СӘТБАЕВ УНИВЕРСИТЕТІ



SATBAYEV UNIVERSITY

Основы искусственного интеллекта. Лекция 11. Компьютерное зрение. Основы компьютерного зрения, введение в библиотеки OpenCV и Pillow

Преподаватель: **Қасенхан Арай Мейрамбайқызы, доктор PhD, ассоциированный профессор кафедры «Программная инженерия»**

a.kassenkhan@satbayev.university

Содержание

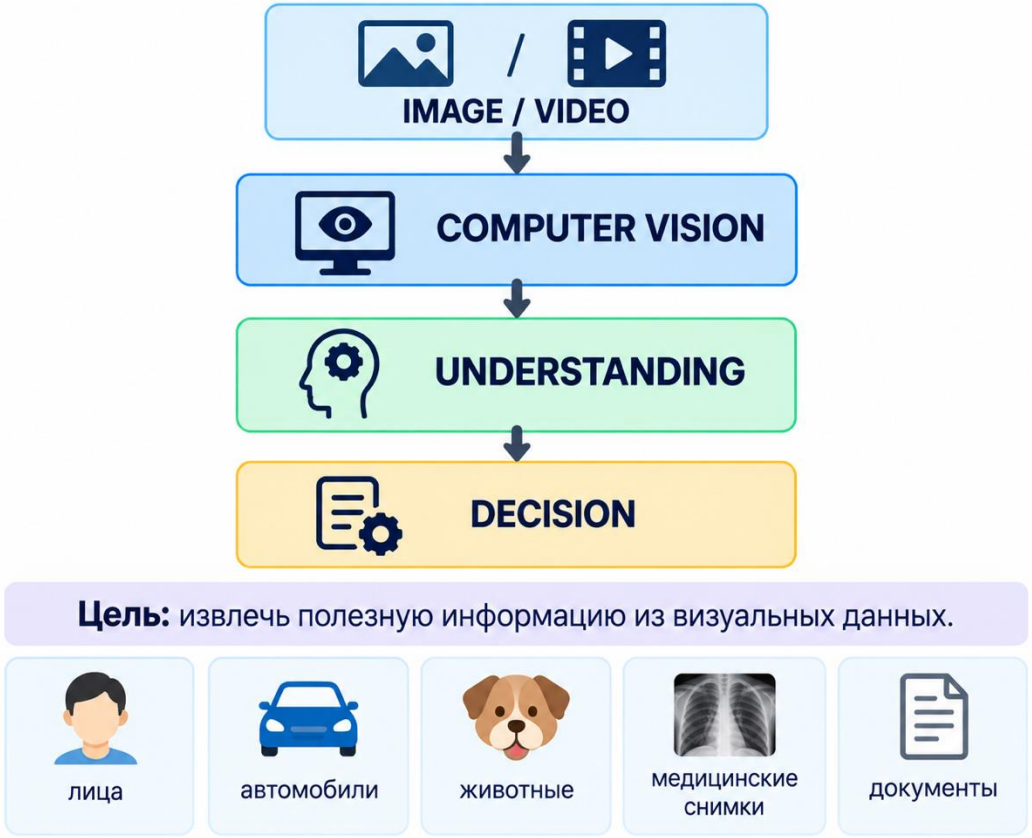
1. Что такое компьютерное зрение
2. Как компьютер видит изображение
3. Классификация
4. Детекция объектов
5. Сегментация
6. Распознавание лиц
7. OpenCV
8. Базовые преобразования
9. Детекция краёв
10. Pillow
11. OpenCV vs Pillow

Результаты обучения

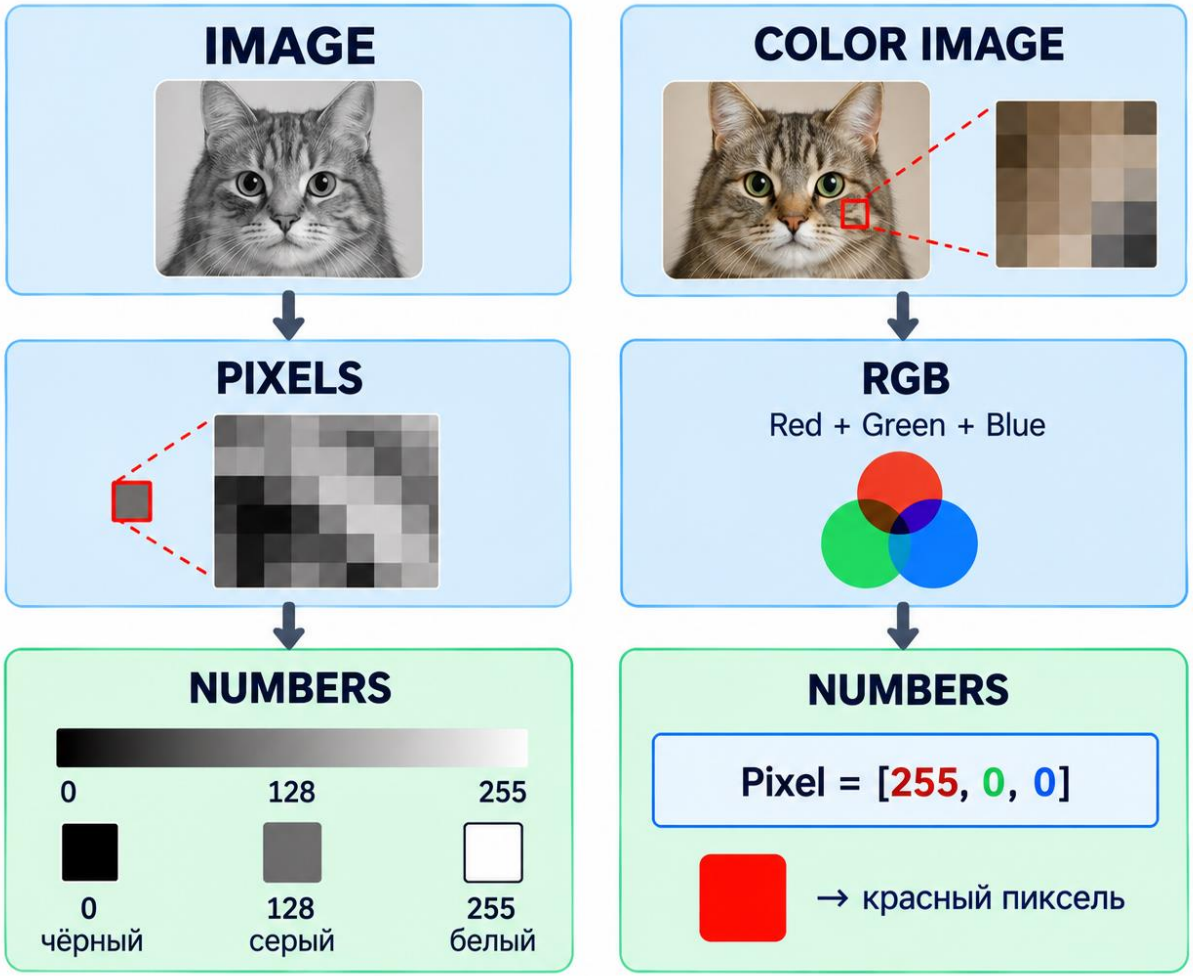
1. **Объяснить, что такое Computer Vision;**
2. **Различать основные задачи CV;**
3. **Понимать, как изображение представляется в компьютере;**
4. **Загружать и преобразовывать изображения;**
5. **Выполнять базовую обработку с OpenCV;**
6. **Понимать различия OpenCV и Pillow.**

Что такое Computer Vision?

Computer Vision — область искусственного интеллекта, которая позволяет компьютерам анализировать и интерпретировать изображения и видео.

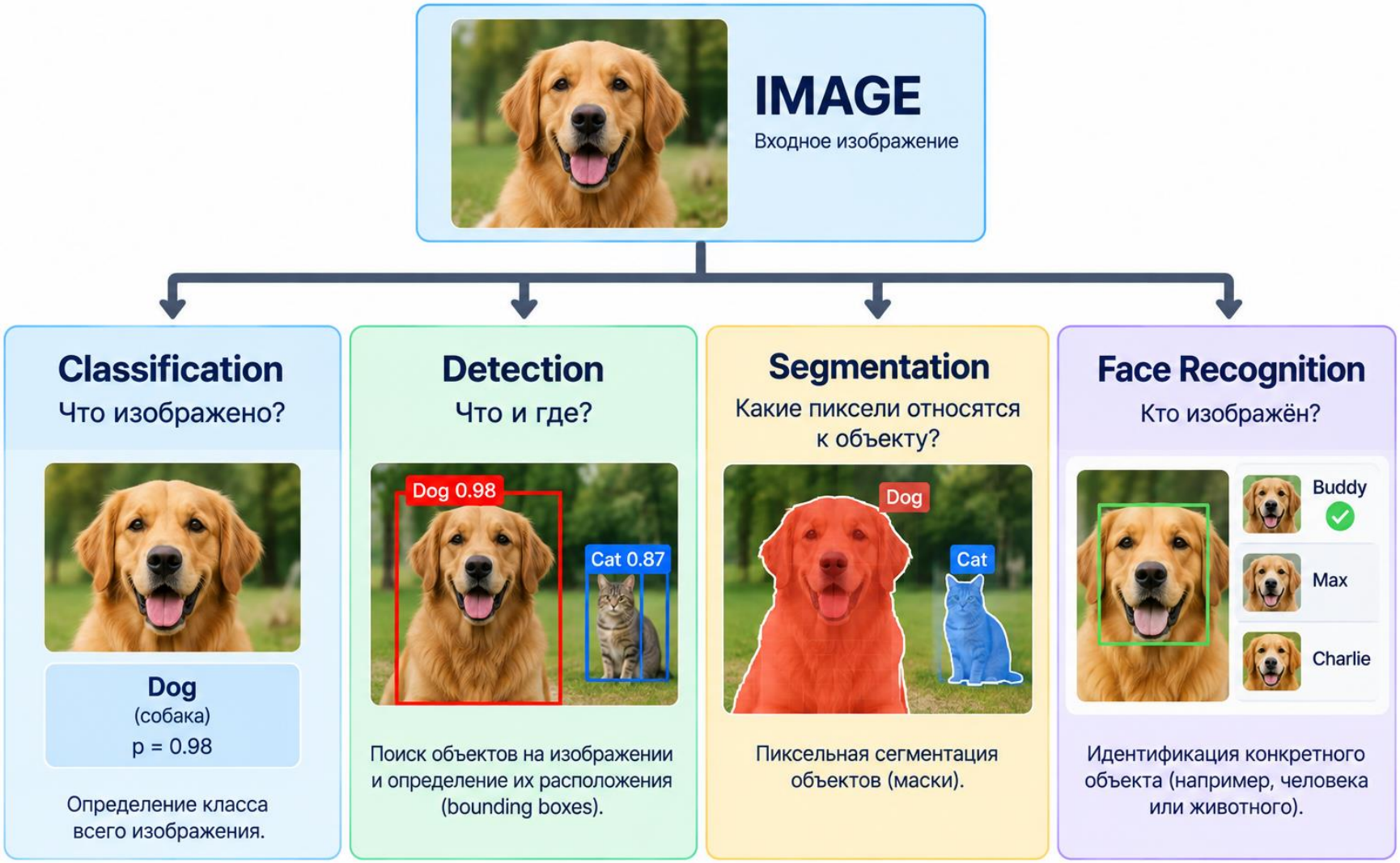


Как компьютер видит изображение?



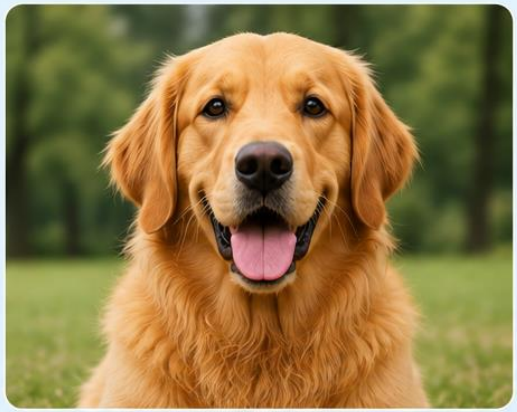
Для компьютера изображение — это массив чисел.

Основные задачи Computer Vision



Classification vs Object Detection

Classification

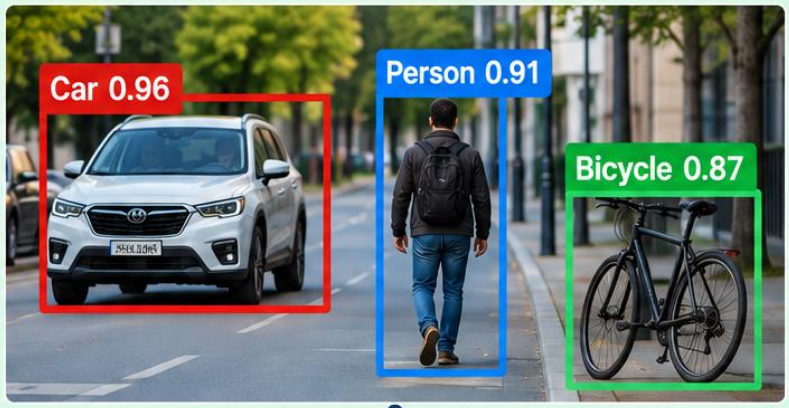


↓

DOG

Определяет класс всего изображения.

Object Detection



↓

Car	96%
Person	91%
Bicycle	87%

Определяет **что** находится на изображении и **где**.

Classification → WHAT? | **Detection → WHAT + WHERE?**

Image Segmentation

Classification

Что изображено?



Car
(автомобиль)

Определяет **класс всего изображения.**

Detection

Что и где?



Car	96%
Person	91%
Tree	87%

Определяет **что** находится на изображении и **где** (рамки вокруг объектов).

Segmentation

Какие пиксели относятся к объекту?



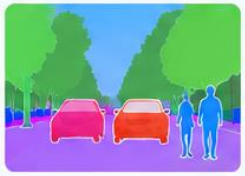
- автомобиль
- дерево
- человек
- дорога

Классифицирует изображение **на уровне пикселей.**



Semantic Segmentation

Все объекты одного класса получают одну категорию.



Все автомобили — один класс «автомобиль»

Instance Segmentation

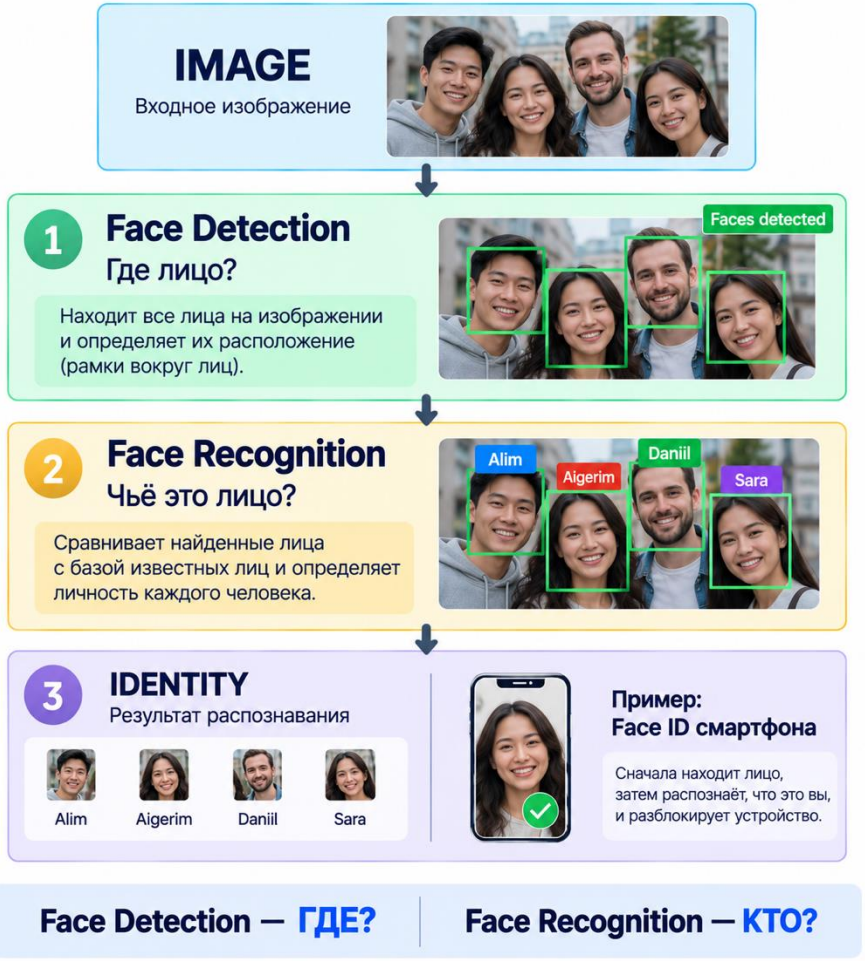
Каждый отдельный объект выделяется отдельно.



Каждый автомобиль — отдельный объект (своя маска).

Classification → WHAT? | **Detection → WHAT + WHERE?** | **Segmentation → WHICH PIXELS?**

Распознавание лиц



Сначала система обнаруживает лицо, затем сравнивает его признаки с сохранёнными данными.

OpenCV

OpenCV (Open Source Computer Vision Library)



Python-библиотека для:

- обработки изображений;
- обработки видео;
- детекции объектов;
- поиска контуров;
- компьютерного зрения в реальном времени.

Установка:

pip install opencv-python

Импорт:

import cv2

Загружаем изображение

Python (OpenCV)

```
import cv2

image = cv2.imread("cat.jpg")

print(image.shape)
```

Вывод:
(720, 1280, 3)

cat.jpg



Размер изображения:
1280 × 720 пикселей

image.shape

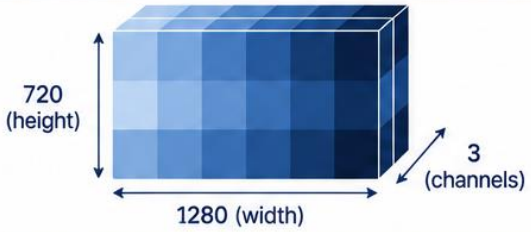
(720, 1280, 3)

720 → height (высота)

1280 → width (ширина)

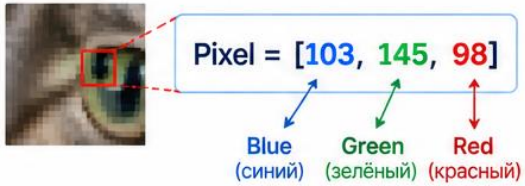
3 → color channels (цветовые каналы)

Изображение как массив чисел



Изображение представлено в виде трёхмерного массива размером (H, W, 3).

Значение одного пикселя



Каждый пиксель хранит значение трёх цветовых каналов **(B, G, R)** от 0 до 255.

Цветовые каналы (B, G, R)



Изображение состоит из трёх каналов: **Blue, Green** и **Red**.

Главная идея: OpenCV загружает изображение как NumPy-массив.

Изменяем размер изображения

Берём тот же cat.jpg, чтобы практика была последовательной.

Original Image
1280 × 720



`(720, 1280, 3)`

720 → height
1280 → width
3 → color channels

Python (OpenCV)

```python
import cv2

# изменяем размер изображения
resized = cv2.resize(
    image,
    (300, 300)
)
print(resized.shape)
```

Результат:

`(300, 300, 3)`

Resized Image
300 × 300



`(300, 300, 3)`

300 → height
300 → width
3 → color channels

Original Image
1280 × 720 → **cv2. resize()** → **300 × 300**

Мы изменили **размер изображения**, сохранив то же самое содержимое.

Цветное → Grayscale

Color Image
H × W × 3



(720, 1280, 3)

Цветовые каналы (BGR в OpenCV)



B (720, 1280) **G** (720, 1280) **R** (720, 1280)

Python (OpenCV)

```
import cv2

# преобразуем в оттенки серого
gray = cv2.cvtColor(
    image,
    cv2.COLOR_BGR2GRAY
)
```

Результат:

```
print(gray.shape)
```

```
(720, 1280)
```

Grayscale
H × W



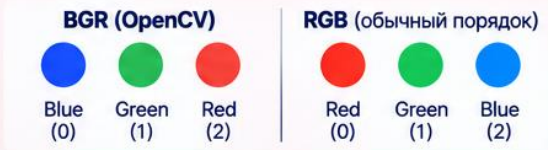
(720, 1280)

52	55	61	66	70
48	52	60	65	68
45	49	56	62	66
43	48	52	58	62
40	44	48	53	58



Каждый пиксель — одно значение от 0 (чёрный) до 255 (белый).

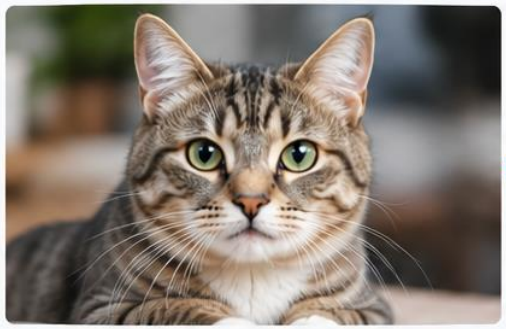
OpenCV по умолчанию использует порядок каналов BGR, а не RGB.
Если вы работаете с изображениями в RGB (например, из matplotlib или PIL), нужно учитывать порядок каналов.



Главная идея: cvtColor() изменяет цветовое пространство изображения.

Фильтрация изображения

INPUT IMAGE
(cat.jpg)



(720, 1280, 3)

Оригинальное изображение

Python (OpenCV)

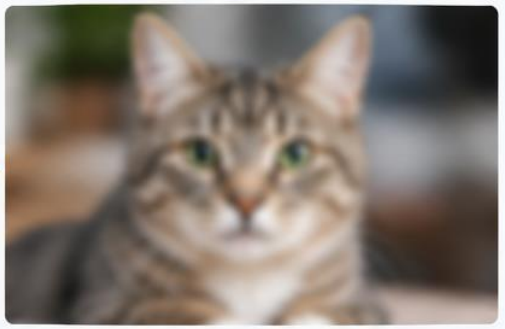
```python
import cv2

# применение гауссового размытия
blurred = cv2.GaussianBlur(
    image,
    (15, 15),    # размер ядра
    0            # стандартное отклонение
)
```

Параметры:

- **(15, 15)** — размер ядра (должен быть нечётным).
- **0** — стандартное отклонение (если 0, то вычисляется автоматически).

SMOOTHED IMAGE
(Gaussian Blur)



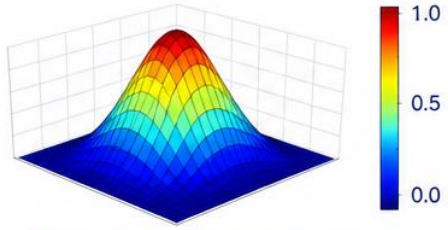
(720, 1280, 3)

Изображение после размытия

Что делает Gaussian Blur?

- Заменяет значение каждого пикселя взвешенным средним соседних пикселей (по гауссовому распределению).
- Уменьшает шум и мелкие детали.
- Используется как этап предварительной обработки (например, перед детекцией границ или объектов).

Пример гауссового ядра (15 × 15)



Гауссово распределение весов

Результат

- Изображение становится более гладким (сглаженным).
- Шум уменьшается.
- Края объектов остаются, но становятся менее резкими.
- Чем больше размер ядра, тем сильнее размытие.

Главная идея: Размытие уменьшает шум и может использоваться как этап предварительной обработки.

Детекция краёв

1. IMAGE
Оригинальное изображение (cat.jpg)



`(720, 1280, 3)`

Цветное изображение (H × W × 3)

2. GRAYSCALE
Преобразуем в оттенки серого



`(720, 1280)`

Одноканальное изображение (H × W)

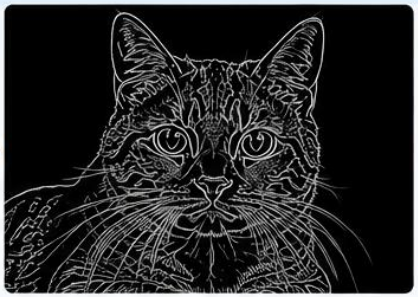
3. CANNY
Находим границы

```python
import cv2

gray = cv2.cvtColor(
    image,
    cv2.COLOR_BGR2GRAY
)

edges = cv2.Canny(
    gray,
    100,
    200
)
```

4. EDGES
Результат детекции границ



`(720, 1280)`

Изображение границ (H × W)

Как это работает?

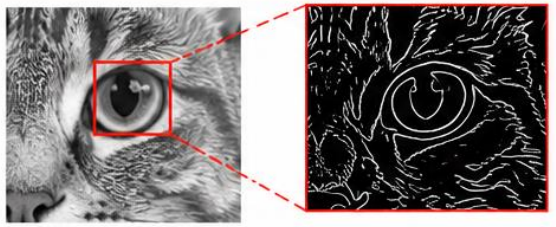
1. Преобразуем изображение в оттенки серого.
2. Применяем алгоритм Canny с пороговыми значениями (100, 200).
3. Получаем изображение, на котором выделены границы объектов (контуры).

Параметры Canny

100 — **нижний порог** (min threshold)
Пиксели с градиентом ниже этого значения не считаются границами.

200 — **верхний порог** (max threshold)
Пиксели с градиентом выше этого значения считаются границами. Значения между порогами обрабатываются дополнительно.

Пример результата (увеличение)



Главная идея:

Детектор Canny находит области изображения с резким изменением яркости — границы объектов.

Pillow

Pillow (Python Imaging Library) - это современная версия библиотеки PIL. Библиотека Python для простой работы с изображениями.



cat.jpg
(1280 × 720)

Пример кода (Pillow) Python

```python
from PIL import Image

image = Image.open("cat.jpg")
image = image.resize((300, 300))
image.save("cat_small.jpg")
```

Всего несколько строк кода — и вы уже можете читать, изменять и сохранять изображения!

1 Open
Открыть изображение



```python
image = Image.open(
  "cat.jpg")
```

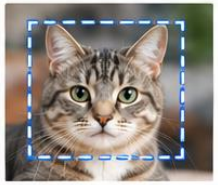
2 Resize
Изменить размер



300 × 300

```python
image = image.resize(
  (300, 300))
```

3 Crop
Обрезать часть изображения



```python
cropped = image.crop(
  (x1, y1, x2, y2))
```

4 Rotate
Повернуть изображение



```python
rotated = image.rotate(
  45)
```

5 Save
Сохранить результат



cat_small.jpg

```python
image.save(
    "cat_small.jpg")
```

Основные задачи с Pillow:

Open → Resize → Crop → Rotate → Save

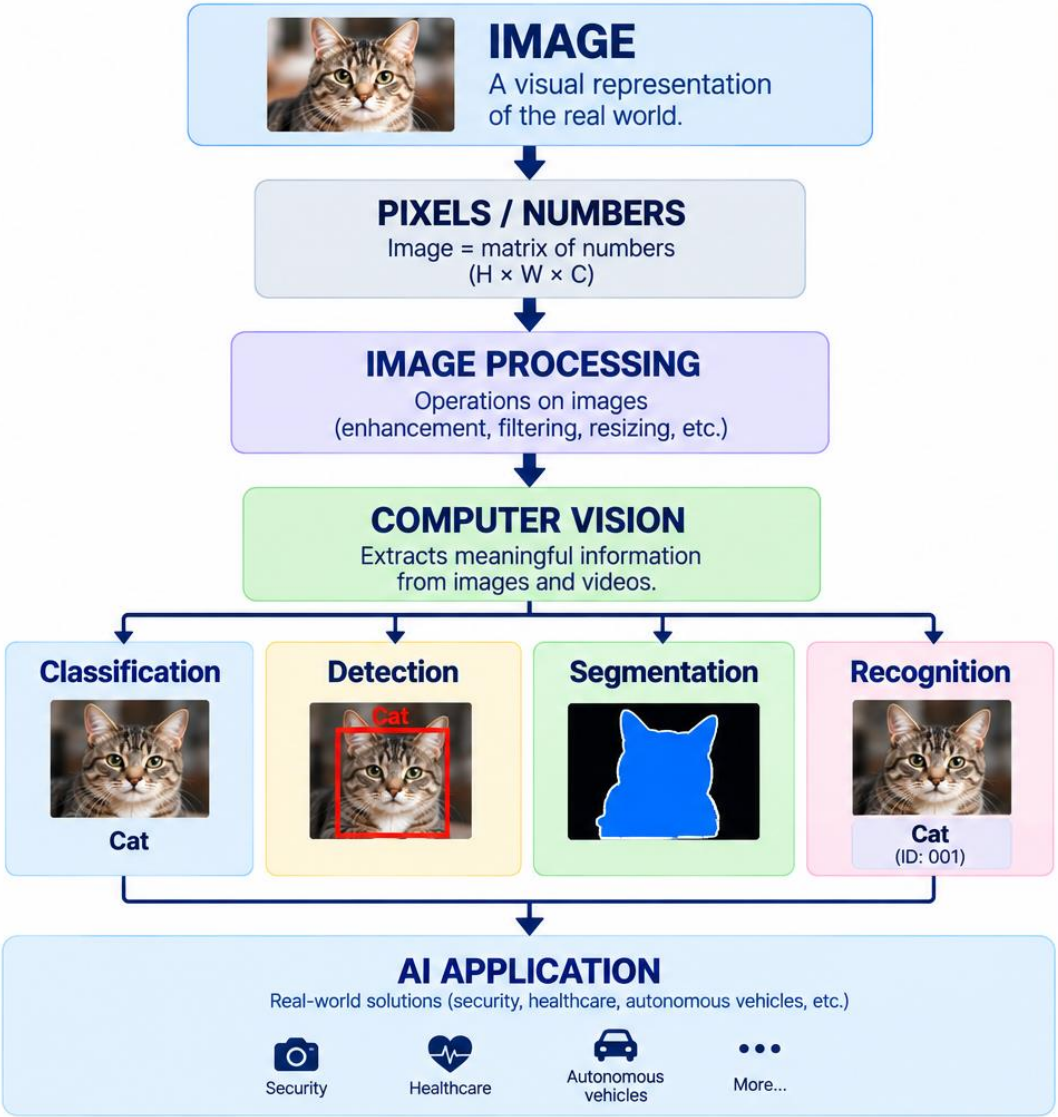
OpenCV или Pillow?

OpenCV	Pillow
Computer Vision	Работа с изображениями
Image processing	Resize / Crop / Rotate
Video	Конвертация форматов
Detection	Добавление текста
Real-time	Простые преобразования

Computer Vision → OpenCV
Simple Image Processing → Pillow

В одном проекте библиотеки могут использоваться вместе.

Заключение



OpenCV

- → Computer Vision
- → Image / Video processing

Pillow

- → Basic image processing

4 ключевые задачи компьютерного зрения

1. **Classification** → что?
2. **Detection** → что и где?
3. **Segmentation** → какие пиксели?
4. **Recognition** → кто?