативности используйте координаты, размеры и цвета, зависящие от номера варианта.

Добавление текста: функция putText

Текст на изображении — удобный способ маркировки, аннотации и добавления метаинформации. В контексте лабораторной работы текст используется для указания номера варианта и уникальной идентификации работы.

```
cv2.putText(canvas, "Variant 12", (20,50),
            cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,0,0), 2)
```

01

Изображение

Первый параметр — массив изображения, на котором будет рисоваться текст. Функция изменяет массив «на месте» (in-place).

02

Содержание

Строка текста для отображения. Может включать буквы, цифры, символы. Для кириллицы могут потребоваться дополнительные настройки.

03

Позиция

Кортеж (x, y) — координаты левого нижнего угла первого символа. Важно: не верхнего левого, а нижнего левого!

04

Шрифт

OpenCV предоставляет встроенные шрифты семейства Hershey. FONT_HERSHEY_SIMPLEX — стандартный читаемый шрифт.

05

Масштаб

Коэффициент масштабирования шрифта. 1.0 — базовый размер, 2.0 — в два раза больше, 0.5 — в два раза меньше.

06

Цвет

Кортеж BGR для цвета текста. (0,0,0) — чёрный, хорошо виден на светлом фоне.

07

Толщина

Толщина линий шрифта в пикселях. 2 — стандартная читаемая толщина, 1 — тонкий текст, 3+ — жирный.

In-place изменение: Функция putText, как и другие функции рисования, изменяет переданный массив напрямую. Это означает, что исходный canvas будет модифицирован. Если нужно сохранить оригинал — сделайте копию перед рисованием: `canvas_copy = canvas.copy()`

В отчёте укажите, что текст наносится поверх изображения, изменяя значения пикселей в соответствующей области. Это важный момент понимания: все операции рисования — это просто изменение чисел в массиве.

Уникализация работ: параметризация по варианту

Чтобы гарантировать индивидуальность каждой работы при обшем алгоритме обработки, все ключевые параметры должны зависеть от номера варианта студента. Это обеспечивает честность оценивания и предотвращает копирование.

```
variant = 12  # ваш номер варианта

# Параметризация размеров canvas
h = 240 + 5 * variant
w = 320 + 3 * variant

# Создание уникального canvas
canvas = np.full((h, w, 3), 255, dtype=np.uint8)
```

1 **Определение варианта**

Номер варианта определяется однозначно: по последним двум цифрам зачётной книжки, по номеру в списке группы, или по другому правилу, установленному преподавателем. Главное — детерминированность.

2 **Параметризация размеров**

Размеры canvas, координаты фигур, радиусы кругов вычисляются по формулам от variant. Например: height = 240 + 5·v, width = 320 + 3·v, radius = 20 + 2·v.

3 **Параметризация преобразований**

Коэффициенты масштабирования, выбор направления flip/rotate, значения alpha/beta также задаются формулами или правилами выбора на основе variant.

4 **Фиксация в отчёте**

В начале отчёта обязательно указывается: «Вариант № X», далее приводятся все вычисленные параметры. Это позволяет преподавателю легко проверить корректность.

Что нельзя делать

- Выбирать параметры вручную после просмотра результата
- Подгонять параметры под «красивую картинку»
- Использовать одинаковые параметры у всех студентов
- Не фиксировать вариант в отчёте

Правильный подход

- Вариант определяется до начала работы
- Все параметры вычисляются по формулам
- Результаты не подгоняются — они какие есть
- Вариант и параметры явно указаны в отчёте

Разделение на каналы: функция split

Операция split разделяет многоканальное изображение на отдельные одноканальные массивы. Это позволяет визуализировать и анализировать вклад каждого цветового канала в итоговое изображение.

```
b, g, r = cv2.split(img)

cv2_imshow(b)  # синий канал
cv2_imshow(g)  # зелёный канал
cv2_imshow(r)  # красный канал
```

Канал B (Blue)

Первый канал BGR-изображения. Массив shape=(H, W) с яркостями синего компонента. На этом изображении области с большим содержанием синего будут светлыми.

Канал G (Green)

Второй канал. Для изображений с растительностью или зелёными объектами этот канал будет ярче остальных. Человеческий глаз наиболее чувствителен к зелёному.

Канал R (Red)

Третий канал. Будет ярче на красных объектах: закаты, цветы, огонь. Для изображений с синим небом красный канал обычно тёмный.

Интерпретация результата

Каждый канал отображается как изображение в оттенках серого, где:

- **Светлые области** — высокое значение данного цветового компонента (близко к 255)
- **Тёмные области** — низкое значение (близко к 0)
- **Средние тона** — промежуточные значения

Это не означает, что изображение стало чёрно-белым — мы просто смотрим на один цветовой компонент отдельно.

Для отчёта

В отчёте представьте:

- Исходное цветное изображение
- Три рисунка каналов B, G, R
- Краткий анализ: где какой канал ярче и почему

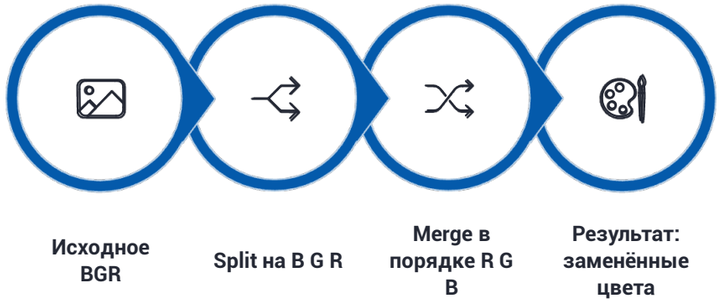
Например: «На изображении зелёной травы канал G значительно светлее каналов B и R, что соответствует высокому содержанию зелёного цвета в сцене».

Это упражнение на понимание: «канал» — не абстрактное понятие, а конкретный массив чисел. Каждый пиксель цветного изображения — это три независимых значения интенсивности.

Сборка и перестановка каналов: функция merge

Если split разделяет изображение на каналы, то merge выполняет обратную операцию — собирает несколько одноканальных массивов в одно многоканальное изображение. Это позволяет экспериментировать с порядком каналов и наблюдать эффекты.

```
img_swap = cv2.merge([r, g, b])  # меняем местами B и R
cv2_imshow(img_swap)
```



Что происходит при перестановке

Нормальный порядок в OpenCV: [B, G, R]. Если мы передадим в merge порядок [R, G, B], то:

- Канал R попадёт на место B (индекс 0)
- Канал G останется на месте (индекс 1)
- Канал B попадёт на место R (индекс 2)

Результат: красные области станут синими, синие — красными, зелёные останутся зелёными. Получится «инвертированное» по красно-синей оси изображение.

Практическая польза

Этот эксперимент наглядно демонстрирует:

- Структуру хранения цветовой информации
- Влияние порядка каналов на восприятие цвета
- Почему важно различать BGR и RGB
- Что изображение — просто числа, которыми можно манипулировать

Для отчёта: Покажите исходное изображение и результат перестановки каналов рядом. Опишите, какие цвета изменились и почему. Например: «Красная роза стала синей, синее небо стало красноватым, зелёная трава осталась зелёной». Это демонстрирует понимание механизма работы с каналами.

Операции split и merge — инструменты для глубокого понимания цветовых представлений. Они используются не только в учебных целях, но и в практических задачах: коррекция цвета, анализ отдельных каналов, создание специальных эффектов.

Изменение яркости: безопасный способ без переполнения

Изменение яркости — базовая операция предобработки. Однако прямое прибавление константы к массиву uint8 может привести к переполнению (overflow), когда значения выходят за пределы 0..255. OpenCV предоставляет безопасные функции для таких операций.

```
brighter = cv2.convertScaleAbs(img, alpha=1.0, beta=40)  # +40 яркости
darker = cv2.convertScaleAbs(img, alpha=1.0, beta=-40)     # -40 яркости

cv2_imshow(brighter)
```

Проблема прямого сложения

Операция `img + 40` на массиве uint8 приведёт к переполнению: если пиксель имел значение 230, после прибавления 40 получится 270, что превысит максимум 255 и «обернётся» к малым значениям (модуль 256). Результат — артефакты.

Функция convertScaleAbs

Выполняет операцию `result = saturate(alpha * img + beta)`, где saturate обрезает значения в диапазоне [0, 255]. При alpha=1.0 это простой сдвиг яркости с корректной обработкой границ.

Параметр beta

Значение, добавляемое к каждому пикселю. Положительное beta увеличивает яркость, отрицательное — уменьшает. Типичные значения: ±20..±50.

Визуальные эффекты

Увеличение яркости (beta>0):

- Светлые области становятся белыми (насыщение на 255)
- Тёмные области становятся видимыми
- Возможна потеря деталей в светах

Уменьшение яркости (beta<0):

- Тёмные области становятся чёрными (насыщение на 0)
- Светлые области становятся средними тонами
- Возможна потеря деталей в тенях

Для расширенного задания

Демонстрация изменения яркости — хороший элемент расширенной части работы. Покажите:

- Два результата: увеличение и уменьшение яркости
- Визуальное сравнение с исходным
- Анализ: где «выбились» области (стали полностью белыми или чёрными)
- Вывод о потере информации при насыщении

Изменение контраста: параметр alpha

Контраст определяет разницу между светлыми и тёмными областями изображения. Высокий контраст делает изображение более «выразительным», низкий — более «плоским». Функция convertScaleAbs позволяет управлять контрастом через параметр alpha.

```
high_contrast = cv2.convertScaleAbs(img, alpha=1.5, beta=0)
low_contrast = cv2.convertScaleAbs(img, alpha=0.7, beta=0)
```

Высокий контраст (alpha > 1)

При alpha=1.5 каждое значение пикселя умножается на 1.5. Тёмные области (малые значения) становятся ешё темнее, светлые (большие значения) — ешё светлее, но с насыщением на 255. Результат: более драматичное изображение с усиленными границами.

Низкий контраст (alpha < 1)

При alpha=0.7 значения «сжимаются» к середине диапазона. Тёмные области становятся светлее, светлые — темнее. Результат: «плоское», размытое по яркости изображение с потерей выразительности.

Математика операции

Формула: `out = saturate(alpha * in + beta)`

При beta=0 (изменение только контраста):

- Пиксель со значением 100 и alpha=1.5 станет 150
- Пиксель со значением 200 и alpha=1.5 станет 300 → 255 (насыщение)
- Пиксель со значением 100 и alpha=0.7 станет 70

Точка «без изменений» — alpha=1.0.

Где пропадают детали

При увеличении контраста: детали теряются в очень тёмных (→0) и очень светлых (→255) областях из-за насыщения.

При уменьшении контраста: вся шкала яркостей сжимается, мелкие различия между близкими тонами исчезают.

Это иллюстрирует компромисс: выразительность vs. сохранение информации.

Для отчёта: Покажите два результата (высокий и низкий контраст) и проанализируйте: где изображение стало более выразительным, а где потеряло детали. Это демонстрирует понимание баланса между визуальным эффектом и потерей информации.

Опциональное расширение: размытие GaussianBlur

Хотя фильтрация изображений — тема следующей лабораторной работы, демонстрация простого размытия полезна как пример «силы OpenCV» и подготовка к более сложным алгоритмам.

```
blur = cv2.GaussianBlur(img, (5,5), 0)
cv2_imshow(blur)
```

1

Исходное изображение

Содержит мелкие детали, текстуры, шум. Каждый пиксель имеет уникальное значение, возможны резкие перепады яркости.

2

Гауссово размытие

Каждый пиксель заменяется взвешенным средним соседних пикселей по гауссовой функции. Чем дальше сосед, тем меньше его вес.

3

Результат

Сглаженное изображение без мелких деталей и шума. Границы объектов остаются видимыми, но менее чёткими.

Параметры функции

- **ksize=(5,5):** размер ядра свёртки. Должен быть нечётным. Большее ядро — сильнее размытие.
- **sigmaX=0:** стандартное отклонение по X. При 0 вычисляется автоматически из ksize.
- Существует также sigmaY, но обычно он равен sigmaX для симметричного размытия.

Практическое применение

- Удаление шума перед обработкой
- Подготовка к выделению границ
- Создание эффекта «мягкого фокуса»
- Построение пирамид изображений

В отчёте можно включить размытие как дополнительный эксперимент в разделе расширенных заданий. Покажите исходное и размытое изображение, опишите, что произошло с мелкими структурами (текстуры, шум). Укажите, что более детальное изучение фильтров — тема следующей лабораторной работы.

Минимальный набор обязательных результатов

Для успешной сдачи лабораторной работы отчёт должен содержать полный набор результатов, демонстрирующих освоение всех базовых операций. Этот чек-лист поможет убедиться, что ничего не упущено.

01

Исходное цветное изображение

Загруженное из файла или сгенерированное. Указать shape, dtype, источник.

02

Grayscale преобразование

Результат cvtColor BGR→GRAY, демонстрация изменения shape с 3 каналов на 1.

03

Уменьшение (resize)

Изображение с fx<1, fy<1. Анализ потери деталей.

04

Увеличение (resize)

Изображение с fx>1, fy>1. Наблюдение артефактов интерполяции.

05

Flip или Rotate

Одно из геометрических преобразований по варианту: отражение или поворот.

06

Canvas с примитивами

Сгенерированное изображение с нарисованными линиями, прямоугольниками, кругами и текстом варианта.

07

Работа с каналами

Либо три рисунка разделённых каналов (split), либо результат перестановки каналов (merge/swap).

08

Расширенное задание

Минимум одно из: изменение яркости, изменение контраста, размытие, эксперимент с интерполяцией.

Проверка перед сдачей: Пройдитесь по чек-листу и убедитесь, что каждый пункт представлен в отчёте как минимум одним рисунком с подписью и сопроводительным текстом. Если хотя бы одной позиции нет — работа считается неполной.

Все рисунки должны иметь аккуратные подписи формата «Рисунок N — описание» и явные ссылки в тексте отчёта. В тексте должны быть объяснения: что показано, какие параметры использованы, какие эффекты наблюдаются.

Система вариантов: формализация параметров

Определение варианта

Variant = номер в списке группы.

Формулы параметров

Все ключевые числа вычисляются по формулам от variant: размеры canvas, координаты фигур, коэффициенты fx/fy, значения alpha/beta, выбор flip/rotate.

Запрет на подбор

Нельзя «подбирать» параметры после просмотра результата. Параметры фиксируются в начале работы и не меняются, даже если результат выглядит «не очень красиво».

Фиксация в отчёте

В начале раздела «Ход работы» обязательно указать:

- «Вариант № X»
- Формулы для всех параметров
- Вычисленные значения

Например: «fx = 0.3 + 0.01·12 = 0.42, fy = 0.3 + 0.005·12 = 0.36»

Пример таблицы параметров по вариантам

Для упрошения работы студентов и стандартизации результатов рекомендуется предоставить готовую таблицу или набор формул для вычисления всех необходимых параметров. Вот пример такой системы:

Масштабирование

fx = 0.3 + 0.01 · variant
fy = 0.3 + 0.005 · variant

Диапазон: fx от 0.3 до ~0.6, fy от 0.3 до ~0.45. Это обеспечивает заметное уменьшение с вариативностью.

Поворот

rotate_code = variant mod 3

- 0 → ROTATE_90_CLOCKWISE
- 1 → ROTATE_180
- 2 → ROTATE_90_COUNTERCLOCKWISE

Яркость

beta = (variant mod 5) · 20

Диапазон: 0, 20, 40, 60, 80. Обеспечивает разнообразие в изменении яркости без экстремальных значений.

Размер canvas

height = 240 + 5 · variant
width = 320 + 3 · variant

Даёт разные, но разумные размеры изображений для генерации.

Пример расчёта для варианта 12

- fx = 0.3 + 0.01·12 = 0.42
- fy = 0.3 + 0.005·12 = 0.36
- rotate_code = 12 mod 3 = 0 → 90° по часовой
- beta = (12 mod 5)·20 = 2·20 = 40
- height = 240 + 5·12 = 300
- width = 320 + 3·12 = 356

Использование в коде

```
variant = 12

# Вычисление параметров
fx = 0.3 + 0.01 * variant
fy = 0.3 + 0.005 * variant
rotate_code = variant % 3
beta = (variant % 5) * 20
h = 240 + 5 * variant
w = 320 + 3 * variant

# Применение
small = cv2.resize(img, (0,0), fx=fx, fy=fy)
```

Структура отчёта: стандартные разделы по ГОСТ

Технический отчёт по лабораторной работе — это связный научно-технический документ, следующий определённой структуре. Он не сводится к «скриншотам подряд», а представляет собой последовательное изложение цели, методов, результатов и выводов.

1. **Титульный лист**
   Название учебного заведения, кафедры, дисциплины, темы работы, ФИО студента и преподавателя, номер группы, дата.
2. **Цель работы**
   Краткая формулировка: освоить базовые операции работы с цифровыми изображениями в OpenCV.
3. **Теоретическая часть**
   Краткое описание основных понятий: что такое цифровое изображение, массив пикселей, shape, dtype, каналы BGR, операции преобразования (1-2 страницы).
4. **Ход работы**
   Описание последовательности действий с фрагментами кода, результатами (рисунками) и комментариями. Основная часть отчёта.
5. **Результаты**
   Систематизированное представление всех полученных изображений с подписями, таблицами параметров, сравнительным анализом.
6. **Выводы**
   Краткие итоги: что освоено, какие эффекты наблюдались, что понято про BGR, shape, интерполяцию и т.д.
7. **Список источников**
   Библиографические описания использованных материалов по ГОСТ: конспект, документация OpenCV.
8. **Приложения**
   Полный листинг кода, если он большой, или ссылка на ноутбук. Дополнительные материалы.

Каждый раздел должен быть оформлен как отдельный заголовок первого уровня. В «Ходе работы» удобно выделять подразделы для каждой операции: 3.1. Загрузка изображения, 3.2. Преобразование в grayscale и т.д. Не смешивайте код, рисунки и текст хаотично — структурируйте повествование логически.

Техническое задание: Цель, среда и ограничения

1. Цель работы

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

2. Среда выполнения: Google Colab (Python 3)

Разрешённые библиотеки:

- ☑ opencv-python (cv2)
- ☑ numpy
- ☑ google.colab.patches (cv2_imshow)
- ☑ google.colab.files

Отображение: строго через cv2_imshow()

ТЗ: Индивидуальный вариант и параметры

3. Вариант V = номер студента в списке группы (1…N)

Формулы параметров (использовать без ручного подбора):

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

Поворот

V mod 3 = 0 ⟶ 90° по часовой

V mod 3 = 1 ⟶ 180°

V mod 3 = 2 ⟶ 90° против часовой

Яркость

beta = (V mod 5) × 20 ⟶ результат: 0, 20, 40, 60 или 80

Размер canvas

H = 240 + 5×V

W = 320 + 3×V

⚠ В отчёте обязательно: указать V, привести вычисленные значения всех параметров, использовать их в коде

ТЗ: Структура ноутбука (пункты 5.1–5.8)

4. Исходные данные: одно цветное изображение (BGR, 3 канала) через files.upload() или сгенерированное

5. Обязательные разделы Colab-ноутбука:

01

5.1. Подготовка среды

импорт cv2, numpy, cv2_imshow; вывод версии OpenCV

02

5.2. Загрузка изображения

files.upload() → cv2.imread() с проверкой на None

03

5.3. Контроль свойств

вывести shape, dtype, значение пикселя img[y,x]; указать порядок BGR

04

5.4. Отображение

cv2_imshow(img)

05

5.5. Grayscale

cv2.cvtColor(BGR2GRAY), показать, сохранить out_gray_V.png

06

5.6. Масштабирование

уменьшить (fx_small, fy_small) и увеличить (fx_big, fy_big), сохранить out_resize_small_V.png, out_resize_big_V.png

07

5.7. Поворот

cv2.rotate по варианту (90°/180°/90°), сохранить out_rotate_V.png

08

5.8. Каналы

split на b,g,r; показать каналы; merge([r,g,b]) для swap, сохранить out_swap_channels_V.png

ТЗ: Генерация canvas и расширенная часть

6. Генерация уникального изображения (обязательно)

Создать canvas размера (H, W, 3) с белым фоном. Нарисовать минимум:

- 1 линию (cv2.line)
- 1 прямоугольник (cv2.rectangle)
- 1 окружность (cv2.circle)
- 1 текст (cv2.putText): "Variant V" и/или ФИО

Показать через cv2_imshow, сохранить out_canvas_V.png, цвета объектов на ваше усмотрение.

7. Расширенная часть (выбрать один вариант):

Вариант A: Яркость/контраст

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

Сохранить out_brightness_V.png

В отчёте объяснить насыщение 0..255 и преимущество convertScaleAbs

Вариант B: Размытие

- blur = cv2.GaussianBlur(img, (5,5), 0)

Сохранить out_blur_V.png

В отчёте указать эффект сглаживания

ТЗ: Именование файлов и порядок сдачи

- **9. Именование файлов результатов (обязательное)**

  Все файлы именовать с указанием варианта V (например, out_gray_12.png):

  - out_gray_V.png
  - out_resize_small_V.png
  - out_resize_big_V.png
  - out_rotate_V.png
  - out_swap_channels_V.png
  - out_canvas_V.png
  - out_brightness_V.png **или** out_blur_V.png

- **10. Порядок сдачи**

  Сдать в установленные преподавателем сроки:

  1. Отчёт (PDF)
  2. Ссылка на Colab-ноутбук в комментариях (режим доступа «просмотр»)
  3. При необходимости: архив с выходными изображениями

11. Работа считается выполненной, если:

Выполнены все обязательные пункты 5–7

Все разделы ноутбука (5.1–5.8), генерация canvas (п.6) и расширенная часть (п.7) реализованы полностью

Соблюдены ограничения по библиотекам

Использованы только разрешённые библиотеки (cv2, numpy, google.colab), вывод через cv2_imshow, без matplotlib/PIL/skimage

Отчёт оформлен корректно

Есть структура по ГОСТ, рисунки с подписями и ссылками в тексте, параметры варианта указаны и использованы, выводы содержательные

Результаты индивидуализированы

Вариант V отражён в параметрах (fx/fy/угол/beta/H/W), в canvas (текст "Variant V"), в именах файлов (out_*_V.png)

Требования к оформлению отчёта по ГОСТ

Оформление технического отчёта регламентируется стандартами ГОСТ и локальными нормами учебного заведения. Соблюдение этих требований — часть профессиональной культуры инженера.

Формат и поля

Формат бумаги: А4 (210×297 мм). Поля: левое 30 мм (для переплёта), правое 10 мм, верхнее и нижнее 20 мм. Это стандартные значения для большинства учебных работ.

Шрифт и кегль

Шрифт: Times New Roman (основной для технических документов). Кегль: 12 или 14 пт для основного текста (уточните на кафедре). Межстрочный интервал: 1.5.

Абзацы и выравнивание

Абзацный отступ: 1.25 см (стандартное значение Tab в Word). Выравнивание основного текста: по ширине. Заголовки: по левому краю или по центру (в зависимости от правил кафедры).

Нумерация и заголовки

Нумерация страниц: со второй страницы (после титульного листа), обычно в правом верхнем углу или по центру внизу. Заголовки разделов: жирным шрифтом, с нумерацией (1, 2, 3...), без точки в конце.

Дополнительные требования

- Переносы слов: включены автоматически
- Выделения: жирный для заголовков, курсив и код для специальных терминов
- Пунктуация: строго по правилам русского языка
- Сокращения: только общепринятые (например, т.е., т.д., и др.)

Что проверяется

- Единообразие оформления по всему документу
- Отсутствие «прыгающих» размеров шрифта
- Корректность межстрочных интервалов
- Правильность оформления заголовков

Оформление рисунков: нумерация, подписи, ссылки

Рисунки (изображения, графики, диаграммы) — обязательный компонент технического отчёта. Их оформление строго регламентировано и является одним из ключевых критериев оценки.

1

Формат подписи

Под каждым рисунком: «Рисунок N — описание». Например: «Рисунок 3 — Результат преобразования в оттенки серого». Подпись размещается под рисунком по центру.

2

Нумерация

Сквозная по всему отчёту: Рисунок 1, Рисунок 2 и т.д. Или по разделам: Рисунок 3.1, Рисунок 3.2 (если в методических указаниях требуется такая нумерация).

3

Ссылки в тексте

В тексте **обязательно** должна быть ссылка на каждый рисунок. Например: «Результат показан на рисунке 3» или «...как видно из рисунка 5...». Рисунок без ссылки фактически не засчитывается.

4

Качество и размер

Изображения должны быть читаемы. Не сжимайте до нечитаемого размера. Старайтесь поддерживать одинаковую ширину для визуальной согласованности (например, все рисунки на 80-90% ширины текстового блока).

Типичные ошибки

- Рисунок вставлен, но нет подписи
- Есть подпись, но в тексте нет ссылки
- Нумерация сбивается или пропущены номера
- Описание в подписи слишком общее («Изображение») вместо конкретного
- Разный размер рисунков без причины

Правильный подход

- Каждый вывод cv2_imshow → один рисунок в отчёте
- Все рисунки пронумерованы последовательно
- Подписи информативны и конкретны
- В тексте явно указано, что показано на рисунке
- Ширина рисунков примерно одинакова

Практический совет: При написании раздела «Ход работы» сначала опишите действие текстом, затем вставьте рисунок с подписью, затем сразу добавьте фразу со ссылкой на этот рисунок. Такой подход гарантирует, что рисунки и текст согласованы.

Оформление кода в отчёте: фрагменты и листинги

Код — важная часть отчёта, но его нужно включать разумно. Цель отчёта — объяснить ход работы, а не утопить читателя в сотнях строк программы.

В основной части: ключевые фрагменты

Включайте только значимые блоки кода, которые иллюстрируют конкретную операцию. Например: код для imread, cvtColor, resize. Каждый фрагмент — 5-15 строк с комментариями.

В приложении: полный листинг

Если у вас большой ноутбук с множеством экспериментов, вынесите полный код в приложение в конце отчёта. Или предоставьте ссылку на Google Colab ноутбук.

Форматирование кода

- **Моноширинный шрифт:** Courier New или Consolas для читаемости
- **Отступы:** сохраняйте Python-отступы (4 пробела)
- **Подсветка синтаксиса:** желательна, но не обязательна
- **Комментарии:** на русском языке, поясняющие каждый блок

Нумерация листингов

Если требуется: «Листинг 1 — Загрузка и проверка изображения». Ссылки в тексте: «Как показано в листинге 2...»

Обычно для лабораторных работ достаточно вставлять код как блоки без формальной нумерации, но с пояснительным текстом до и после.

```
# Пример правильно оформленного фрагмента кода в отчёте:

# Загрузка изображения с проверкой успешности
img = cv2.imread(fname, cv2.IMREAD_COLOR)
if img is None:
    raise FileNotFoundError(f"Не удалось прочитать файл: {fname}")

# Вывод параметров изображения
print("Shape:", img.shape)
print("Data type:", img.dtype)
```

Такой фрагмент самодостаточен: есть комментарии, понятно, что делает код, можно воспроизвести. Это оптимальный баланс между полнотой и краткостью для основной части отчёта.

Воспроизводимость: что фиксировать обязательно

Номер варианта

Указать явно в начале отчёта: «Вариант № 12». Привести все формулы параметров и вычисленные значения.

Версия библиотек

Зафиксировать версию OpenCV: например, «cv2.__version__ = 4.7.0». Это важно, так как поведение функций может меняться между версиями.

Источник данных

Указать, откуда взято исходное изображение: загружено (имя файла, размер), или сгенерировано (параметры генерации: размер, фон, содержимое).

Имена выходных файлов

Перечислить все сохранённые файлы-результаты с описанием: out_gray.png — grayscale, out_resize_042.png — уменьшено с fx=0.42 и т.д.

Shape/dtype до и после

Для ключевых операций указать, как изменились shape и dtype. Например: «Исходное: (480, 640, 3) uint8, после grayscale: (480, 640) uint8».

Пример раздела в отчёте

Параметры выполнения:

- Вариант: 12
- OpenCV версия: 4.7.0
- Исходное изображение: загружено из photo.jpg, (800, 1200, 3) uint8
- fx = 0.42, fy = 0.36, beta = 40, rotate = 90° CW
- Canvas: (300, 356, 3)

Список источников: оформление по ГОСТ

Список литературы (источников) — обязательный раздел любого технического отчёта. Даже если вы использовали только конспект лекции и документацию OpenCV, их необходимо корректно оформить.

01

Конспект лекции / материалы курса

Если есть официальный конспект, методические указания или презентации курса — указываем как первый источник. Формат: Автор (преподаватель). Название курса/раздела. Учебное заведение, год.

02

Документация OpenCV

Официальная документация библиотеки. Формат для электронного ресурса: OpenCV documentation [Электронный ресурс]. — Режим доступа: https://docs.opencv.org/, свободный. — Загл. с экрана. — Дата обращения: DD.MM.YYYY.

03

Документация Google Colab

Если использовали специфичные возможности Colab (например, cv2_imshow): Google Colab documentation [Электронный ресурс]. — Режим доступа: https://colab.research.google.com/. — Дата обращения: DD.MM.YYYY.

04

Дополнительные источники

Если использовались учебники, статьи, онлайн-руководства — оформите их по соответствующим правилам ГОСТ (для книг, статей, электронных ресурсов).

Примеры оформления

Книга:Шапиро Л., Стокман Дж. Компьютерное зрение. — М.: Бином, 2013. — 752 с.

Электронный ресурс: OpenCV: Image Processing [Электронный ресурс]. — Режим доступа: https://docs.opencv.org/4.x/d7/da8/tutorial_py_image_processing.html, свободный. — Дата обращения: 15.12.2024.

Требования

- Минимум 2-4 источника для лабораторной работы
- Все источники, на которые есть ссылки в тексте
- Нумерованный список в порядке упоминания или по алфавиту
- Для интернет-источников — обязательна дата обращения

Типичные ошибки и способы их диагностики

Половина проблем при выполнении лабораторной работы связана не с алгоритмами, а с дисциплиной проверки и пониманием особенностей API. Вот наиболее частые ошибки и как их избежать.

img is None после imread

Причина: файл не найден, неправильное имя, неподдерживаемый формат.
Решение: всегда проверяйте `if img is None` сразу после imread. Печатайте имя файла и проверяйте его наличие в рабочей директории.

Перепутаны x и y при индексации

Причина: массив NumPy индексируется как [row, col] = [y, x], а не [x, y].
Решение: запомните: `img[y, x]` или `img[строка, столбец]`. Проверяйте на маленьком примере.

Перепутаны BGR и RGB в объяснениях

Причина: интуитивно ожидается RGB, но OpenCV использует BGR.
Решение: всегда помните и указывайте в отчёте: OpenCV = BGR. При работе с matplotlib (если используете) нужна конвертация.

Переполнение uint8 при изменении яркости

Причина: прямое `img + 40` вызывает overflow на uint8.
Решение: используйте `cv2.convertScaleAbs()` для безопасного изменения яркости и контраста.

Рисунки в отчёте не соответствуют тексту

Причина: вставили не ту картинку, или перепутали порядок.
Решение: сохраняйте результаты с говорящими именами, проверяйте соответствие перед финализацией отчёта.

Отсутствуют подписи рисунков

Причина: забыли добавить или не знали о требовании.
Решение: каждый рисунок должен иметь подпись «Рисунок N — описание» и ссылку в тексте. Проверьте перед сдачей.

Совет по диагностике: Всегда печатайте shape и dtype после каждой ключевой операции во время разработки. Это помогает отловить проблемы на ранней стадии. Например: `print("After cvtColor:", gray.shape, gray.dtype)`

Многие из этих ошибок можно предотвратить простыми проверками и внимательностью к деталям. Оформление отчёта — это тоже часть оценки: рисунки без подписей фактически не засчитываются как результат работы.

Подготовка к защите: контрольные вопросы

Зашита лабораторной работы — это проверка понимания материала. Недостаточно «запустить код» — нужно уметь объяснить каждый шаг словами и ответить на вопросы преподавателя.

1 Что такое пиксель и разрешение изображения?

Пиксель — элементарная точка изображения. Разрешение — размер изображения в пикселях (ширина × высота). Определяет детальность.

2 Что означает атрибут shape для изображения?

Кортеж размерностей массива: (высота, ширина, каналы). Для цветного: (H, W, 3), для grayscale: (H, W).

3 Почему тип данных uint8 и диапазон 0..255?

uint8 — 8-битное беззнаковое целое, 2^8=256 значений: 0..255. Это стандартное представление интенсивности пикселя, достаточное для человеческого восприятия и экономное по памяти.

4 Чем grayscale отличается от цветного по структуре?

Grayscale — двумерный массив (H, W), один канал яркости. Цветное — трёхмерный (H, W, 3), три канала BGR. Grayscale занимает в 3 раза меньше памяти.

5 Что такое BGR и почему это важно?

BGR — порядок цветовых каналов в OpenCV: Blue, Green, Red (индексы 0, 1, 2). Отличается от привычного RGB. Важно при конвертации между библиотеками и интерпретации значений.

6 Как работает resize и что такое интерполяция?

Resize изменяет число пикселей. При увеличении нужно «придумать» новые пиксели — это интерполяция (вычисление промежуточных значений). При уменьшении детали теряются.

7 Зачем проверять результат imread на None?

imread не вызывает исключение при ошибке, а возвращает None. Без проверки все последующие операции упадут с непонятными ошибками. Проверка — признак грамотного кода.

8 Как обеспечена уникальность вашего варианта?

Все параметры (размеры, коэффициенты fx/fy, beta, rotate и т.д.) вычисляются по формулам от номера варианта. Это даёт разные, но детерминированные результаты.

Итоговый чек-лист: шаги после занятия

После занятия вам предстоит самостоятельно воспроизвести весь сценарий в Google Colab и оформить результаты в виде технического отчёта. Следуйте этому чек-листу, чтобы ничего не упустить:

01

Выполнить в Google Colab

Создайте новый ноутбук, последовательно реализуйте все операции: импорты, загрузку/генерацию изображения, все преобразования (grayscale, resize, flip/rotate, примитивы, работу с каналами), расширенные задания.

02

Сохранить все результаты

Используйте cv2.imwrite для сохранения всех ключевых результатов с говоряшими именами файлов. Скачайте их на локальный компьютер для вставки в отчёт.

03

Сформировать структуру отчёта

Создайте документ Word/PDF с разделами: Титульный лист, Цель, Теория, Ход работы, Результаты, Выводы, Список источников, Приложения. Заполните каждый раздел.

04

Оформить рисунки и ссылки

Вставьте все изображения-результаты, пронумеруйте их, добавьте подписи «Рисунок N — описание». Убедитесь, что на каждый рисунок есть ссылка в тексте.

05

Указать вариант и параметры

В начале раздела «Ход работы» явно зафиксируйте номер варианта и все вычисленные параметры. Проверьте корректность формул.

06

Подготовиться к защите

Перечитайте отчёт, убедитесь, что можете объяснить каждую операцию. Повторите ответы на контрольные вопросы. Будьте готовы показать ноутбук и результаты.

Финальная проверка перед сдачей

- ✓ Все обязательные результаты представлены
- ✓ Рисунки пронумерованы и подписаны
- ✓ Ссылки на рисунки в тексте присутствуют
- ✓ Вариант и параметры указаны
- ✓ Список источников оформлен
- ✓ Отчёт оформлен по ГОСТ
- ✓ Ноутбук работает и воспроизводим

Что дальше

На следующих занятиях мы будем расширять инструментарий: фильтрация, выделение границ, морфологические операции, детекция признаков. Поэтому важно, чтобы базовые операции работы с изображениями были доведены до автоматизма.

Успешное выполнение этой работы — фундамент для всего курса машинного зрения. Удачи!