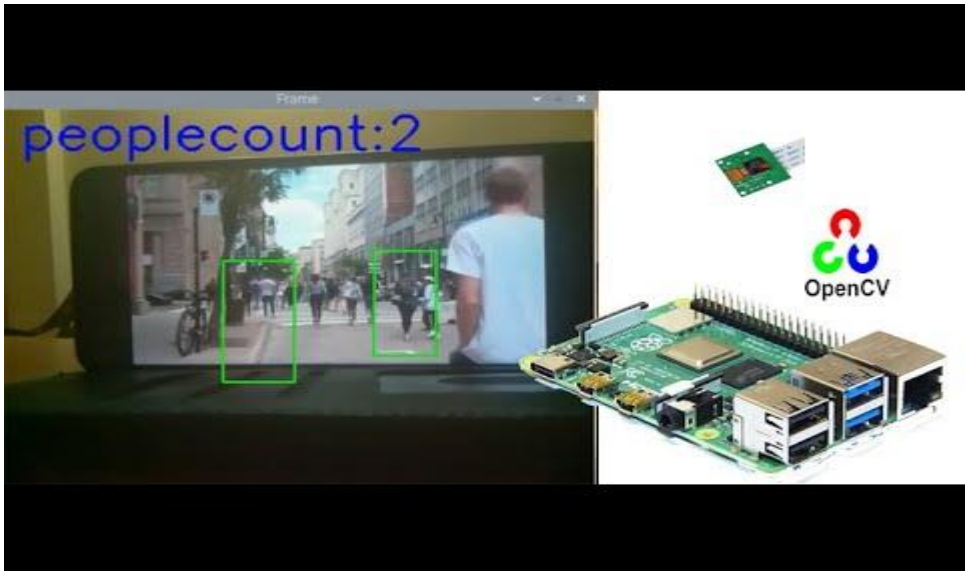
ПРАКТИЧЕСКИЕ НАВЫКИ

ТЕМА **Детектирование объектов и отслеживание движения. Технологии детектирования объектов и отслеживания движения на платформе Raspberry Pi. Алгоритмы отслеживания движения объектов с использованием OpenCV.**

Цель работы: Освоить методы регистрации и распознавания объектов с использованием библиотеки OpenCV: захват изображений объектов, их сохранение в базе, а затем распознавание в реальном времени с камеры.

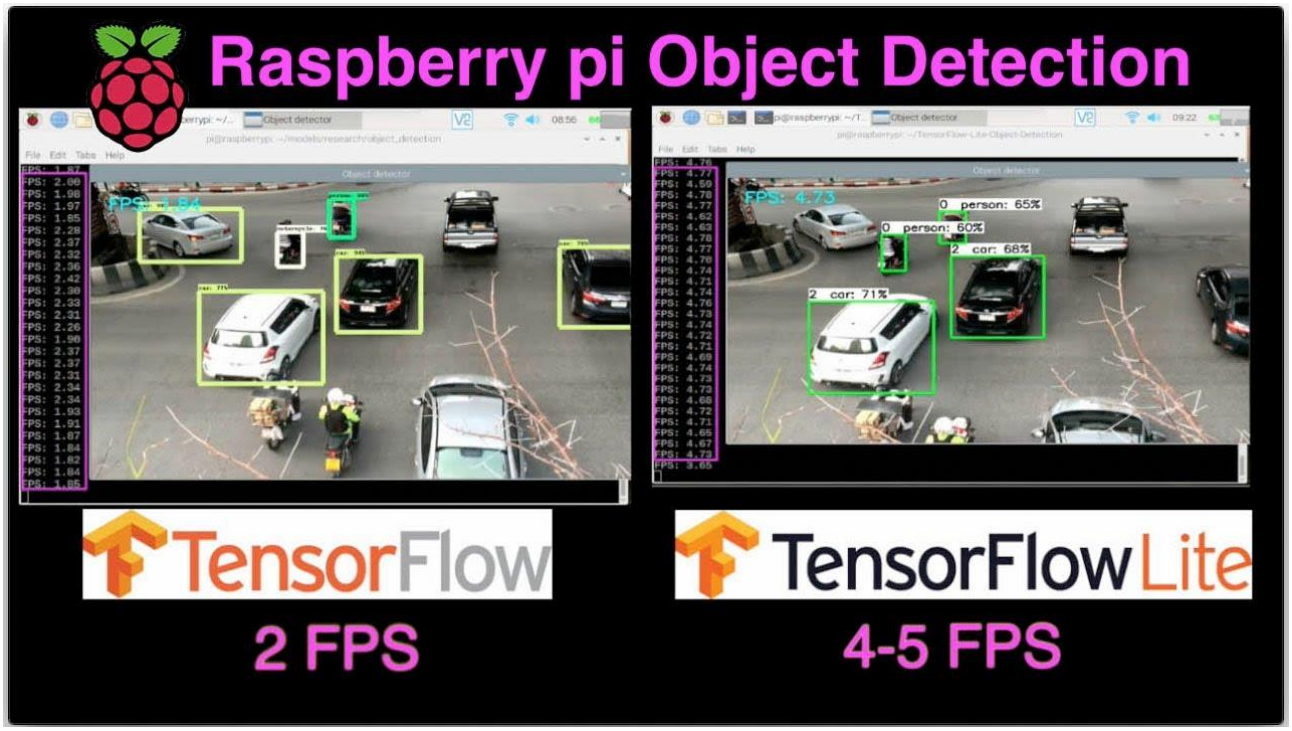
Теоретические основы темы.

Обнаружение объекта – это процесс нахождения объектов на изображении или в видеопотоке и определения их местоположения. На платформе Raspberry Pi это можно реализовать с помощью Haar cascade classifier или нейронных сетей (MobileNet-SSD, Tiny-YOLO).



Отслеживание объекта – это процесс наблюдения за траекторией уже обнаруженного объекта на последующих кадрах. В библиотеке OpenCV широко применяются трекеры KCF, CSRT и MOSSE.

Обнаружение движения – осуществляется путём вычисления различий между последовательными кадрами (frame differencing), с помощью вычитания фона (background subtraction) или анализа оптического потока (optical flow).



Практическая часть

В первую очередь выбирается необходимая среда для работы языка программирования Python.Для выполнения задания устанавливаются необходимые библиотеки Python. Для обнаружения объектов в Python устанавливается библиотека OpenCV.

```
pip3 install opencv-python
```

Подключается Camera Module или USB-камера, и с помощью OpenCV осуществляется захват видеопотока.

```
import cv2
```

```
cap = cv2.VideoCapture(0)
```

На следующем этапе формируется небольшой датасет. Для этого несколько объектов фотографируются, и система автоматически сохраняет их с именами.

```
import cv2
import os
data_path = "objects_db"
if not os.path.exists(data_path):
    os.makedirs(data_path)
cap = cv2.VideoCapture(0)
print(">>> 3 ta obyektni rasmga olish uchun tayyorlaning.")
for i in range(1, 4):
    input(">>> {}-obyektni kameraga qo'ying va ENTER bosing...".format(i))
    ret, frame = cap.read()
    obj_path = os.path.join(data_path, f"object_{i}.jpg")
    cv2.imwrite(obj_path, frame)
    print(f"{i}-obyekt saqlandi: {obj_path}")
print(">>> Obyektlar bazaga qo'shildi ✓")
orb = cv2.ORB_create()
object_images = []
object_kp_des = []
for filename in os.listdir(data_path):
    img = cv2.imread(os.path.join(data_path, filename), 0)
    kp, des = orb.detectAndCompute(img, None)
    object_images.append((filename, img))
    object_kp_des.append((kp, des))
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
print(">>> Tanib olish jarayoni boshlandi. 'q' ni bosing to'xtatish uchun.")

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    kp_frame, des_frame = orb.detectAndCompute(gray, None)
    if des_frame is None:
        cv2.imshow("Real Time Detection", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
        continue
    best_match_name = None
```

```
    best_score = 0
    for idx, (kp_obj, des_obj) in enumerate(object_kp_des):
        if des_obj is None:
            continue
        matches = bf.match(des_obj, des_frame)
        matches = sorted(matches, key=lambda x: x.distance)
        score = len([m for m in matches if m.distance < 60])  # mosliklar soni
        if score > best_score:
            best_score = score
            best_match_name = object_images[idx][0]
    if best_match_name and best_score > 15:
        cv2.putText(frame, f"Topildi: {best_match_name}", (20, 40),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    else:
        cv2.putText(frame, "Aniqlanmadi", (20, 40),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
    cv2.imshow("Real Time Detection", frame)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break
cap.release()
cv2.destroyAllWindows()
```