OpenCV

Что такое OpenCV?

Библиотека компьютерного зрения и машинного обучения с открытым исходным кодом. В неё входят более 2500 алгоритмов, в которых есть как классические, так и современные алгоритмы для компьютерного зрения и машинного обучения. Эта библиотека имеет интерфейсы на различных языках, среди которых есть Python, Java, C++ и Matlab.

Импорт и просмотр изображения

```
import cv2
image = cv2.imread("./путь/к/изображению.расширение")
cv2.imshow("Image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

При чтении способом выше изображение находится в цветовом пространстве не RGB (как все привыкли), а BGR. Возможно, в начале это не так важно, но как только вы начнёте работать с цветом — стоит знать об этой особенности. Есть 2 пути решения:

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:

```
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.

На протяжении работы для вывода изображений будет использоваться следующий код:

```
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()
```

Кадрирование

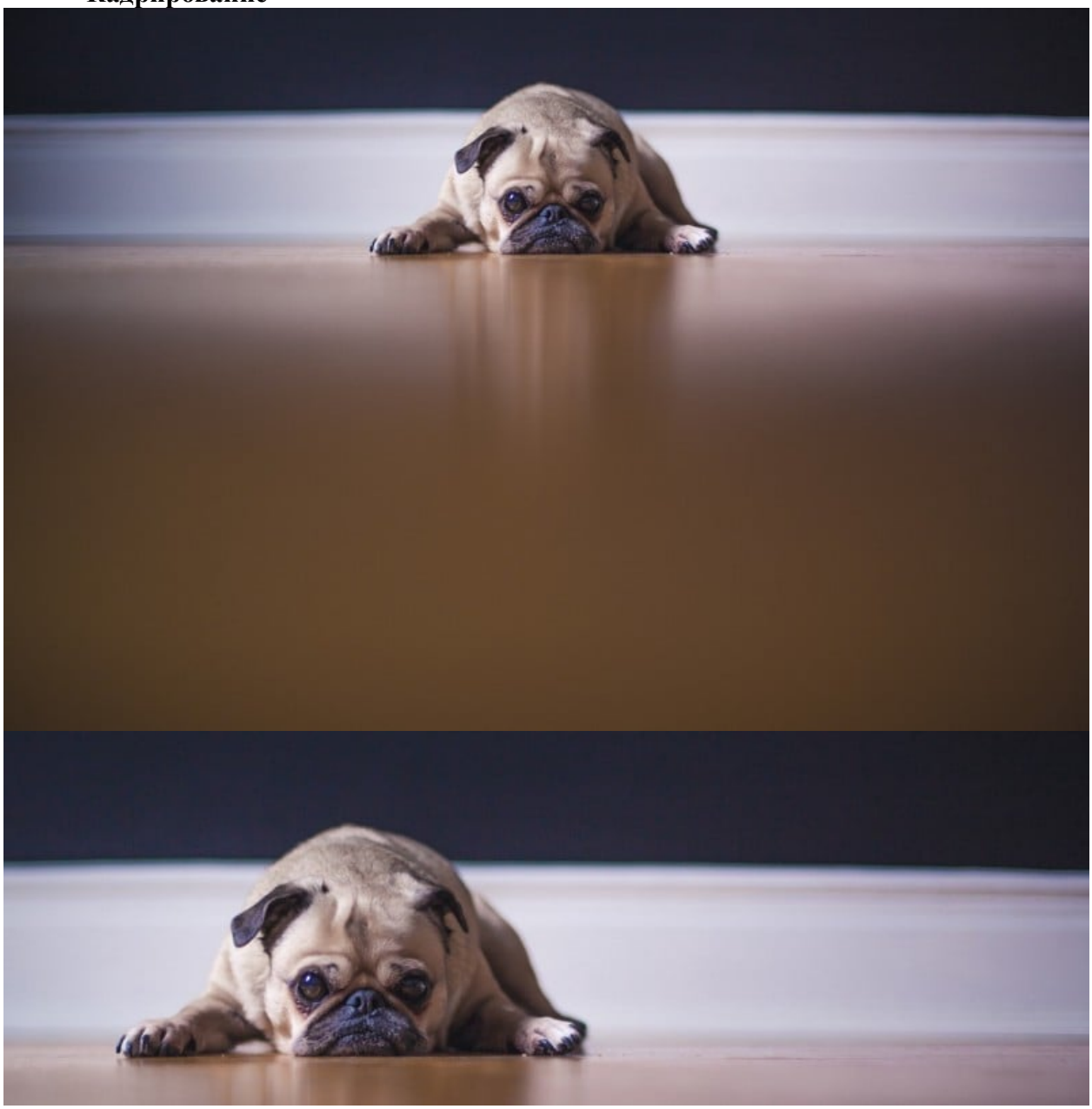

Рисунок 1. Пёсик после кадрирования

```
import cv2
cropped = image[10:500, 500:2000]
viewImage(cropped, "Пёсик после кадрирования")
Где image[10:500, 500:2000] — это image[y:y + высота, x:x + ширина].
```

Изменение размера

```
import cv2
scale_percent = 20 # Процент от изначального размера
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
viewImage(resized, "После изменения размера на 20 %")
```

Эта функция учитывает соотношение сторон оригинального изображения.

Поворот

```
import cv2
(h, w, d) = image.shape
center = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(center, 180, 1.0)
rotated = cv2.warpAffine(image, M, (w, h))
viewImage(rotated, "Пёсик после поворота на 180 градусов")
```

`image.shape` возвращает высоту, ширину и каналы. `M` — матрица поворота — поворачивает изображение на 180 градусов вокруг центра. `-ve` — это угол поворота изображения по часовой стрелке, а `+ve`, соответственно, против часовой.

```
import cv2
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
ret, threshold_image = cv2.threshold(im, 127, 255, 0)
viewImage(gray_image, "Пёсик в градациях серого")
viewImage(threshold_image, "Чёрно-белый пёсик")
```

`gray_image` — это одноканальная версия изображения.

Функция `threshold` возвращает изображение, в котором все пиксели, которые темнее (меньше) 127 заменены на 0, а все, которые ярче (больше) 127, — на 255.

Для ясности другой пример:

```
ret, threshold = cv2.threshold(im, 150, 200, 10)
```

Здесь всё, что темнее, чем 150, заменяется на 10, а всё, что ярче, — на 200.

Размытие/сглаживание

```
import cv2
blurred = cv2.GaussianBlur(image, (51, 51), 0)
viewImage(blurred, "Размытый пёсик")
```

Функция `GaussianBlur` (размытие по Гауссу) принимает 3 параметра:

1 Исходное изображение.

2 Кортеж из 2 положительных нечётных чисел. Чем больше числа, тем больше сила сглаживания.

3 `sigmaX` и `sigmaY`. Если эти параметры оставить равными 0, то их значение будет рассчитано автоматически.

Рисование прямоугольников

```
import cv2
output = image.copy()
cv2.rectangle(output, (2600, 800), (4100, 2400), (0, 255, 255), 10)
viewImage(output, "Обводим прямоугольником лицо пёсика")
```

Эта функция принимает 5 параметров:

1 Само изображение.

2 Координата верхнего левого угла `(x1, y1)`.

3 Координата нижнего правого угла `(x2, y2)`.

4 Цвет прямоугольника (GBR/RGB в зависимости от выбранной цветовой модели).

5 Толщина линии прямоугольника.

Рисование линий

```
import cv2
output = image.copy()
cv2.line(output, (60, 20), (400, 200), (0, 0, 255), 5)
viewImage(output, "2 пёсика, разделённые линией")
```

Функция `line` принимает 5 параметров:

1 Само изображение, на котором рисуется линия.

2 Координата первой точки `(x1, y1)`.

3 Координата второй точки `(x2, y2)`.

4 Цвет линии (GBR/RGB в зависимости от выбранной цветовой модели).
5 Толщина линии.

Текст на изображении

```
import cv2
output = image.copy()
cv2.putText(output, "We <3 Dogs", (1500, 3600),cv2.FONT_HERSHEY_SIMPLEX, 15,
(30, 105, 210), 40)
viewImage(output, "Изображение с текстом")
```

Функция `putText` принимает 7 параметров:

1 Непосредственно изображение.
2 Текст для изображения.
3 Координата нижнего левого угла начала текста `(x, y)`.
4 Используемый шрифт.
5 Размер шрифта.
6 Цвет текста (GBR/RGB в зависимости от выбранной цветовой модели).
7 Толщина линий букв.

Распознавание лиц

```
import cv2
image_path = "./путь/к/фото.расширение"
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
image = cv2.imread(image_path)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(
    gray,
    scaleFactor= 1.1,
    minNeighbors= 5,
    minSize=(10, 10)
)
faces_detected = "Лиц обнаружено: " + format(len(faces))
print(faces_detected)
# Рисуем квадраты вокруг лиц
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (255, 255, 0), 2)
viewImage(image,faces_detected)
```

`detectMultiScale` — общая функция для распознавания как лиц, так и объектов. Чтобы функция искала именно лица, мы передаём ей соответствующий каскад.

Функция `detectMultiScale` принимает 4 параметра:

1 Обрабатываемое изображение в градации серого.

2 Параметр `scaleFactor`. Некоторые лица могут быть больше других, поскольку находятся ближе, чем остальные. Этот параметр компенсирует перспективу.

3 Алгоритм распознавания использует скользящее окно во время распознавания объектов. Параметр `minNeighbors` определяет количество объектов вокруг лица. То есть чем больше значение этого параметра, тем больше аналогичных объектов необходимо алгоритму, чтобы он определил текущий объект, как лицо. Слишком маленькое значение увеличит количество ложных срабатываний, а слишком большое сделает алгоритм более требовательным.

4 `minSize` — непосредственно размер этих областей.

Contours — распознавание объектов

Распознавание объектов производится с помощью цветовой сегментации изображения. Для этого есть две функции: `cv2.findContours` и `cv2.drawContours`.

В этой статье детально описано обнаружение объектов с помощью цветовой сегментации. Всё, что вам нужно для неё, находится там.

Сохранение изображения

```
import cv2
image = cv2.imread("./импорт/путь.расширение")
cv2.imwrite("./экспорт/путь.расширение", image)
```

Заключение

OpenCV — отличная библиотека с лёгкими алгоритмами, которые могут использоваться в 3D-рендере, продвинутом редактировании изображений и видео, отслеживании и идентификации объектов и людей на видео, поиске идентичных изображений из набора и для много-много чего ещё.

Эта библиотека очень важна для тех, кто разрабатывает проекты, связанные с машинным обучением в области изображений.

Задание

1 Загрузить изображение, где более 2 людей
2 Изменить размер изображения
3 Нарисовать фигуру на изображении. Четные – квадрат, нечетные – круг
4 В фигуру вставить текст с вашей ФИО
5 С помощью OpenCV определить количество лиц на изображении