# Лабораторная работа №1

# Установка

Вариант для Windows, версия Python 3.\* (тестировалось на 3.7.3).

Действия в консоли CMD.exe.

Первым делом нам потребуется библиотека Numpy , если она не стоит то ставим:

**pip install numpy**

Далее ставим из репозитория OpenCV, вы так же можете собрать и из исходников, ( ссылка для мазохистов - <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_setup_in_windows/py_setup_in_windows.html> ).

**pip install opencv-python # Только основные методы**

**pip install opencv-contrib-python # Все методы если не ошибаюсь**

# Начала работы с изображениями

## Загрузка изображений

Используйте функцию **cv2.imread()** для чтения изображения. Изображение должно быть в рабочем каталоге или должен быть указан полный путь к изображению.

Второй аргумент - это флаг, который определяет способ чтения изображения.

- cv2.IMREAD\_COLOR : загружает цветное изображение. Любой прозрачностью изображения можно пренебречь. Это флаг по умолчанию.

- cv2.IMREAD\_GRAYSCALE : загружает изображение в режиме оттенков

серого

- cv2.IMREAD\_UNCHANGED : загружает изображение как таковое, включая альфа-канал.

Вместо этих трех флагов вы можете просто передать целые числа **1, 0** или **-1** соответственно.

Отображение изображения

Используйте функцию cv2.imshow() для отображения изображения в окне. Окно автоматически подстраивается под размер изображения.

Первый аргумент-это имя окна, которое является строкой. второй аргумент-это наш имидж. Вы можете создать столько окон, сколько пожелаете, но с разными именами окон.

Импорт и просмотр изображения

**import cv2**

**image = cv2.imread("./путь/к/изображению.расширение")**

**cv2.imshow("Image", image)**

**cv2.waitKey(0)**

**cv2.destroyAllWindows()**

**cv2.waitKey ()** - это функция привязки клавиатуры. Его аргументом является время в миллисекундах. Функция ожидает заданные миллисекунды для любого события клавиатуры. Если вы нажмете любую клавишу за это время, программа продолжится. Если передается 0, то он ожидает бесконечно долго нажатия клавиши. Он также может быть настроен на обнаружение определенных нажатий клавиш, таких как, если нажата клавиша а и т. д., которые мы обсудим ниже.

**cv2.destroyAllWindows()** просто уничтожает все созданные нами окна. Если вы хотите уничтожить какое-либо конкретное окно, Используйте функцию **cv2.destroyWindow()**, где вы передаете точное имя окна в качестве аргумента.

При чтении способом выше изображение находится в цветовом пространстве не RGB (как все привыкли), а BGR. Возможно, вначале это не так важно, но как только вы начнёте работать с цветом — стоит знать об этой особенности.

Есть 2 пути решения:

1) Поменять местами 1-й канал (R — красный) с 3-м каналом (B — синий), и тогда красный цвет будет (0,0,255), а не (255,0,0).

Поменять цветовое пространство на RGB:

2) rgb\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

И тогда в коде работать уже не с image, а с rgb\_image.

(Чтобы закрыть окно, в котором отображается изображение, нажмите любую клавишу. Если использовать кнопку закрытия окна, можно наткнуться на подвисания - варитивно.)

Типичный код для вывода изображений:

**import cv2**

**def viewImage(image, name\_of\_window):**

**cv2.namedWindow(name\_of\_window, cv2.WINDOW\_NORMAL)**

**cv2.imshow(name\_of\_window, image)**

**cv2.waitKey(0)**

**cv2.destroyAllWindows()**

Примечание

Существует особый случай, когда вы уже можете создать окно и загрузить в него изображение позже. В этом случае вы можете указать, можно ли изменить размер окна или нет. Это делается с помощью функции **cv2.namedWindow()**. По умолчанию используется флаг **cv2.WINDOW\_AUTOSIZE**. Но если вы укажете флаг **cv2.WINDOW\_NORMAL**, вы можете изменить размер окна. Это будет полезно, когда изображение слишком велико по размеру и добавляет панель треков в окна.

Смотрите код ниже:

**import cv2**

**image = cv2.imread("dHWei9jpqrM.jpg")**

**cv2.namedWindow("image", cv2.WINDOW\_NORMAL)**

**cv2.imshow("image", image)**

**cv2.waitKey(0)**

**cv2.destroyAllWindows()**

## Сохранение изображения

Используйте функцию **cv2.imwrite()** для сохранения изображения.

Первый аргумент-это имя файла, второй аргумент-изображение, которое вы хотите сохранить.

**cv2.imwrite (“gray.png”,img)**

Это позволит сохранить изображение в формате PNG в рабочем каталоге.

Программа ниже загружает изображение в оттенках серого, отображает его, сохраняет изображение, если вы нажмете " s " и выходите, или просто выходите без сохранения, если вы нажмете клавишу ESC.

**import numpy as np**

**import cv2**

**img = cv2.imread('messi5.jpg',0)**

**cv2.imshow('image',img)**

**k = cv2.waitKey(0)**

**if k == 27: # wait for ESC key to exit**

**cv2.destroyAllWindows()**

**elif k == ord('s'): # wait for 's' key to save and exit**

**cv2.imwrite('messigray.png',img)**

**cv2.destroyAllWindows()**

Если вы используете 64-разрядном компьютере, вы будете иметь, чтобы изменить к = cv2.waitKey(0) следующим образом : K = cv2.waitKey(0) & значение 0xFF.

**Применение библиотеки Matplotlib**

Matplotlib-это библиотека построения графиков для Python, которая предоставляет вам широкий спектр методов построения графиков. Здесь вы узнаете, как отображать изображение с помощью Matplotlib. Вы можете масштабировать изображения, сохранять их и т. д. С помощью Matplotlib.

# Работа с видеопотоком

Для захвата видео необходимо создать объект **VideoCapture**. Его аргументом может быть либо индекс устройства, либо имя видеофайла. Индекс устройства-это просто номер, указывающий, какая камера. Обычно подключается одна камера (как в моем случае). Поэтому я просто передаю 0 (или -1). Вы можете выбрать вторую камеру, пройдя 1 и так далее. После этого вы можете захватывать кадр за кадром. Но в конце не забудьте отпустить захват.

**import numpy as np**

**import cv2**

**cap = cv2.VideoCapture(0)**

**while(True):**

**# Capture frame-by-frame**

**ret, frame = cap.read()**

**# Our operations on the frame come here**

**gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)**

**# Display the resulting frame**

**cv2.imshow('frame',gray)**

**if cv2.waitKey(1) & 0xFF == ord('q'):**

**break**

**# When everything done, release the capture**

**cap.release()**

**cv2.destroyAllWindows()**

## Web camera

С простыми USB-web камерами очень просто, главное найти номер нужной камеры, если их подключено несколько:

**import cv2**

**cap = cv2.VideoCapture(0)**

**while True:**

**ret, img = cap.read()**

**cv2.imshow("camera", img)**

**if cv2.waitKey(10) == 27: # Клавиша Esc**

**break**

**cap.release()**

**cv2.destroyAllWindows()**

Подключаемся ( захватываем) нашу веб камеру. 0 — это индекс камеры, если их несколько то будет 0 или 1 и т.д.

**ret, img = cap.read()**

Читаем с устройства кадр(картинку) , метод возвращает флаг **ret (True , False)** и **img** — саму картинку ( массив numpy) .

**cv2.imshow(«camera», img)**

Функция **imshow** отображает изображение в указанном окне. Если окно не было создано, то создается новое. **«camera»** — имя окна , **img** — массив картинки.

По сути мы получаем картинку(кадр) с камеры и показываем его.

**cap.read()** возвращает значение **bool (True/False)**. Если кадр прочитан правильно, он будет истинным. Таким образом, вы можете проверить конец видео, проверив это возвращаемое значение.

Иногда **cap** может не инициализировать захват. В этом случае этот код показывает ошибку. Вы можете проверить, инициализирован ли он или нет методом **cap.isOpened().** Если это правда, то хорошо. В противном случае откройте его с помощью **cap.open()**.

Вы также можете получить доступ к некоторым функциям этого видео с помощью метода **cap.get(propId)**, где **propId**-это число от 0 до 18. Каждое число обозначает свойство видео (если оно применимо к этому видео). Некоторые из этих значений можно изменить с помощью **cap.set(propId, value)**. Значение-это новое значение, которое вы хотите.

(

1. CV\_CAP\_PROP\_POS\_MSEC текущая позиция видеофайла в миллисекундах или временная метка захвата видео.
2. CV\_CAP\_PROP\_POS\_FRAMES 0-индекс кадра, который будет декодирован/захвачен следующим.
3. CV\_CAP\_PROP\_POS\_AVI\_RATIO относительное положение видеофайла: 0 - начало фильма, 1 - Конец фильма.
4. CV\_CAP\_PROP\_FRAME\_WIDTH ширина кадров в видеопотоке.
5. CV\_CAP\_PROP\_FRAME\_HEIGHT Высота кадров в видеопотоке.
6. CV\_CAP\_PROP\_FPS Частота кадров.
7. CV\_CAP\_PROP\_FOURCC 4-символьный код кодека.
8. CV\_CAP\_PROP\_FRAME\_COUNT количество кадров в видеофайле.
9. CV\_CAP\_PROP\_FORMAT формат объектов Mat, возвращаемых функцией retrieve() .
10. CV\_CAP\_PROP\_MODE Backend-специфичное значение, указывающее текущий режим захвата.
11. CV\_CAP\_PROP\_BRIGHTNESS яркость изображения (только для камер).
12. CV\_CAP\_PROP\_CONTRAST контрастность изображения (только для камер).
13. CV\_CAP\_PROP\_SATURATION насыщенность изображения (только для камер).
14. CV\_CAP\_PROP\_HUE оттенок изображения (только для камер).
15. CV\_CAP\_PROP\_GAIN усиление изображения (только для камер).
16. CV\_CAP\_PROP\_EXPOSURE Экспозиция (только для камер).
17. CV\_CAP\_PROP\_CONVERT\_RGB булевы флаги, указывающие, является ли изображения должны быть преобразованы в RGB.
18. CV\_CAP\_PROP\_WHITE\_BALANCE\_U значение U настройки баланса белого (Примечание: В настоящее время поддерживается только бэкендом DC1394 v 2.x)
19. CV\_CAP\_PROP\_WHITE\_BALANCE\_V значение V настройки баланса белого (Примечание: В настоящее время поддерживается только бэкендом DC1394 v 2.x)
20. CV\_CAP\_PROP\_RECTIFICATION Флаг выпрямления для стереокамер (Примечание: В настоящее время поддерживается только бэкендом DC1394 v 2.x)
21. CV\_CAP\_PROP\_ISO\_SPEED скорость ISO камеры (Примечание: В настоящее время поддерживается только бэкендом DC1394 v 2.x)
22. Размер CV\_CAP\_PROP\_BUFFER количество кадров, хранящихся во внутренней буферной памяти (Примечание: В настоящее время поддерживается только бэкендом DC1394 v 2.x)

)

Например, я могу проверить ширину и высоту кадра с помощью **cap.get(3)** и **cap.get(4)**. По умолчанию он дает мне 640x480. Но я хочу изменить его на 320x240. Просто используйте **ret = cap.set(3,320)** и **ret = cap.set(4,240)**. (для лучшей читабельности кода лучше пользоваться именами параметров).

**cv2.waitKey(10)**

Как понятно из названия, ожидает нажатия клавиши в мсек. Возвращает код клавиши или -1 если ничего не было нажато.

## Настройки камеры

Все отлично, но хотелось бы немного более тонко настроить вебку :

**cap = cv2.VideoCapture(0)**

**cap.set(cv2.CAP\_PROP\_FPS, 24) # Частота кадров**

**cap.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 600) # Ширина кадров в видеопотоке.**

**cap.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 480) # Высота кадров в видеопотоке.**

**# Все параметры.**

Можем даже немного поиграть с цветами :

**gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)**

**cv2.imshow("camera", gray) # Будет показывать в оттенках серого.**

**cv2.cvtColor()** — метод преобразует изображение из одного цветового пространства в другое. **cv2.COLOR\_BGR2GRAY** — преобразование между RGB / BGR и оттенками серого. Но это все уже работа с изображением.

## Запись видео с вебки

С записью видео тоже ничего сложного:

**import cv2**

**cap = cv2.VideoCapture(0)**

**codec = cv2.VideoWriter\_fourcc(\*'XVID')**

**out = cv2.VideoWriter('captured.avi',codec, 25.0, (640,480))**

**while(cap.isOpened()):**

**ret, frame = cap.read()**

**if cv2.waitKey(1) & 0xFF == or ('q') or ret == False:**

**break**

**cv2.imshow('frame', frame)**

**out.write(frame)**

**out.release()**

**cap.release()**

**cv2.destroyAllWindows()**

**codec = cv2.VideoWriter\_fourcc(\*’XVID’)**

Данный метод просто передает индификатор кодека, которым будем кодировать видео. Вы можете использовать и другие: MJPG , X264 .

Тут мы создаем объект в который, по сути, будет записываться видео кадр за кадром:

**out = cv2.VideoWriter(‘captured.avi’,codec, 25.0, (640,480))**

**cv2.VideoWriter(filename, fourcc, fps, frameSize)**

Записываем очередной кадр..

**out.write(frame)**

Все кадры хранятся в памяти , теперь нам надо закрыть запить и сохранить все в файл :

**out.release()**

Ниже код захвата с камеры, переворачивающий каждый кадр в вертикальном направлении и сохраняющий его.

**import numpy as np**

**import cv2**

**cap = cv2.VideoCapture(0)**

**# Define the codec and create VideoWriter object**

**fourcc = cv2.VideoWriter\_fourcc(\*'XVID')**

**out = cv2.VideoWriter('output.avi',fourcc, 20.0, (640,480))**

**while(cap.isOpened()):**

**ret, frame = cap.read()**

**if ret==True:**

**frame = cv2.flip(frame,0)**

**# write the flipped frame**

**out.write(frame)**

**cv2.imshow('frame',frame)**

**if cv2.waitKey(1) & 0xFF == ord('q'):**

**break**

**else:**

**break**

**# Release everything if job is finished**

**cap.release()**

**out.release()**

**cv2.destroyAllWindows()**

## Воспроизведение видео из файла

Это то же самое, что захват с камеры, просто измените индекс камеры с именем видеофайла. Кроме того, при отображении кадра используйте соответствующее время для **cv2.waitKey()**. Если он слишком мал, видео будет очень быстрым, а если он слишком высок, видео будет медленным (Ну, именно так вы можете отображать видео в замедленном режиме). 25 миллисекунд будет нормально в нормальных случаях.

**import numpy as np**

**import cv2**

**cap = cv2.VideoCapture('output.avi')**

**while(cap.isOpened()):**

**ret, frame = cap.read()**

**gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)**

**cv2.imshow('frame',gray)**

**if cv2.waitKey(1) & 0xFF == ord('q'):**

**break**

**cap.release()**

**cv2.destroyAllWindows()**

# Функции рисования в OpenCV

*Цель*

*Научитесь рисовать различные геометрические фигуры с помощью OpenCV*

*Вы узнаете эти функции : cv2.line(), cv2.circle() , cv2.rectangle(), cv2.ellipse(), cv2.putText() и т. д.*

Во всех вышеперечисленных функциях вы увидите некоторые общие аргументы, как показано ниже:

* **img** : изображение, на котором вы хотите нарисовать фигуры
* **color** : цвет фигуры. для BGR передайте его как кортеж, например: (255,0,0) для синего цвета. Для оттенков серого просто передайте скалярное значение.
* **thickness** -толщина : Толщина линии или круга и т. д. Если для замкнутых фигур, таких как круги, передается значение -1, то оно заполняет фигуру. толщина по умолчанию = 1
* **lineType** : тип линии, будь то 8-соединенная, сглаженная линия и т.д. По умолчанию это 8-connected. cv2.LINE\_AA дает сглаженную линию, которая отлично смотрится для кривых.

### Отрисовка линий

Чтобы нарисовать линию, вам нужно передать начальные и конечные координаты линии. Мы создадим черное изображение и проведем на нем синюю линию от левого верхнего угла до правого нижнего.

**import numpy as np**

**import cv2**

**# Create a black image**

**img = np.zeros((512,512,3), np.uint8)**

**# Draw a diagonal blue line with thickness of 5 px**

**img = cv2.line(img,(0,0),(511,511),(255,0,0),1)**

**cv2.imshow('lime',img)**

**# отрисовываем окошко**

**if cv2.waitKey(10000) & 0xFF == ord('q'):**

**cv2.destroyAllWindows()**

Чтобы нарисовать прямоугольник, вам нужен верхний левый угол и нижний правый угол прямоугольника. На этот раз мы нарисуем зеленый прямоугольник в правом верхнем углу изображения.

**img = cv2.rectangle(img,(384,0),(510,128),(0,255,0),3)**

### Отрисовка окружности

Чтобы нарисовать круг, вам нужны координаты его центра и радиус. Мы нарисуем круг внутри прямоугольника, нарисованного выше.

**img = cv2.circle(img,(447,63), 63, (0,0,255), -1)**

### Эллипс

Чтобы нарисовать эллипс, нам нужно передать несколько аргументов. Одним из аргументов является расположение центра (x,y). Следующий аргумент-длина осей (длина большой оси, длина малой оси). угол-это угол поворота эллипса в направлении против часовой стрелки. startAngle и endAngle обозначают начало и конец дуги эллипса, измеренной по часовой стрелке от главной оси, т. е. давая значения 0 и 360, мы получаем полный эллипс. Для получения более подробной информации обратитесь к документации cv2.ellipse(). Ниже пример рисует полуэллипс в центре изображения.

**img = cv2.ellipse(img,(256,256),(100,50),0,0,180,255,-1)**

### Рисование Многоугольника

Чтобы нарисовать многоугольник, сначала вам нужны координаты вершин. Сделайте эти точки массивом формы ROWSx1x2, где строки-это количество вершин, и он должен быть типа int32. Здесь мы рисуем небольшой многоугольник с четырьмя вершинами желтого цвета.

**pts = np.array([[10,5],[20,30],[70,20],[50,10]], np.int32)**

**pts = pts.reshape((-1,1,2))**

**img = cv2.polylines(img,[pts],True,(0,255,255))**

Если третий аргумент cv2.polylines() - False, вы получите полилинию, соединяющую все точки, а не замкнутую фигуру.

**cv2.polylines()** можно использовать для рисования нескольких линий. Просто создайте список всех линий, которые вы хотите нарисовать, и передайте его в функцию. Все линии будут нарисованы индивидуально. Это более лучший и быстрый способ нарисовать группу линий, чем вызов cv2.line() для каждой линии.

**Добавление текста к изображениям**

Чтобы поместить тексты в изображения, вам нужно указать следующие вещи.

- Текстовые данные, которые вы хотите записать

- Координаты положения того места, где вы хотите его поместить (т. е. нижний левый угол, где начинаются данные).

- Тип шрифта (проверьте документы cv2.putText() для поддерживаемых шрифтов)

- Масштаб шрифта (определяет размер шрифта)

- обычные вещи, такие как цвет, толщина, тип линии и т. д. Для лучшего внешнего вида рекомендуется использовать lineType = cv2.LINE\_AA.

Мы напишем OpenCV на нашем изображении белым цветом.

**font = cv2.FONT\_HERSHEY\_SIMPLEX**

**cv2.putText(img,'OpenCV',(10,500),font,4,(255,255,255),2,cv2.LINE\_AA)**

И суммарно имеем примерно такой скрипт для отрисовки вышеупомянутых примеров:

**import numpy as np**

**import cv2**

**# Create a black image**

**img = np.zeros((512,512,3), np.uint8)**

**# Draw a diagonal blue line with thickness of 5 px**

**img = cv2.line(img,(0,0),(511,511),(255,0,0),1)**

**img = cv2.rectangle(img,(384,0),(510,128),(0,255,0),3)**

**img = cv2.circle(img,(447,63), 63, (0,0,255), -1)**

**img = cv2.ellipse(img,(256,256),(100,50),0,0,210,255,1)**

**pts = np.array([[10,5],[20,30],[70,20],[50,10]], np.int32)**

**pts = pts.reshape((-1,1,2))**

**img = cv2.polylines(img,[pts],True,(0,255,255))**

**font = cv2.FONT\_HERSHEY\_SIMPLEX**

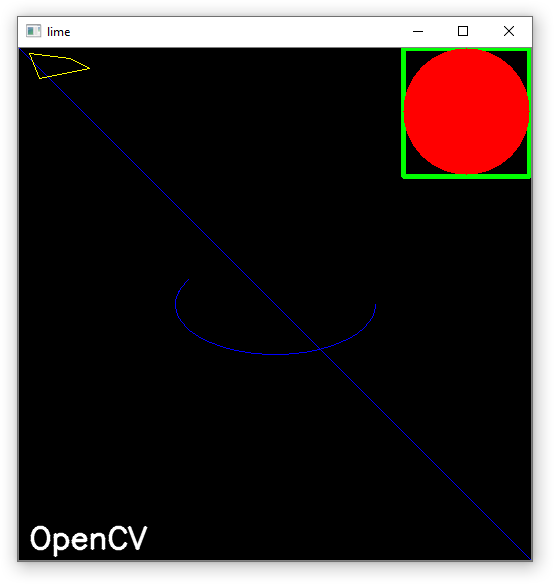
**cv2.putText(img,'OpenCV',(10,500), font,1,(255,255,255),2,cv2.LINE\_AA)**

**cv2.imshow('lime',img)**

**# show image**

**if cv2.waitKey(10000) & 0xFF == ord('q'):**

**cv2.destroyAllWindows()**

****

Вопросы к лабораторной работе

1. привести подробное описание основных функций, рассмотренных в работе - функция, что делает, аргументы, параметры, возвращаемые результаты, типы данных и тп.
2. выполнить примеры, описанные в лабораторной - предоставить комментированный код, поэтапные результаты выполнения действий (исходные фото-, видео- данные для примеров берутся собственные, или из открытых источников или из базового набора библиотеки OpenCV)