Технологии компьютерного зрения

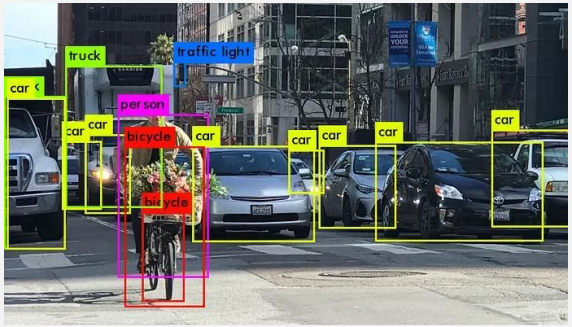
Что такое детекция объектов?

Детекция объектов – процесс нахождения объектов на изображении или в видео и определение их классов.

Основные задачи:

✔ **Локализация** – определение координат объектов 🏷️

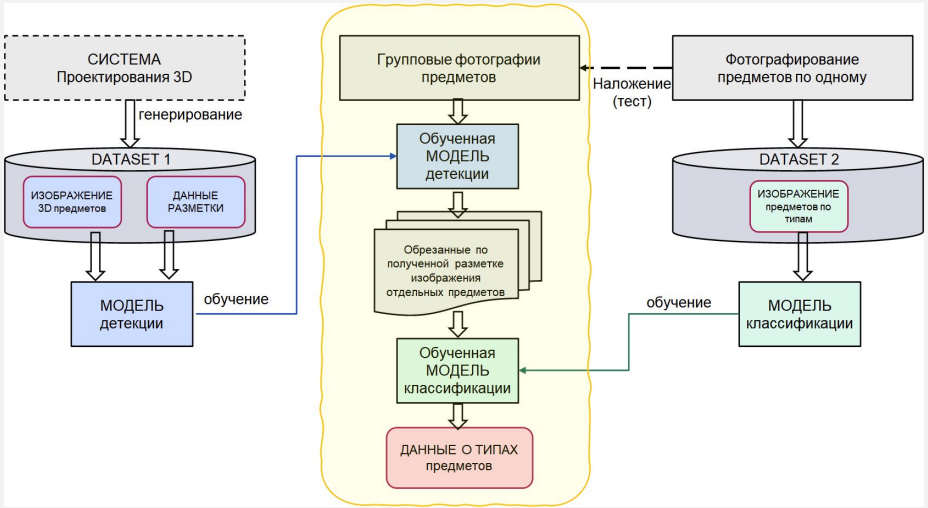
✔ **Классификация** – определение типа объекта 📦



Разница между классификацией и детекцией

Классификация: Что находится на изображении?

Детекция: Где находится объект и к какому классу он относится?



Что такое компьютерное зрение?

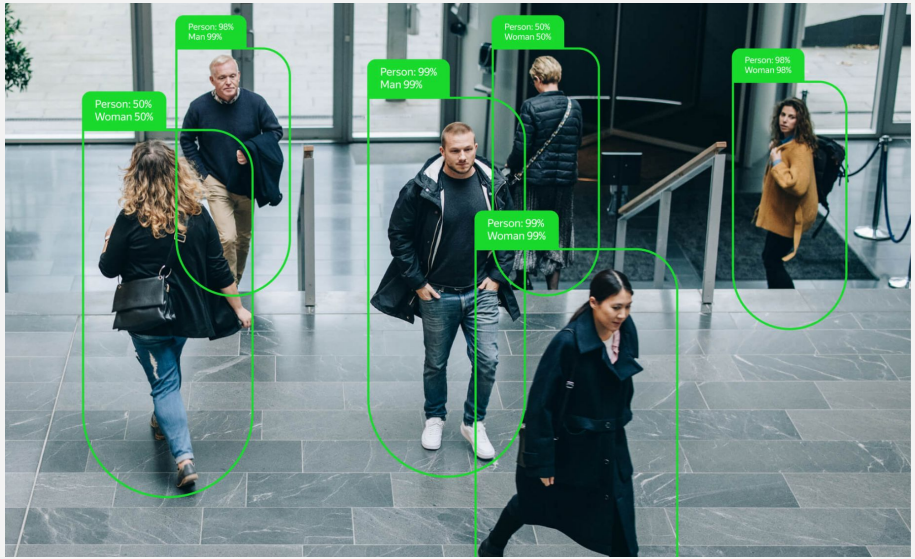
Компьютерное зрение (Computer Vision) – это область искусственного интеллекта, которая позволяет компьютерам извлекать, анализировать и интерпретировать визуальную информацию из изображений и видео.

- **Зачем оно нужно?**

Используется для автоматической обработки изображений, распознавания объектов, анализа видео, навигации автономных систем.

Применение

- Распознавание лиц и объектов
- Автономные автомобили
- Видеонаблюдение
- Робототехника
- Медицина



Основные этапы обработки изображений в компьютерном зрении

1. Получение изображения

— Камеры, датчики, видеопотоки

2. Предобработка изображения

— Грейскейл (Grayscale)

— Нормализация

— Фильтрация шума

— Выравнивание гистограммы

Основные этапы обработки изображений в компьютерном зрении

3. Выделение признаков

— Края (Edge Detection)
— Контуры (Contours)
— Углы (Corners)

4. Классификация и распознавание объектов

— Алгоритмы машинного обучения
— Нейронные сети

5. Интерпретация и принятие решений

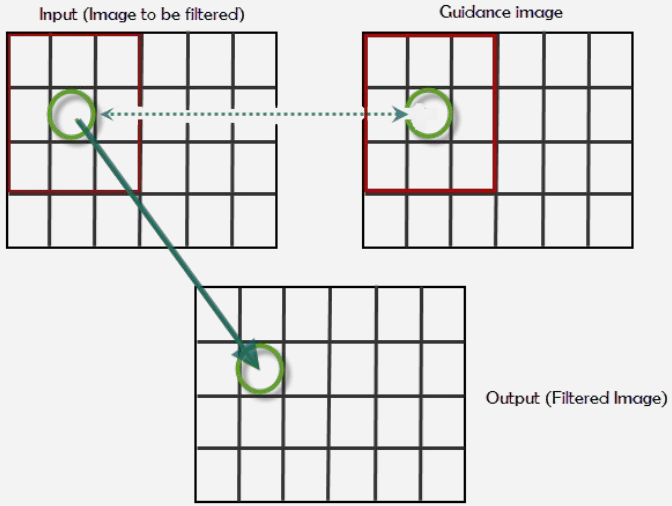
— Использование результатов для автоматизации

Основные алгоритмы обработки изображений

- **Фильтрация изображений (Image Filtering)**
- **Выделение особенностей (Feature Extraction)**
- **Сегментация изображений (Image Segmentation)**
- **Глубокие нейросети (Deep Learning)**

Фильтрация изображений (Image Filtering)

- **Gaussian Blur** – Размытие для удаления шумов
- **Median Filtering** – Удаление шума, устойчив к импульсным помехам
- **Sobel, Canny** – Выделение границ объектов



Пример кода для фильтрации изображений

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

image = cv2.imread("image.jpg",
cv2.IMREAD_GRAYSCALE)

gaussian_blur = cv2.GaussianBlur(image, (5,5), 0)
median_blur = cv2.medianBlur(image, 5)
edges_sobel = cv2.Sobel(image, cv2.CV_64F, 1, 0,
ksize=5)
edges_canny = cv2.Canny(image, 50, 150)
```

Пример кода для фильтрации изображений

```
plt.figure(figsize=(10,5))
plt.subplot(2,2,1), plt.imshow(gaussian_blur,
cmap='gray'), plt.title("Gaussian Blur")
plt.subplot(2,2,2), plt.imshow(median_blur,
cmap='gray'), plt.title("Median Filtering")
plt.subplot(2,2,3), plt.imshow(edges_sobel,
cmap='gray'), plt.title("Sobel Edges")
plt.subplot(2,2,4), plt.imshow(edges_canny,
cmap='gray'), plt.title("Canny Edges")
plt.show()
```

Выделение особенностей (Feature Extraction)

Процесс извлечения информативных признаков из изображения.

Популярные методы выделения особенностей:

✔ **SIFT (Scale-Invariant Feature Transform)** – находит ключевые точки, устойчивые к масштабированию

✔ **SURF (Speeded Up Robust Features)** – более быстрый аналог SIFT

✔ **ORB (Oriented FAST and Rotated BRIEF)** – быстрый и эффективный

✔ **HOG (Histogram of Oriented Gradients)** – используется для детекции объектов

Выделение ключевых точек методом ORB в OpenCV

```
import cv2

image = cv2.imread("image.jpg", cv2.IMREAD_GRAYSCALE)
orb = cv2.ORB_create()

keypoints, descriptors = orb.detectAndCompute(image, None)

image_with_keypoints = cv2.drawKeypoints(image, keypoints,
None, color=(0,255,0))

cv2.imshow("ORB Features", image_with_keypoints)
cv2.waitKey(0)          cv2.destroyAllWindows()
```

Сегментация изображений (Image Segmentation)

- **Пороговая сегментация (Thresholding)** – Простое разделение пикселей по яркости
- **Адаптивная пороговая сегментация** – Улучшенный вариант для неоднородного фона
- **Разделение на регионы (Region-based Segmentation)** – Выделение объектов по признакам
- **Методы кластеризации (K-Means, Watershed)** – Продвинутые алгоритмы сегментации

Пример кода для пороговой сегментации

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

image = cv2.imread("image.jpg",
cv2.IMREAD_GRAYSCALE)

_, thresh = cv2.threshold(image, 127, 255,
cv2.THRESH_BINARY)
```

Пример кода для пороговой сегментации

```
adaptive_thresh = cv2.adaptiveThreshold(image, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,

cv2.THRESH_BINARY, 11, 2)

plt.figure(figsize=(10,5))
plt.subplot(1,2,1), plt.imshow(thresh, cmap='gray'),
plt.title("Simple Thresholding")
plt.subplot(1,2,2), plt.imshow(adaptive_thresh,
cmap='gray'), plt.title("Adaptive Thresholding")
plt.show()
```

Глубокие нейросети (Deep Learning)

Разновидность машинного обучения, использующая многослойные нейронные сети. Позволяет моделировать сложные зависимости в данных.

Чем глубокие сети отличаются от обычных нейронных сетей?

✔ Большее количество скрытых слоев

✔ Способность извлекать сложные иерархические признаки

✔ Обучаются на больших объемах данных с использованием мощных GPU

Обнаружение границ на изображениях

- **Оператор Собеля (Sobel Operator)** – выявляет градиенты изображения
- **Оператор Кэнни (Canny Edge Detector)** – более точный алгоритм
- **Методы Лапласиана (Laplacian)** – выявление изменений яркости
- **Применение:** распознавание объектов, анализ контуров

Пример кода для обнаружения границ

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

image = cv2.imread("image.jpg", cv2.IMREAD_GRAYSCALE)

sobelx = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobely = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
laplacian = cv2.Laplacian(image, cv2.CV_64F)
canny = cv2.Canny(image, 100, 200)
```

Пример кода для обнаружения границ

```
plt.figure(figsize=(12,6))
plt.subplot(2,2,1),    plt.imshow(sobelx,    cmap='gray'),
plt.title("Sobel X")
plt.subplot(2,2,2),    plt.imshow(sobely,    cmap='gray'),
plt.title("Sobel Y")
plt.subplot(2,2,3),   plt.imshow(laplacian,   cmap='gray'),
plt.title("Laplacian")
plt.subplot(2,2,4),     plt.imshow(canny,     cmap='gray'),
plt.title("Canny")
plt.show()
```

Детекция объектов (Object Detection)

Определение местоположения и классов объектов на изображении. Используется в видеонаблюдении, автономных автомобилях, медицинской диагностике.

Популярные алгоритмы:

✔ **Haar Cascade Classifier** – метод на основе признаков

✔ **YOLO (You Only Look Once)** – глубокая нейронная сеть для быстрого обнаружения

✔ **Faster R-CNN** – мощная, но требовательная к ресурсам модель

Пример кода детекции объектов с использованием YOLO в OpenCV

```
import cv2
import numpy as np

net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
layer_names = net.getUnconnectedOutLayersNames()

with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]

image = cv2.imread("image.jpg")
height, width = image.shape[:2]

blob = cv2.dnn.blobFromImage(image, 1/255.0, (416, 416), swapRB=True,
crop=False)
net.setInput(blob)
outputs = net.forward(layer_names)
```

Пример кода детекции объектов с использованием YOLO в OpenCV

```
for output in outputs:
    for detection in output:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5:  # Порог уверенности
            center_x, center_y, w, h = (detection[:4] * [width,
height, width, height]).astype("int")
       x, y = int(center_x - w / 2), int(center_y - h / 2)
cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(image, classes[class_id], (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

cv2.imshow("Object Detection", image)
cv2.waitKey(0)        cv2.destroyAllWindows()
```

Распознавание лиц

Технология, использующая алгоритмы компьютерного зрения для идентификации и верификации лиц. Основано на извлечении уникальных характеристик лица.

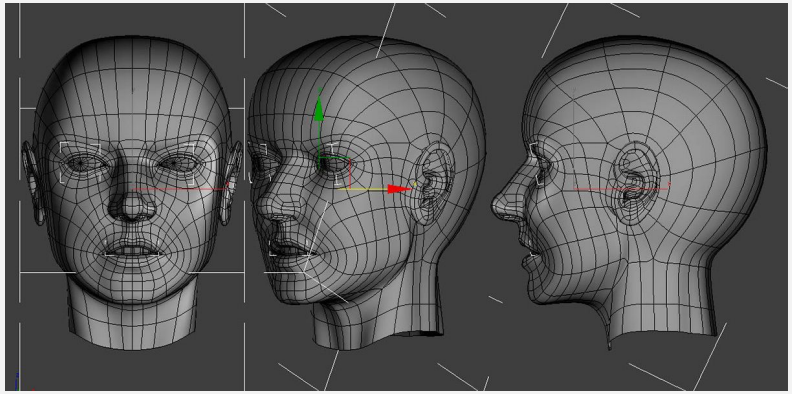
Как работает распознавание лиц?

📸 **Обнаружение лица** – определение области с лицом на изображении

🧩 **Выделение признаков** – анализ формы глаз, носа, рта, расстояния между элементами

🔍 **Сравнение с базой данных** – сопоставление с известными лицами

Распознавание лиц



Популярные алгоритмы и модели

- ✔ Haar Cascades – метод на основе каскадных классификаторов
- ✔ HOG (Histogram of Oriented Gradients) + SVM
- ✔ Deep Learning (FaceNet, DeepFace, ArcFace)

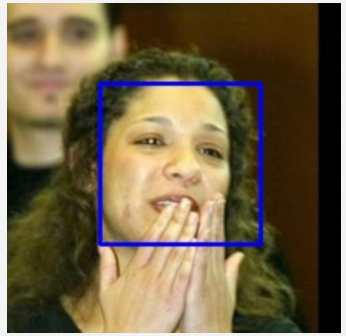
Распознавание лиц с использованием OpenCV и dlib

```
import cv2
import dlib

detector = dlib.get_frontal_face_detector()

image = cv2.imread("face.jpg")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

faces = detector(gray)
```



Распознавание лиц с использованием OpenCV и dlib

```
for face in faces:
        x,  y,  w,  h  =  face.left(),  face.top(),
face.width(), face.height()
    cv2.rectangle(image, (x, y), (x + w, y + h), (0,
255, 0), 2)

cv2.imshow("Detected Faces", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Ограничения технологий детекции объектов

1️⃣ **Требование больших объемов данных**

✔ Нейросети требуют тысячи размеченных изображений для обучения.

✔ Создание таких датасетов – дорогостоящий и трудоемкий процесс.

2️⃣ **Высокая вычислительная сложность**

✔ Глубокие модели требуют мощных GPU.

✔ В реальном времени могут работать только оптимизированные архитектуры, такие как YOLO.

Ограничения технологий детекции объектов

3️⃣ Проблемы с детекцией в сложных условиях

✔ Плохое освещение, погодные условия, шум мешают работе алгоритмов.

✔ Камеры с низким разрешением могут ухудшить качество предсказаний.

4️⃣ Уязвимость к атакам и ложным детекциям

✔ Малейшие изменения в изображении могут сбить алгоритм (Adversarial attacks).

✔ Возможны ошибки: неверная классификация, отсутствие детекции.