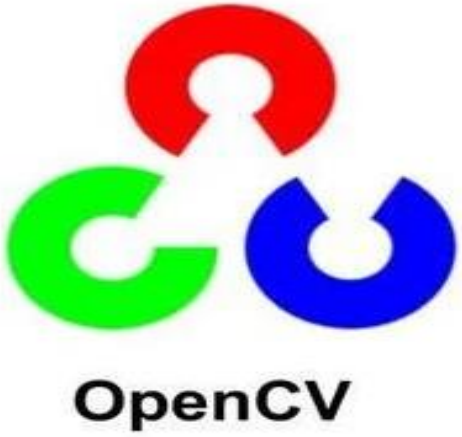
OpenCV

Семинар 6

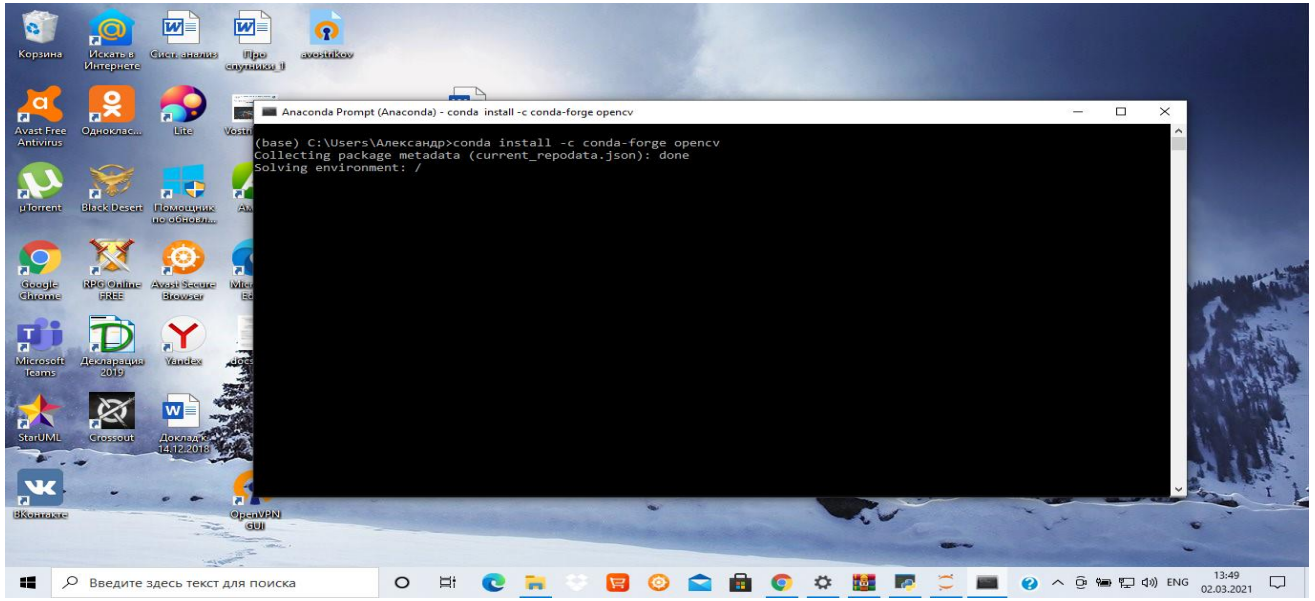
Введение

- OpenCV — это open source (с открытым кодом) библиотека компьютерного зрения, которая предназначена для анализа, классификации и обработки изображений. Широко используется в таких языках как C, C++, Python и Java.



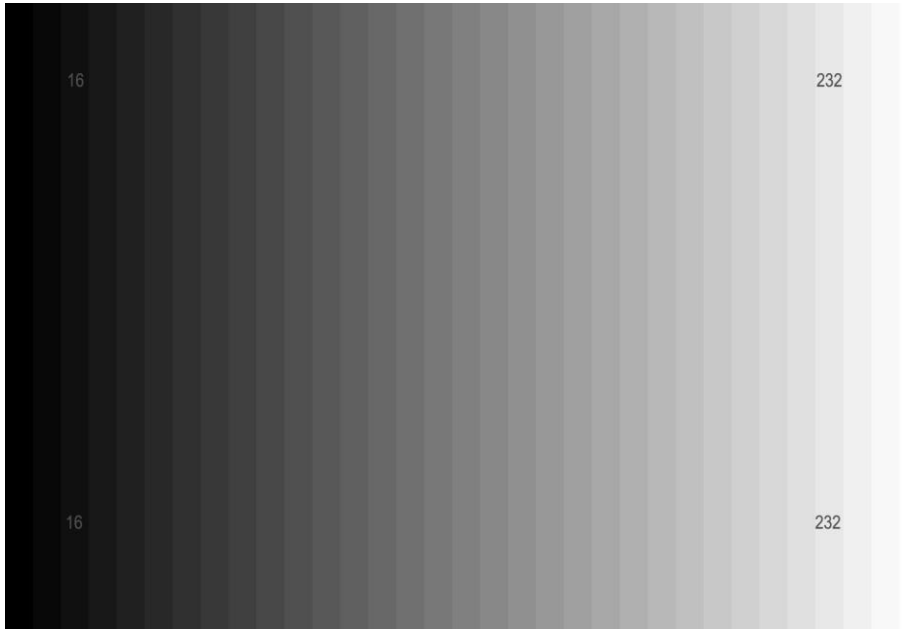
Установка

- Инструкция для установки python на windows и на ubuntu, установка OpenCV на windows и на ubuntu.
- Для проверки результата установки просто введите в терминале Python следующую команду:
  import cv2
- Если не появилось сообщение об ошибке, значит библиотека была установлена успешно.



Все дело в пикселях

- В большинстве изображений пиксели представлены двумя способами: в оттенках серого и в цветовом пространстве RGB. В изображениях в оттенках серого каждый пиксель имеет значение между 0 и 255, где 0 соответствует чёрному, а 255 соответствует белому. А значения между 0 и 255 принимают различные оттенки серого, где значения ближе к 0 более тёмные, а значения ближе к 255 более светлые:



Все дело в пикселях

- Цветные пиксели обычно представлены в цветовом пространстве RGB(red, green, blue — красный, зелёный, синий), где одно значение для красной компоненты, одно для зелёной и одно для синей. Каждая из трёх компонент представлена целым числом в диапазоне от 0 до 255 включительно, которое указывает как «много» цвета содержится. Исходя из того, что каждая компонента представлена в диапазоне [0,255], то для того, чтобы представить насыщенность каждого цвета, нам будет достаточно 8-битного целого беззнакового числа. Затем мы объединяем значения всех трёх компонент в кортеж вида (красный, зеленый, синий). К примеру, чтобы получить белый цвет, каждая из компонент должна равняться 255: (255, 255, 255). Тогда, чтобы получить чёрный цвет, каждая из компонент должна быть равной 0: (0, 0, 0). Ниже приведены распространённые цвета, представленные в виде RGB кортежей:

Black	rgb(0, 0, 0)
White	rgb(255, 255, 255)
Red	rgb(255, 0, 0)
Blue	rgb(0, 0, 255)
Green	rgb(0, 255, 0)
Yellow	rgb(255, 255, 0)
Magenta	rgb(255, 0, 255)
Cyan	rgb(0, 255, 255)
Violet	rgb(136, 0, 255)
Orange	rgb(255, 136, 0)

Основные операции с изображениями

- Вывод изображения на экран
- Сохранение изображений
- Арифметика изображений
- Сглаживание изображений
- Преобразование изображений

Загрузка, отображение и сохранение изображения

- def loading_displaying_saving():
- img = cv2.imread('girl.jpg', cv2.IMREAD_GRAYSCALE)
- cv2.imshow('girl', img)
- cv2.waitKey(0)
- cv2.imwrite('graygirl.jpg', img)
- Для загрузки изображения мы используем функцию cv2.imread(), где первым аргументом указывается путь к изображению, а вторым аргументом, который является необязательным, мы указываем, в каком цветовом пространстве мы хотим считать наше изображение. Чтобы считать изображение в RGB — cv2.IMREAD_COLOR, в оттенках серого — cv2.IMREAD_GRAYSCALE. По умолчанию данный аргумент принимает значение cv2.IMREAD_COLOR. Данная функция возвращает 2D (для изображения в оттенках серого) либо 3D (для цветного изображения) массив NumPy. Форма массива для цветного изображения: высота x ширина x 3, где 3 — это байты, по одному байту на каждую из компонент. В изображениях в оттенках серого всё немного проще: высота x ширина.
- С помощью функции cv2.imshow() мы отображаем изображение на нашем экране. В качестве первого аргумента мы передаём функции название нашего окна, а вторым аргументом изображение, которое мы загрузили с диска, однако, если мы далее не укажем функцию cv2.waitKey(), то изображение моментально закроется. Данная функция останавливает выполнение программы до нажатия клавиши, которую нужно передать первым аргументом. Для того, чтобы любая клавиша была засчитана передаётся 0.

Арифметика изображений

- Сложение изображений

```
import cv2

# Считываем два изображения
image_1 = cv2.imread('bike.jpg')
image_2 = cv2.imread('car.jpg')

# Суммируем массивы двух изображений по всем каналам
result = cv2.add(image_1, image_2)

cv2.imshow('result', result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

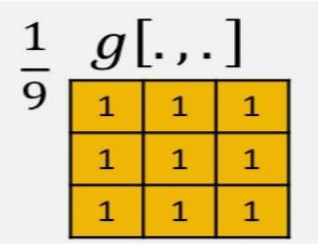
Арифметика изображений

- Смешение изображений весьма похоже на их сложение, за исключением того, что теперь мы можем контролировать вклад каждого из входящих изображений в результирующее. В общем случае, если мы хотим, чтобы одно из входящих изображений было более контрастным, а другое более размытым при их слиянии, мы должны вместо сложения изображений использовать их смешение.

```
import cv2

# Считываем два изображения
image_1 = cv2.imread('bike.jpg')
image_2 = cv2.imread('car.jpg')

result = cv2.addWeighted(image_1, 0.9, image_2, 0.1)

cv2.imshow('result', result)
cv2.waitKey(0) # Программа останавливается до нажатия любой клавиши
cv2.destroyAllWindows()
```

Сглаживание изображений

- Сглаживание изображений является крайне полезной операцией и очень часто используется перед тем как передать картинку для обработки в модель машинного обучения. В основном это нужно делать для фильтрации высокочастотных шумов, применяя для этого низкочастотный фильтр. Существует множество различных фильтров, например усредняющий фильтр (box filter), медианный фильтр (median filter), фильтр типов волн (модовый фильтр, mode filter), фильтр Гаусса (Gaussian filter) и многие другие.
- Фильтр или маска



Сглаживание изображений

```
# Загрузка первоначального изображения
original_image = cv2.imread('my_bike.png')

# Фильтрация изображения усредняющим фильтром 3X3
average_image = cv2.blur(original_image,(3,3))

# Применение фильтра Гаусса к первоначальному изображению
gaussian_image = cv2.GaussianBlur((original_image,(3,3),0))

# Применение медианного фильтра к первоначальному изображению
median_image = cv2.medianBlur(original_image,3)
```

Преобразование изображений

- Масштабирование

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread('my_bike.jpg')

# Увеличиваем масштаб/расширяем в 2 раза по ширине и высоте
result_1 = cv2.resize(image, None, fx=2, fy=2, interpolation=cv2.INTER_CUBIC)

# Уменьшаем масштаб/сжимаем в 2 раза по ширине и высоте
result_2 = cv2.resize(image, None, fx=2, fy=2, interpolation=cv2.INTER_AREA)

# Выводим на экран получившиеся изображения
plt.imshow(result_1)
plt.imshow(result_2)
plt.show()
```

Преобразование изображений

- Вращение позволяет нам перемещать изображение вокруг определенной оси на заданный угол.
- Перед тем как мы научимся вращать наши изображения при помощи библиотеки OpenCV, давайте вспомним, что существует линейный оператор под названием матрица поворота, который как раз и осуществляет преобразования такого рода.

Преобразование изображений. Вращение

```
import cv2
import matplotlib.pyplot as plt

# Загружаем изображение велосипеда
image = cv2.imread('my_bike.jpg',0)

# ряды и колонки
r, c = image.shape

matrix = cv2.getRotationMatrix2D((cols/2,rows/2), 180, 1)
result = cv2.warpAffine(image,matrix,(c,r))

# Выводим на экран повернутое изображение
plt.imshow(result)
plt.show()
```

Пример работы с библиотекой

- colab.research.google.com/drive/1lFUcbNwdQMYfa05PMlrb8c33jmmR6R9U#scrollTo=iQCW9fnUFt85

Пример работы с библиотекой

- https://colab.research.google.com/drive/17zN9Bm_iig-ZZz9tb8KCRcnBLtJS9YrN?usp=sharing

Задание 1

- Средствами библиотеки OpenCV необходимо:

1) Сложить с разными коэффициентами 2 изображения. Вывод результата на экран
2) Результат сложения изображений повернуть на 30 градусов. Вывод результата на экран

Задание 2

- Наложить маску на лицо при использовании веб-камеры

Задание 3

- Распознать свое лицо на видео (из предыдущего задания)
- Сделать и сохранить стоп-кадр с помощью веб-камеры
- Обработать стоп-кадр. Результатом обработки должен стать контур Вашего лица: а) с наименьшим количеством деталей; б) с наибольшим количеством деталей (прорисован каждый волос)