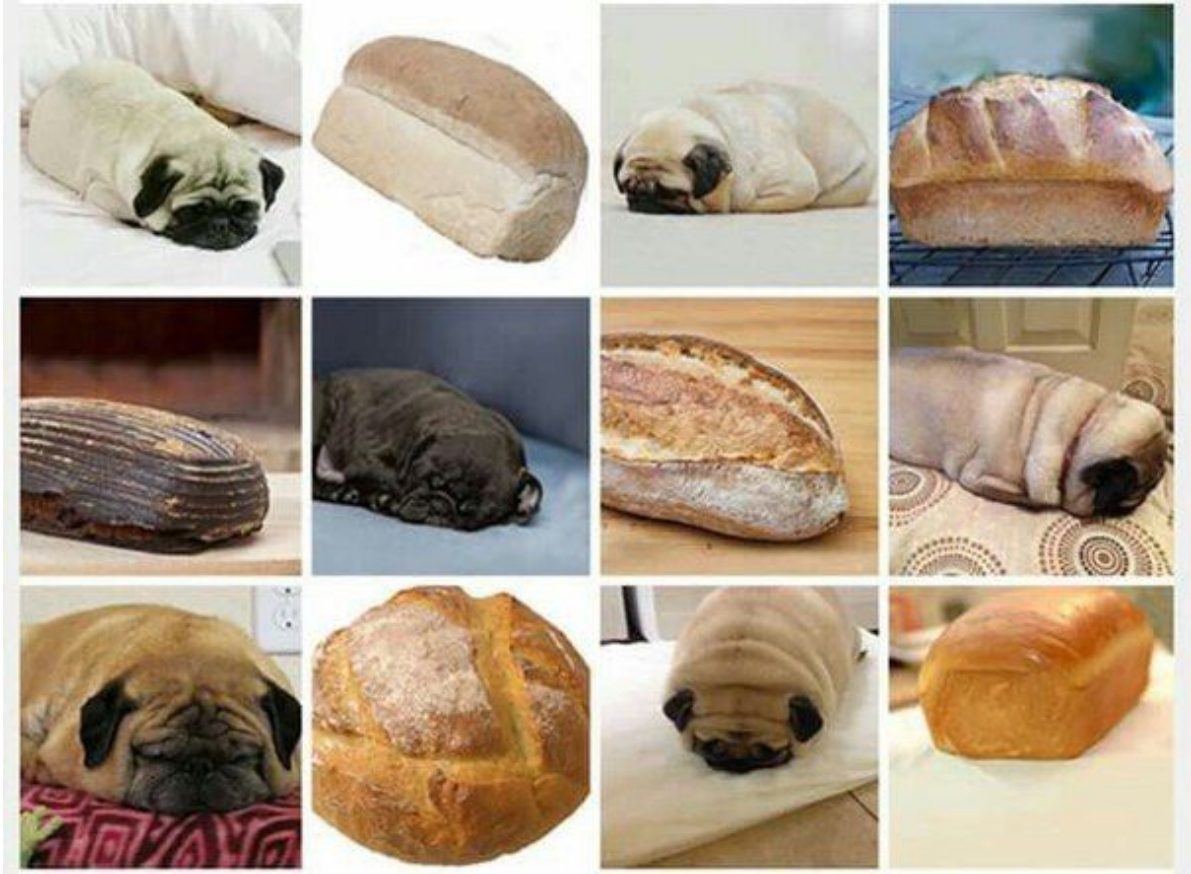
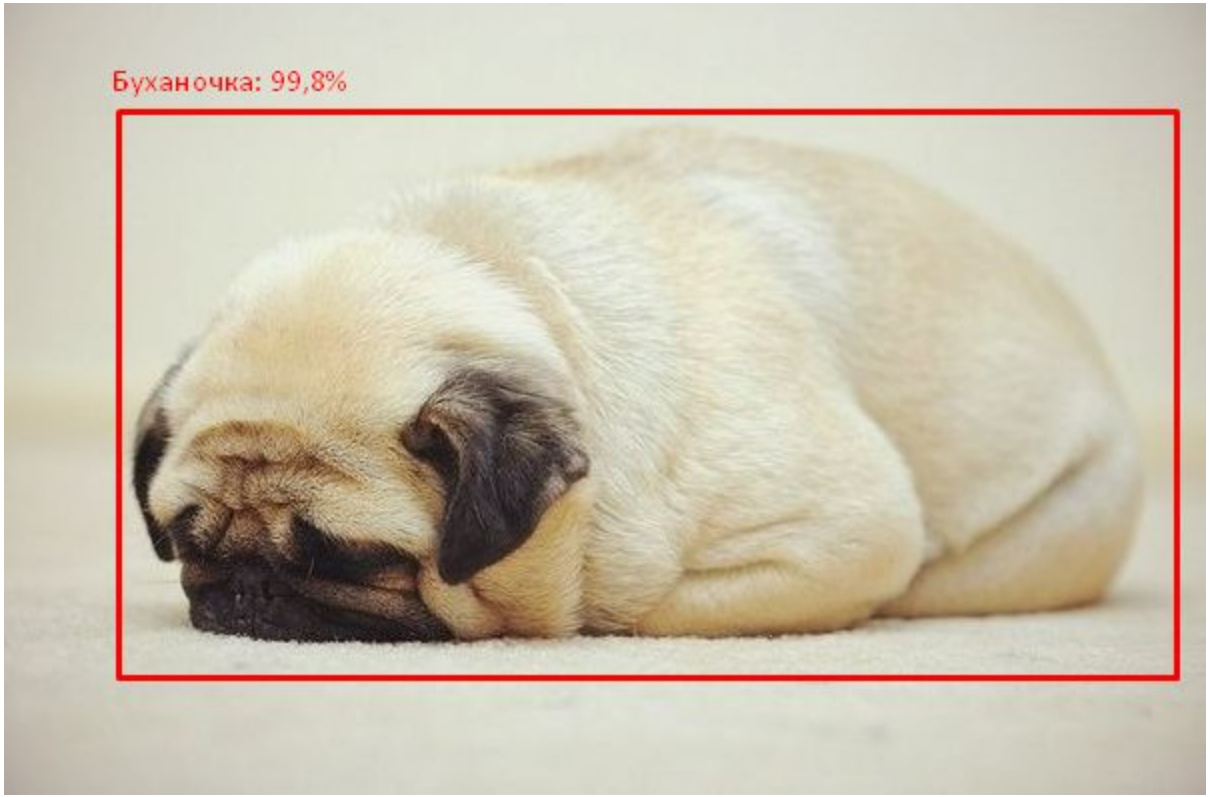
OpenCV

Основные понятия и определения

Pug or Loaf?




Буханочка: 99,8%

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.

Пиксели

Пиксель — это строительный блок изображения. Если представить изображение в виде сетки, то каждый квадрат в сетке содержит один пиксель, где точке с координатой (0, 0) соответствует верхний левый угол изображения. К примеру, представим, что у нас есть изображение с разрешением 400x300 пикселей. Это означает, что наша сетка состоит из 400 строк и 300 столбцов. В совокупности в нашем изображении есть 400*300 = 120000 пикселей.

Если рассматриваем стандартное изображение (1920x1080), то кол-во пикселей будет 2 073 600. Что соответствует кол-ву входных нейронов.

RGB. Цветовое пространство

Импорт и просмотр изображений

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

Загрузка и сохранение изображения

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

Особенности импорта изображения OpenCV

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).
2. 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

Стандартная форма

```
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

Буханочка до кадрирования

Буханочка после кадрирования

Кадрирование

```
import cv2
cropped = image[10:500, 500:2000]
viewImage(cropped, "Буханочка обрезалась")
```

Где **image[10:500, 500:2000]** — это **image[y:y + высота, x:x + ширина]**.

Изменение размера

Нормальная
буханочка

Маленькая
буханочка

Изменение размера

```
import cv2
scale_percent = 30 # Изменения размера в процентах
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, "Буханочка уменьшилась")
```

Поворот

Нормальная
буханочка

Перевернутая
буханочка

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, "Буханочка перевернулась")
```

Поворот. Примечание

`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.

Градация серого

Преобразует изображение в серое, с 256 оттенками серого, от 0 (чёрный) до 255 (белый)

Перевод в градацию серого

Нормальная
буханочка

Буханочка
посерела

Перевод в градацию серого

```
import cv2
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
viewImage(gray_image, "Буханочка посерела")
```

Перевод в черно-белое изображение по порогу

Нормальная
буханочка

Буханочка
почернела/побелела

Перевод в градацию серого

```
import cv2
ret, threshold_image = cv2.threshold(im, 127, 255, 0)
viewImage(threshold_image, "Чёрно-белая буханочка")
```

Размытие

Нормальная буханочка

Мутная буханка

Размытие

```
import cv2

blurred = cv2.GaussianBlur(image, (51, 51), 0)

viewImage(blurred, "Мутная буханка")
```

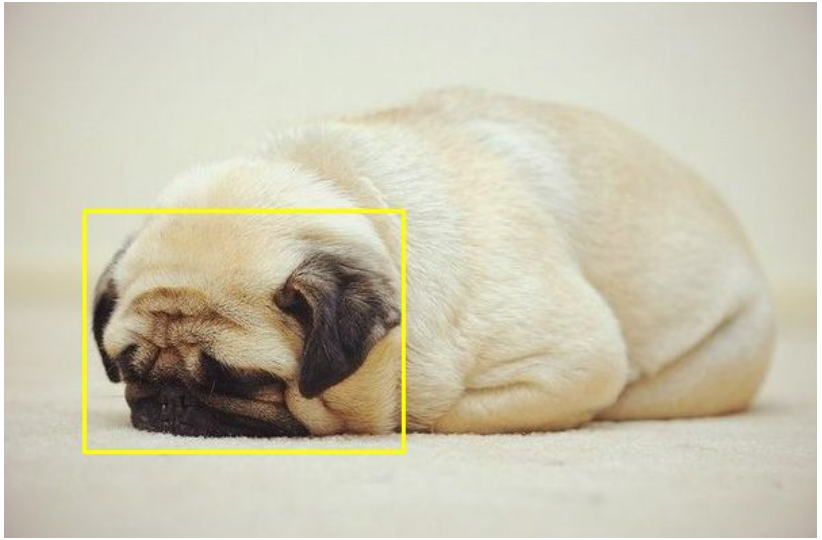
Размытие. Примечание

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

Нормальная буханочка



Буханочка в прямоугольнике

Рисование прямоугольников

```
import cv2
output = image.copy()
cv2.rectangle(output, (2600, 800), (4100, 2400), (0, 255, 255), 10)
viewImage(output, "Буханочка в прямоугольнике")
```

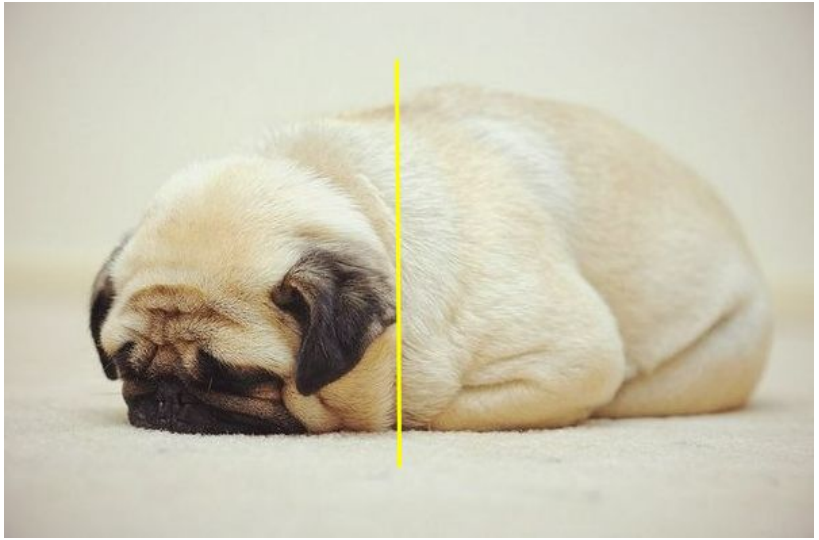
Рисование прямоугольников. Примечание

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 линии

Нормальная буханочка



Буханочка и линия

Рисование линии

```
import cv2
output = image.copy()
cv2.line(output, (60, 20), (400, 200), (0, 255, 255), 5)
viewImage(output, "Буханочка и линия")
```

Рисование линии. Примечание

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

Нормальная буханочка

“Буханочка”

Текст

```
import cv2
output = image.copy()
cv2.putText(output, "Буханочка", (1500, 3600),cv2.FONT_HERSHEY_SIMPLEX, 15, (255, 0, 0), 40)
viewImage(output, "Буханочка")
```

Текст. Примечание

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)
```