University of Nottingham
UK | CHINA | MALAYSIA

Computer Vision (COMP3065 UNNC) Person Tracking Implementation Base on YOLOv5 and DeepSORT

Nan DONG
20319111

May 4, 2024

School of Computer Science
University of Nottingham Ningbo China

Chapter 1

Project Description

1.1 Objective Description

The objective of this project is to implement multi-target detection and tracking of pedestrians in videos. The goal is to handle complex scenarios with as much accuracy as possible, thereby enhancing the accuracy of pedestrian tracking tasks and reducing misidentifications in situations where multiple targets are present in the videos. Additionally, the project aims to ensure robustness in most pedestrian detection tasks, especially in varying lighting conditions and angles.

1.2 Functionalities Description

This section is designed to introduce the specific architecture of the project. In this project, version 7.0 of YOLOv5 is initially employed to detect individuals in each frame of the video. DeepSORT utilizes this information to track the movement trajectories of each individual, maintaining continuous tracking and displaying their trajectories within the video. By integrating these two algorithms, accurate detection and tracking of multiple individuals in videos can be achieved. According to tests conducted on different videos, the project is able to accurately perform multi-pedestrian detection and tracking, as well as trajectory display

tasks [1].

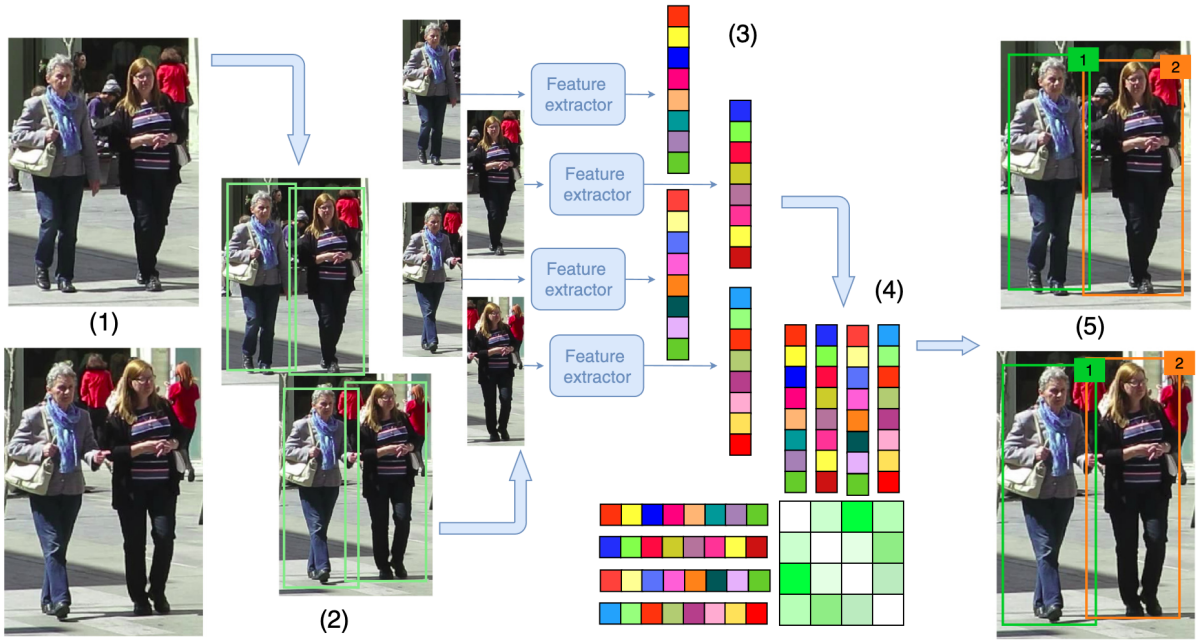

Figure 1.1: Four Steps in MULTI-OBJECT TRACKING

1.3 Structure Description

The main code implementation of the project is divided into three parts. To run the code, this project requires the configuration of environments such as CUDA, PyTorch, and OpenCV. Next, I will describe the specific functionality of each part.

In 'main.py', the primary responsibility is to manage the control flow of the entire system. After setting basic parameters such as the font, the video is opened and video frames are read frame by frame. For each frame of the video, the 'detect' method of the 'Detector' class is called to perform object detection. If targets are detected, the 'update' method of the 'tracker.py' module is invoked to track these targets.

In 'Detector.py', a class named 'Detector' is defined to perform object detection. The primary functions of this class include the 'init' method, which sets up initial configurations such as image dimensions, thresholds, and strides, and initializes the YOLOv5 model weights.

For images inputted in the main function, the ‘preprocess‘ method is used to perform various preprocessing tasks, such as resizing and format conversion. The crucial ‘detect‘ method is responsible for executing object detection and returns information about the detected bounding boxes of the targets.

In ‘tracker.py‘, two functions, ‘draw_bboxes‘ and ‘update‘, are defined for drawing bounding boxes on images and updating target tracking information, respectively. The ‘draw_bboxes‘ function renders the bounding boxes and text labels, and draws a green dot at the center of the target. The function first calculates the thickness of the lines, then iterates through the list of bounding boxes to draw rectangular bounding boxes, text labels, and points on the image. It returns the image with the drawn elements. The ‘update‘ function is used to update the target tracking information. It receives a list containing bounding box information, converts it into the format required by DeepSort, and then invokes DeepSort’s ‘update‘ method to track the targets. Finally, it converts the updated tracking results back into a list of bounding boxes ready for drawing, and returns this list.

Chapter 2

Implementation

2.1 Methodology

This section will introduce the basic principles of some core methods used in the project, as well as their simple workflows. These parts play a crucial role in the project.

2.1.1 YOLOv5

YOLOv5 is an efficient object detection algorithm known for its high accuracy and fast detection speed, making it suitable for various computer vision tasks. It is a detector trained using deep learning methods, with CSPDarknet53 serving as its base network [2]. Additional convolutional layers are added to the top of the network to further extract image features, which are then used to predict the positions and categories of objects. The basic idea of the algorithm is to divide the image into a fixed-size grid and then predict bounding boxes and the probability of each bounding box corresponding to a target class for each grid cell [3].

The algorithm's workflow begins with processing the input image, followed by feature extraction using the CSPDarknet53 model. The extracted features are then passed to the detection head for predicting the bounding boxes, classes, and confidence scores of the targets. Next, multi-class classification is performed using the softmax function. Finally, non-

maximum suppression is applied, and the results are output [4] . This workflow and model architecture contribute to the YOLOv5 model's superior performance in terms of speed, accuracy, and overall efficiency [5].

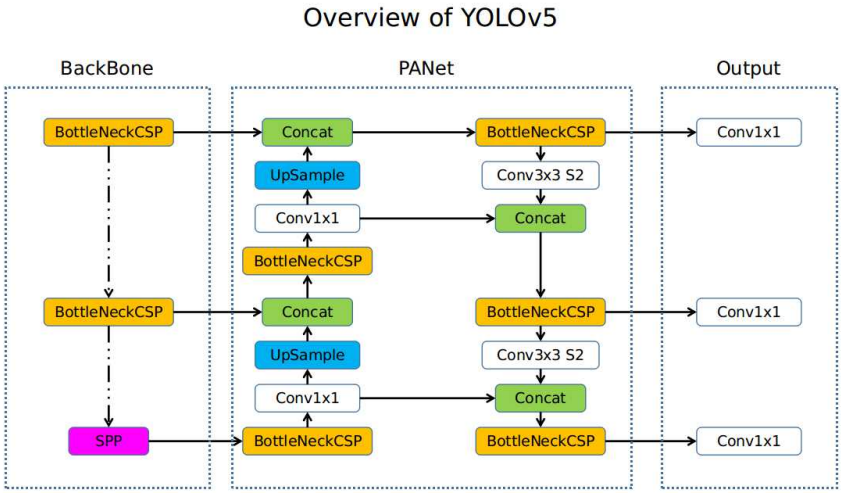

Figure 2.1: Overview of YOLOv5

2.1.2 DEEPSort

The project implements the deepSORT algorithm for pedestrian tracking, which combines deep learning with the traditional object tracking SORT algorithm. SORT is a simple online real-time object tracking algorithm that estimates the future position of a target using a Kalman filter with the current observed target position and velocity information, along with the previous motion model[6]. It associates detection results in the current frame with tracked targets from the previous frame and then uses the Hungarian algorithm to match detection results with tracked targets. These algorithms are briefly described as follows.

- Kalman Filter Algorithm: The algorithm is divided into two processes, prediction and update. It defines the motion state of the target as an 8-dimensional vector of normal distributions. Prediction: When the target moves, the algorithm predicts the current frame's target box position and velocity parameters based on the target box and

velocity parameters from the previous frame. Update: The algorithm linearly combines the predicted state and the observed state, both represented as normal distributions, to obtain the current system's predicted state [7].

- Hungarian Algorithm: The Hungarian algorithm is used to associate the detection results in the current frame with the tracked targets from the previous frame, thereby achieving multi-object tracking. It solves an assignment problem by processing similarity matrices between consecutive frames [6].

DeepSORT enhances SORT by introducing deep learning methods, using deep learning networks (such as CNNs) to extract target appearance features. This allows for more accurate differentiation between different targets, reducing the probability of target confusion and improving tracking accuracy. Deep learning networks can learn more complex target feature representations, are more adaptable to different targets, have better generalization capabilities, and are more robust to occlusion and appearance changes [1].

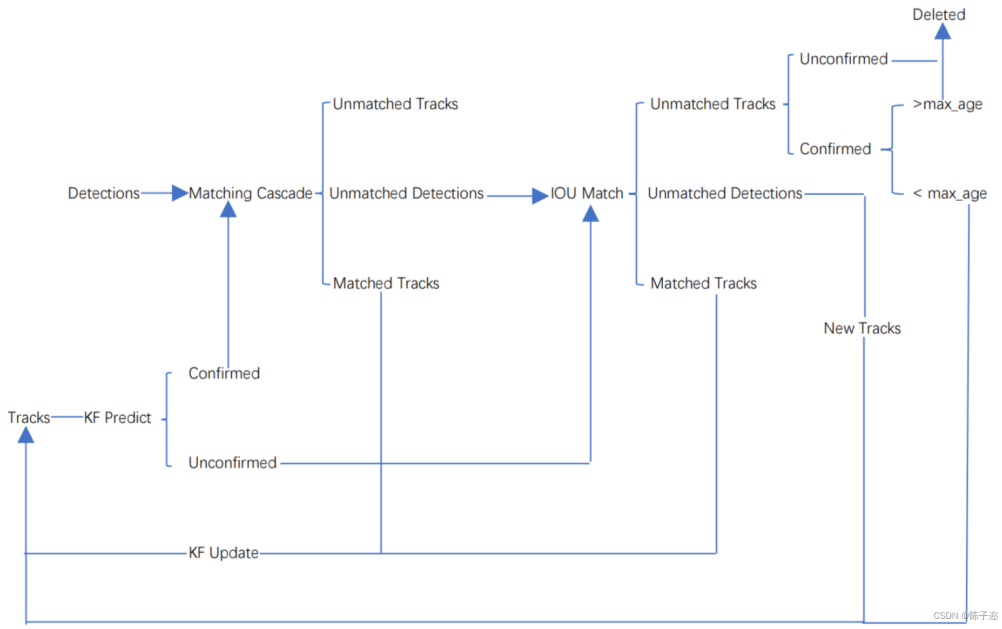

Figure 2.2: DeepSORT Algorithm Working Flow

2.1.3 Market1501

The Market-1501 dataset was collected at Tsinghua University, with data captured by six cameras. These cameras cover various indoor and outdoor scenes. Images captured by each camera have different perspectives and lighting conditions. Each pedestrian in the dataset is assigned a unique ID, and under each ID, there are multiple images showing the appearance changes of the same pedestrian in different scenes and at different times. The dataset includes a total of 1501 pedestrians, with an average of 21.75 images per person. Therefore, this dataset is often used for pedestrian re-identification purposes [8].

2.2 Implementation Process of the project

At the outset of this project, I began by learning the fundamental concepts of object detection and object tracking based on existing knowledge. Object detection refers to the process of locating and classifying specific objects (such as humans or vehicles) in images or videos, while object tracking involves tracing the movement trajectories of these objects across consecutive frames. These two aspects form the core of this project, and their accuracy and robustness determine the project's success. This chapter will introduce the specific implementation processes of these two aspects, including their motivations, model training results, and other related details.

2.2.1 Pedestrian Detection

Based on the knowledge acquired in the course, my initial approach involved extracting pedestrian features from images using the Histogram of Oriented Gradients (HOG) algorithm, with the extracted HOG features serving as inputs. Subsequently, I aimed to utilize a Support Vector Machine (SVM) classifier to differentiate between pedestrians and non-pedestrians. Following the implementation, I conducted a series of optimizations on the code, including image preprocessing and tuning of HOG parameters, among others. The

final detection results are as follows.

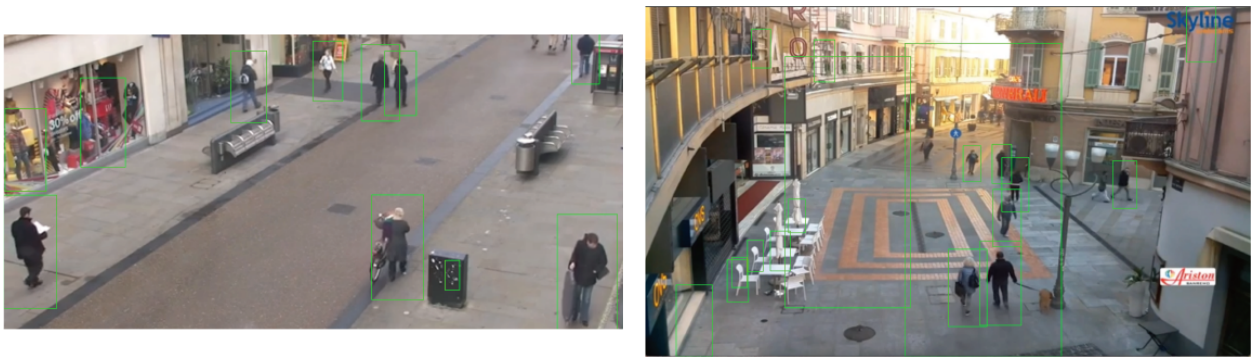

Figure 2.3: Screen Capture From Test Video1

In practical applications, it was found that this method did not achieve high accuracy in complex scenarios. The main issues were errors and missed detections, and the method incurred significant computational costs, making real-time detection in videos challenging. Analysis revealed that the HOG detector is sensitive to factors such as lighting, occlusion, and pose variation, which may lead to reduced detection accuracy. Additionally, this detection method often resulted in overlapping detection boxes, causing multiple detection results to overlap.

As I delved deeper into the code modifications, I became more aware of the limitations of the HOG approach. While this method could detect pedestrians in two simple campus videos I recorded, it proved inadequate for more complex tasks due to its low detection accuracy. Therefore, I decided to employ deep learning methods and came across the YOLOv5 detector. The reason for choosing the YOLO algorithm is that it is a popular object detection algorithm known for its speed and high accuracy. It can achieve fast and accurate object detection using a single neural network model, making it advantageous for real-time monitoring scenarios such as video surveillance.

To ensure the algorithm could run, I spent a lot of time setting up the necessary environment, including PyTorch, CUDA, and other components. After successfully running YOLOv5, I used the model trained by the official team on the COCO dataset. COCO (Com-

mon Objects in Context) is a large image dataset specifically designed for tasks like object detection, segmentation, and pose estimation. The dataset contains over 1.5 million annotated instances, with a total size of 30GB [9]. Therefore, compared to the HOG method, YOLO detectors typically achieve higher accuracy in object detection tasks, especially in detecting people with different poses and when they are partially occluded by objects. The image below shows the pedestrian recognition results using YOLOv5, clearly demonstrating the improvement in accuracy.

Figure 2.4: Screen Capture From Test Video1

2.2.2 Pedestrian Tracking

After configuring YOLOv5 for the project, I proceeded to configure the DEEPSort algorithm section. This component is designed to enable continuous tracking of detected pedestrians. To enable DEEPSort to run, I first needed to train its feature model. I used the Market1501 dataset for this purpose. I chose this dataset because the scenes are similar to the school environment, which can better adapt to the final recorded video. Additionally, this dataset is widely used in the fields of pedestrian re-identification and pedestrian detection, providing ample literature and research results for reference and inspiration.

During the training process, I first configure relevant parameters and devices, such as the dataset directory, whether to use CUDA, GPU ID, learning rate, and logging intervals. After completing this configuration, I load training and testing data and initialize the model.

I set the loss function to cross-entropy loss and the optimizer to stochastic gradient descent (SGD) with momentum and weight decay. I then define a 'train' function to handle each training epoch and a 'val' function to handle each validation epoch. I also use matplotlib to plot the training and validation losses and error rates to monitor the training process and performance. These processes lasted for 60 epochs, with the following performance.

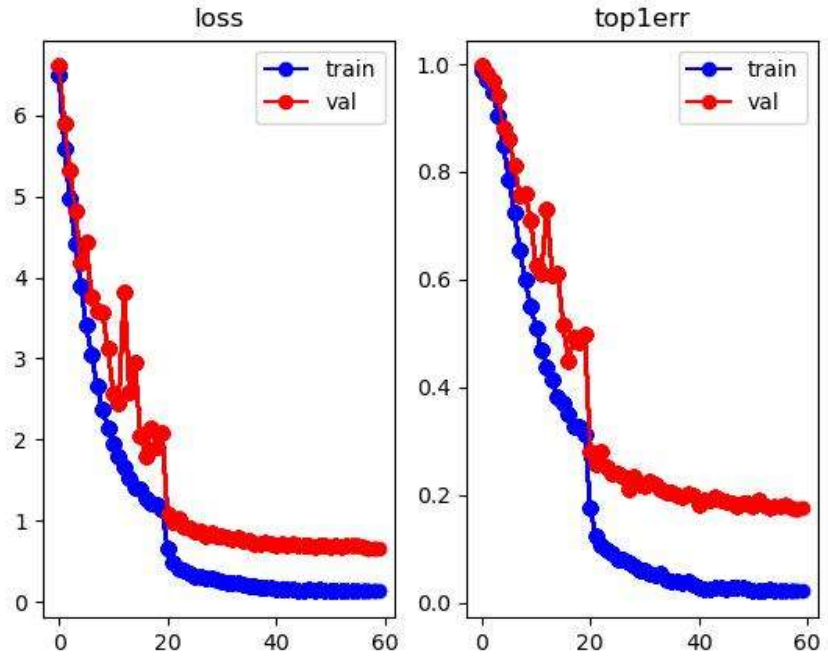

Figure 2.5: Training Performance

After 60 training epochs, the model achieved an accuracy of over 83%. I then incorporated this model into the DEEPSort algorithm and developed a series of pedestrian tracking functionalities based on this algorithm, including textual information and tracking of pedestrian trajectories. Below are some frames extracted from the test video to demonstrate the tracking results.

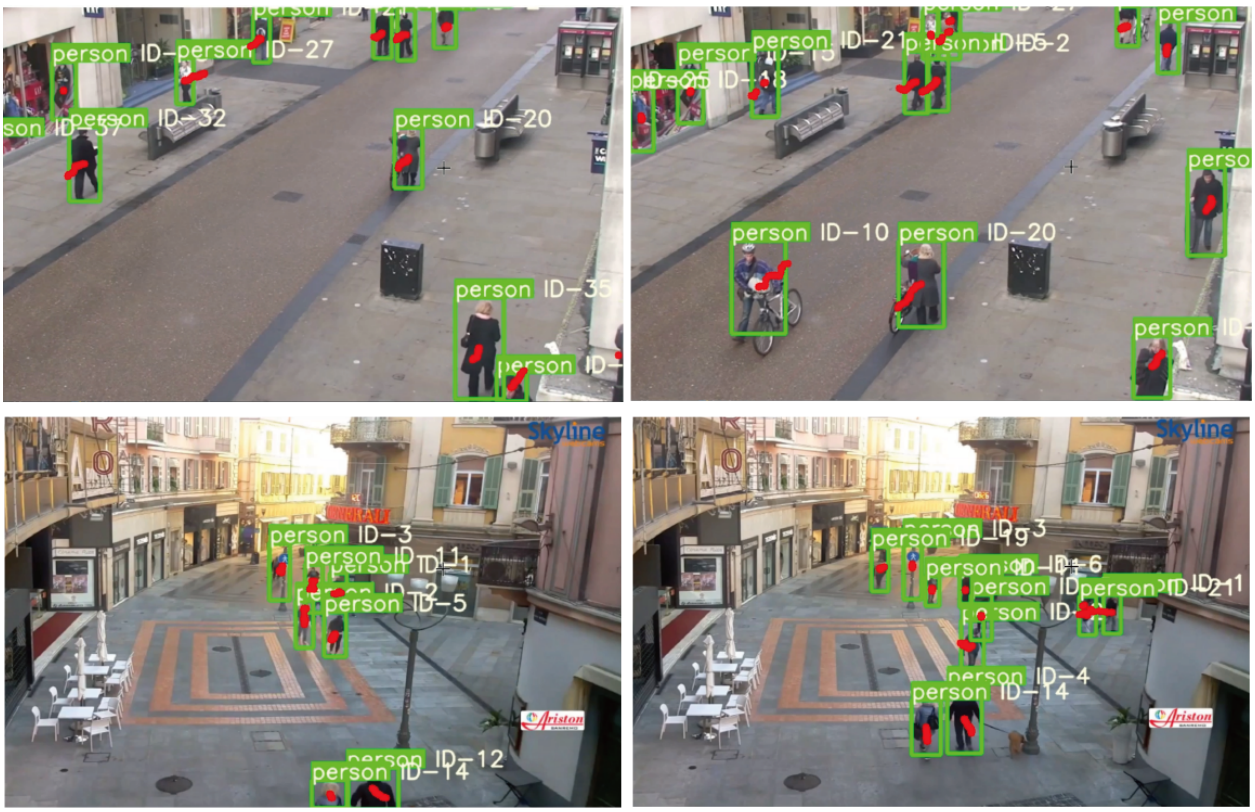

Figure 2.6: Training Performance

Through observations of the pedestrian tracking performance in two test videos, it was found that, except for mannequins inside the display window that were not correctly detected, or the occasional disappearance of pedestrian detection boxes due to people entering shadows or being obscured, the detection and tracking performance was good for the rest of the time in these two test videos. Therefore, I used this model to detect objects in a target video recorded on campus.

Chapter 3

Outcome and Reflection

3.1 Outcome Presentation

To validate the results of this project, I used pedestrian videos recorded on a smartphone within the school premises. To demonstrate the accuracy and robustness of the algorithm, I selected videos under various lighting conditions, different angles, both indoors and outdoors, and in different weather conditions. The tests indicate that the algorithm performs well overall and can handle the aforementioned scenarios. However, the algorithm fails to successfully detect pedestrians when they are occluded, leading to missing detection boxes and trajectories in some frames. The screenshots below illustrate the prject's performance in these videos.


person ID-3
person ID-2
person ID-1
person ID-9
person ID-2
person ID-1
2004
2024
20th
Anniversary
person ID-1
person
person ID-10
person ID-11
person ID-3
person ID-2
person ID-2

3.2 Project Reflection

This project utilized YOLOv5 in conjunction with the DEEPsort algorithm to implement video pedestrian tracking, a common approach for accurately achieving this functionality. However, due to a lack of knowledge in computer vision and deep learning, much time was spent exploring and selecting the appropriate methods. Additionally, unfamiliarity with the setup required for deep learning resulted in a significant expenditure of time in configuring the environment. Through this assignment, I gained a deeper understanding of the field, comprehensively grasping how similar projects and engineering tasks are progressively realized, from model construction to training perspectives. This understanding represents the greatest takeaway from this project. The following section provides a comprehensive analysis of the project's strengths and limitations based on its performance across two test videos and four target videos.

3.2.1 Project Advantages

In the total of six videos, the algorithm performs excellently, specifically in detecting pedestrians appearing in the videos. It has a high tolerance for variations in lighting, angles, and weather conditions, and can handle complex human figures. In most cases, it effectively tracks pedestrians, maintaining stable tracking even when pedestrians are obscured or intersect with each other. These strengths demonstrate that the project successfully meets its expected goals, mainly due to the following reasons.

- YOLOv5 offers rapid and accurate object detection capabilities, particularly adept at recognizing multiple objects quickly in complex environments. It is also optimized for real-time applications, capable of processing video streams with minimal latency [4].
- DeepSORT effectively tracks pedestrians by using appearance features, maintaining stable tracking even when pedestrians are obscured or intersect with each other. The

Hungarian algorithm ensures the optimal matching between tracked targets and detected targets.

- Combining both methods enables effective handling of common issues such as occlusion and lighting changes, aiding in re-identification and continued tracking after a target briefly disappears [10].

3.2.2 Project Limitations

Based on the result videos, the project has the following issues:

- Misidentification: Mannequins in the display window are incorrectly identified as people, and pedestrians walking dogs are initially detected as people.
- Failure to detect: Pedestrians are not correctly detected when heavily occluded. Pedestrians wearing dark clothes in shadowed areas are not correctly detected in some images.
- Tracking stability: In extremely crowded scenes, the tracking algorithm may encounter more challenges, such as frequent occlusions and target intersections, which may result in frequent tracking ID switches or loss.

In the future, these project shortcomings can be addressed through several methods. For example, upgrading the YOLO model to the latest version to observe if there are still instances of misidentification and failure to detect. Introducing post-processing methods such as shape- or motion-based filtering to help eliminate misidentifications can also be beneficial. Additionally, training the model using larger-scale datasets or other datasets that include more diverse scenes and pose variations can improve performance. Lastly, optimizing the parameters of the Hungarian algorithm for video mayby can enhance its accuracy.

Bibliography

[1] Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88, 2020.

[2] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. A review of yolo algorithm developments. *Procedia computer science*, 199:1066–1073, 2022.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[4] Do Thuan. Evolution of yolo algorithm and yolov5: The state-of-the-art object detention algorithm. 2021.

[5] Juan Du. Understanding of object detection based on cnn family and yolo. In *Journal of Physics: Conference Series*, volume 1004, page 012029. IOP Publishing, 2018.

[6] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[7] Youngjoo Kim, Hyochoong Bang, et al. Introduction to kalman filter and its applications. *Introduction and Implementations of the Kalman Filter*, 1:1–16, 2018.

[8] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015.

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[10] Yuqiao Gai, Weiyang He, and Zilong Zhou. Pedestrian target tracking based on deepsort with yolov5. In *2021 2nd International Conference on Computer Engineering and Intelligent Control (ICCEIC)*, pages 1–5. IEEE, 2021.