1 Introduction

The goal of this project is to develop a program that recognizes and tracks pedestrians and exports a video of the results. The key function implemented in this project is to stably track pedestrians in different scenarios and record traces of pedestrian movement.

2 Methodologies

The methodologies employed in this project consists of several steps:

1. Background Subtraction: Initially, the background subtraction technique is applied to the input video frames using the K-nearest neighbors (KNN) algorithm to extract foreground objects. The process begins with initializing the subtractor through the "cv2.createBackgroundSubtractorKNN()" function, which creates a background subtractor object maintaining a statistical model of the background and utilizing it to identify foreground objects in the current frame. Subsequently, the subtractor is applied to each frame of the video to generate a binary mask, where foreground pixels are marked as white and background pixels as black. The binary mask is then thresholded to refine the segmentation results by removing noise and retaining only significant foreground objects. To further enhance the segmentation results, additional operations such as erosion and dilation are performed on the thresholded binary mask. Erosion eliminates isolated foreground pixels, while dilation fills gaps in the foreground regions, resulting in more continuous foreground objects. These steps effectively extract foreground objects from the video sequence, ensuring reliable input data for subsequent pedestrian tracking.

2. Contour Detection: In the first frame of the video sequence, contour detection algorithms are employed to identify the contours of foreground objects. Subsequently, bounding boxes are generated around the detected pedestrian contours. To facilitate individual pedestrian tracking, a unique identifier (ID) is assigned to each bounding box. The identified contours are then filtered based on their area and aspect ratio criteria to isolate regions that are likely to represent pedestrians.

3. Centroid Tracking: The centroid of each pedestrian bounding box is monitored across consecutive frames to estimate pedestrian movement and trajectory.

4. Motion Analysis: Firstly, a dictionary named "prev_pedestrians" is created to store the historical positions of pedestrians detected in previous frames. In each subsequent frame, the center positions of pedestrians detected in the current frame are stored in the "prev_pedestrians" dictionary. The code then iterates through the contours detected in the current frame, comparing the center position with the historical positions stored in "prev_pedestrians". If the distance between the current position and any historical position is below a certain threshold, the pedestrian is considered the same as those detected in previous frames, and their position information is updated. If the current

position does not match any historical positions, a new pedestrian object is created.

3 Results

The pedestrian tracking program was evaluated on four videos. Among them, 1.mp4, 2.mp4, and 3.mp4 were captured by me with mobile phone, and demo.avi[1] was found online. The following results were obtained:

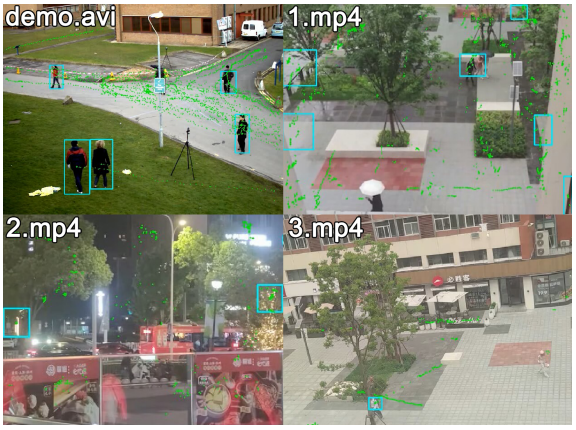

Figure 1: Result

In demo.avi, the program can accurately recognize and achieve target tracking by leaving traces of pedestrians based on their centroids. Overall, the stability is high. Only brief target losses occur when pedestrians are too far away or obstructed by street lights.

However, in the videos I captured myself, the program's effectiveness was not satisfactory due to significant interference. Filmed with a mobile phone, these videos exhibited noticeable camera shake and poor color rendering, adding significant pressure when it came to background removal. Additionally, the backgrounds were not static. In 1.mp4, the rainy weather and accompanying wind caused noticeable swaying of leaves and plants in the footage, making them easily mistaken for pedestrians. Furthermore, various other disturbances, such as reflections on water and glass, various obstructions such as fences and umbrellas that cause incomplete pedestrian images, and flying birds, were present. Moreover, there were additional interferences. 3.mp4 was shot through glass which added a level of greyness. 2.mp4 was filmed at night, making it difficult to distinguish pedestrians wearing dark clothing, and the video itself contained considerable image noise.

In summary, the program can track pedestrians to some extent, but it is prone to misjudgments and losing targets in the presence of significant interference in the video.

[1] `https://github.com/gkm0120/Notes/blob/master/OpenCV/demo.avi`

4 Evaluation

Strengths:

- Background Subtraction: The implemented background subtraction method effectively extracts foreground objects, providing reliable input data for pedestrian tracking.
- Centroid Tracking: Tracking pedestrians based on centroid movement allows for estimating pedestrian trajectories.
- Motion Analysis: Incorporating motion analysis by comparing current pedestrian positions with historical ones enhances tracking accuracy and reduces false identifications.

Weaknesses:

- Sensitivity to Interference: The program struggles to maintain accuracy in the presence of significant interference such as camera shake, dynamic backgrounds, reflections, and incomplete pedestrian images. This sensitivity leads to misjudgments and occasional target losses.
- Challenges with Night Footage: Tracking performance is compromised in low-light conditions, particularly when pedestrians wear dark clothing, causing difficulties in distinguishing them from the background.
- Limited Adaptability: The program's performance varies significantly depending on the characteristics of the input video footage. While it performs well in controlled video like the demo.avi, its effectiveness diminishes in real-world scenarios with diverse and challenging conditions.

As a result, Background Subtraction fails to effectively handle dynamic backgrounds, leading to numerous false detections in noisy environments. Initially, I intended to assign a unique identifier to each pedestrian and represent their motion trajectories with different colors, in order to predict their next movements based on these trajectories. However, due to inaccuracies, target loss, and instances of repeatedly detecting the same individual, even if successfully implemented, the results were below expectations.

In conclusion, the implemented program can achieve the basic goal of pedestrian tracking and trajectory; however, it faces significant challenges in handling real-world interference, particularly in low-light conditions and dynamic backgrounds. To enhance its performance in varying scenarios and reduce misjudgments, it is essential to explore additional methods, such as experimenting with deep learning and YOLO models.