Системы мониторинга окружающей среды: компьютерное зрение и обработка изображений (Базовые алгоритмы OpenCV, детекция объектов, сегментация, распознавание разметки и знаков)

Основные задачи CV в ADAS включают обнаружение объектов (пешеходов, автомобилей, препятствий), сегментацию сцен (разделение на регионы, такие как дорога, небо, трава), распознавание дорожной разметки (линии, полосы) и знаков (стоп, скорость). Мы рассмотрим базовые алгоритмы библиотеки OpenCV, а также продвинутые методы детекции, сегментации и распознавания.

Библиотека OpenCV (Open Source Computer Vision Library) - это открытая платформа для обработки изображений и видео, написанная на C++, но с bindings для Python, что делает её удобной для прототипирования. Она используется в реальных ADAS-системах, таких как в Tesla или Mobileye.

Базовые алгоритмы OpenCV

Начнём с основ. Обработка изображений начинается с загрузки и базовых операций. Изображение представляется как матрица пикселей (RGB или grayscale).

Загрузка и отображение изображений

В Python с OpenCV:

```python
import cv2

img = cv2.imread('image.jpg')  # Загрузка изображения
cv2.imshow('Image', img)       # Отображение
cv2.waitKey(0)                 # Ожидание нажатия клавиши
cv2.destroyAllWindows()
```

Фильтры и преобразования

Для улучшения качества изображений применяются фильтры:

•**Гауссово размытие** (Gaussian Blur) для снижения шума:

```python
blurred = cv2.GaussianBlur(img, (5, 5), 0)
```

Обнаружение краёв с помощью Canny:

```python
edges = cv2.Canny(img, 100, 200)
```

Это полезно для выделения контуров дорог или знаков в условиях плохой освещённости.

Цветовые пространства

Часто переводим в HSV (Hue, Saturation, Value) для лучшего выделения цветов, например, жёлтой разметки:

```python
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

Эти базовые операции - фундамент для более сложных задач в мониторинге окружающей среды

Детекция объектов

Детекция объектов - это задача по нахождению и классификации объектов на изображении, обычно с bounding boxes (прямоугольниками вокруг объектов). В ADAS это критично для обнаружения пешеходов, велосипедистов или других транспортных средств.

Классические методы

OpenCV предоставляет Haar Cascades - каскадные классификаторы на основе признаков Хаара. Они быстрые, но менее точные в сложных сценах. Пример детекции лиц (аналогично для знаков):

```python
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
faces = face_cascade.detectMultiScale(gray_img, 1.1, 4)
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
```

Современные методы

Для реального времени используются нейронные сети, такие как YOLO (You Only Look Once) или SSD. OpenCV поддерживает их через DNN модуль. Пример с YOLO:

Python

```python
net = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
net.setInput(blob)
outs = net.forward(output_layers)
```

В ADAS такие алгоритмы помогают избежать столкновений, анализируя видео с камер.
Вот пример детекции объектов в сцене дорожного движения с использованием компьютерного зрения в ADAS:



Сегментация

Сегментация - это разделение изображения на сегменты (пиксели с похожими свойствами). В дорожном движении это используется для выделения проезжей части, пешеходных зон или препятствий.

Типы сегментации

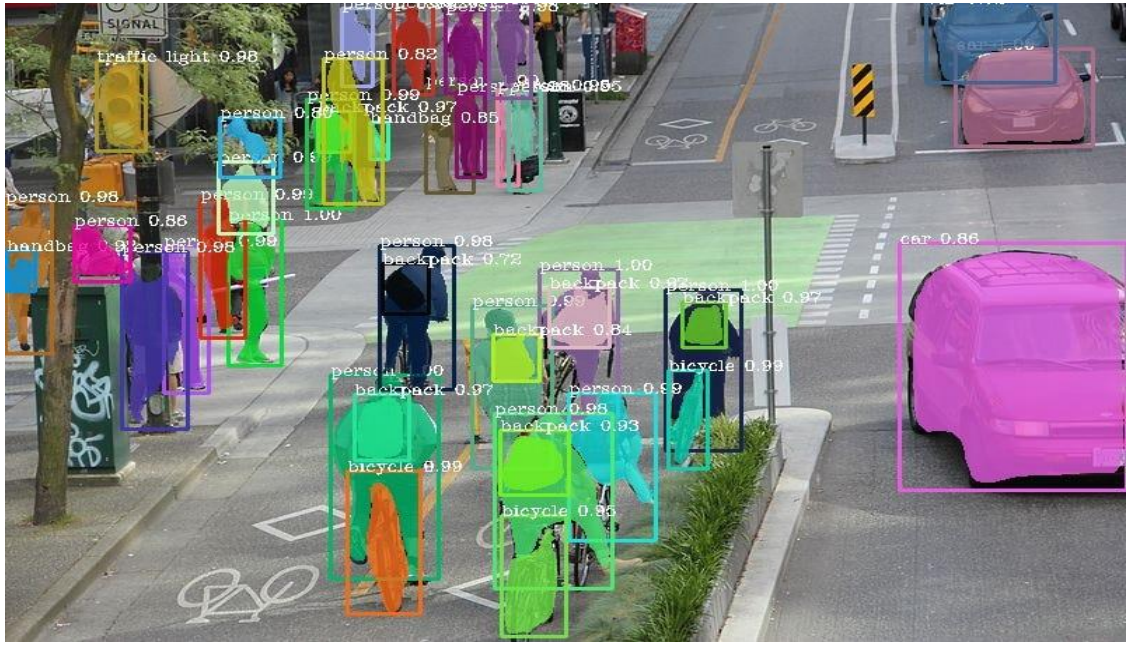
•**Семантическая сегментация**: Присваивает класс каждому пикселю (дорога, небо, машина).

•**Инстанс сегментация**: Разделяет объекты одного класса (каждую машину отдельно).

Для продвинутого - сети вроде Mask R-CNN или U-Net, интегрируемые в OpenCV.

Сегментация помогает в понимании сцены, например, для автономного вождения в плохую погоду.

Пример сегментации в сцене трафика:



Распознавание разметки и знаков

Распознавание дорожной разметки

Обнаружение линий (lane detection) - ключ для Lane Departure Warning. Используем Hough Transform в OpenCV для поиска прямых линий после Canny:

Python

```
lines = cv2.HoughLinesP(edges, 1, np.pi/180, 100, minLineLength=100, maxLineGap=10) for line in lines: x1, y1, x2, y2 = line[0] cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
```

Для кривых линий - полиномиальная аппроксимация.

Распознавание знаков

Traffic Sign Recognition (TSR) сочетает детекцию и классификацию. Сначала детектируем круглые/треугольные формы, затем классифицируем с CNN. В OpenCV:

•Детекция кругов: cv2.HoughCircles()

•Классификация: DNN с предобученной моделью.

В ADAS это интегрируется с GPS для соблюдения правил.

Пример распознавания разметки и знаков в автономных транспортных средствах:

