Detecting and Tracking pedestrians with YOLOv5 and DeepSort

Rui Liu
z5328240

Leyang Li
z5285799

Yueyang Gu
z5251261

Siyuan li
z5378016

Jiaxi Cheng
z5405222

***Abstract*— Tracking has wide applications in various real-world problems, including human-computer interaction, autonomous vehicles, robotics, crowd surveillance, traffic monitoring, ocean and space exploration, precision surgery, and biochemistry. With this wide range of practical applications, object tracking has naturally become an active research field in computer vision. Meanwhile, object tracking still encounters numerous difficulties because of the large volume of data, background clutter, motion blur, and illumination variations. It is quite challenging for humans to perform accurate, robust, complete, efficient, and reproducible recognition and analysis of the relevant information in realistic scenarios. One of the problems addressed by computer vision is multi-object tracking in real-time videos or time-lapse image sequences. This project develops and evaluates a pedestrian detection, tracking, and motion trajectory analysis system in real-world video recordings. Since Many traditional and/or machine or deep learning-based computer vision methods could be used for this, we review the latest trends and advances in the tracking area, and broadly categorize, introduce and compare some of the most popular methods in the first part of this work. In the second part of this work, there is a detailed description of our model based on You Only Look Once (YOLOv5) and DeepSort. The YOLOv5 network was pre-trained and tested on an external dataset which resulted in an enhanced tracking precision accuracy**.

Keywords—tracking, multi-object, pedestrian, YOLOv5, deep sort

I. INTRODUCTION

A. Background

Computer vision refers to the realization of human's visual function -- perception, recognition, and understanding of three-dimensional scenes in the objective world with computers, which means that the research goal of computer vision technology is to make computers have the ability to recognize three-dimensional environmental information through two-dimensional images. Therefore, it is not only necessary to make the machine perceive the geometric information (shape, position, posture, motion, etc.) of objects in the three-dimensional environment, but also to describe, store, recognize and understand them. It can be argued that computer vision is different from the study of human or animal vision: it relies on geometry, physics, and learning techniques to build models, so as to use statistical methods to process data. There are mainly two types of methods: one is the bionics method, that is, according to the structural principles of the human visual system, the corresponding processing module is established to complete similar functions and work; The other is the engineering method, which starts from the analysis of the function of the human visual process, does not deliberately simulate the internal structure of the human visual system, but only considers the input and output of the system, and uses any existing feasible means to realize the system function. The second kind of method is the current trend of computer vision technology research, which has developed a set of independent computational theories and algorithms.

Object detection is a research hotspot in computer vision, which has application requirements in many fields, such as monitoring security, automatic driving, traffic monitoring, and robot vision scenarios [1]. Object detection generally detects some predefined categories of object instances (such as people and cars, etc.).

Several related fields such as Intelligent monitoring, human-computer interaction, and virtual reality utilize multi-target tracking as a core component of computer vision. The multi-target tracking method works by expressing the target as efficiently as possible, and it locates the most similar candidate target area within the image sequence. Because multiple targets can be blocked or eclipsed, tracking multiple targets is more challenging. Multi-target tracking has become a popular topic in computer vision over the past few years as many researchers have carried out different researches on it. Target-specific coordinate motion trajectories were traditionally calculated solely from pixels in multi-target tracking. This, however, affects the recognition of abnormal behavior, like trailing and wandering, by causing deviations in the tracking result.

B. Motivation and contribution

Pedestrian tracking is one of the most classic cases of multi-object tracking problems in traffic environments.

Challenges may arise in situations where objects enter or leave the scene, have a similar appearance, and change appearance over time due to illumination changes, scale and shape changes, deformations, and partial occlusions, making it hard to keep track of their unique identity. Therefore, it is still a highly active research area in computer vision.

The main contribution of this work is the implementation, systematic analysis, and experimental evaluation of a well-performed tracking model on a typical pedestrian tracking application in daily scene. The Moderate complexity of both the scenarios and methods prepares us for the tougher task in the future study of computing vision.

C. Tasks

The aim of this project is to develop and evaluate an interactive pedestrian tracking program.

There are three main steps of our tasks: track pedestrians, count pedestrians, and analyze pedestrians. Among them, the first task is the most basic and challenging one. It involves two

main sub-tasks, the detection of objects and the tracking of objects between the different frames.

D. Dataset

The given dataset is composed of a total of four pedestrian videos during various times of the day and in varying background lighting conditions. RGB cameras were deployed at varying heights and in varying lighting conditions. The videos were recorded with a resolution of 1920×1280 pixels (frame width×height) at 30 frames per second. The length of each one of the videos is within 20 seconds.

The dataset provides annotations where every pixel has a semantic label and all pixels belonging to the most salient object class (pedestrian) have a unique tracking ID.

As for the pretrained YOLOv5, it is trained on the Coco dataset. Coco dataset is a large-scale dataset that can be used for image detection, semantic segmentation, and image captioning. It has more than 330k images (220K of which are labeled images), including 1.5 million targets, 80 target categories (object categories: pedestrian, car, elephant, etc.), 91 material categories (stuff categories: grass, wall, sky, etc.), each image contains five sentence descriptions of images, and there are 250000 pedestrians with key points [2].

E. Challenges

In addition to the difficulties of general human body detection, such as clothing change and posture change, pedestrian detection also has the following difficulties due to its specific application field: the camera is moving, so the method widely used in the field of intelligent monitoring to detect dynamic targets cannot be directly used; Pedestrian detection is faced with an open environment, considering different road conditions, weather and light changes, which puts forward high requirements for the robustness of the algorithm. Real-time performance is the requirement that the system must meet, which requires the image processing algorithm should not be too complex.

As for the specific dataset, through observation, we could find that the frames of each video have the following characteristics:

Good characteristics that are conducive to the choice of appropriate traditional methods.

- The video background remains unchanged
- The angle of the camera is fixed
- The video is noticeably short, thus illumination remains unchanged, and so does the illuminant direction.
- Only pedestrians move, and they move in a slow pace.

Bad characteristics/challenges prompted us to choose a more modern approach, which is deep learning.

- The contour profile of pedestrians is not obvious
- The light is dim, and the picture is blurred
- There is wind (the trees leaves sway)
- Sometimes have very crowded scenes
- Background clutter: streets have more distractions and clutter, such as retail stores, trees, windows etc., which will lead to false positives

II. Literature Review

A. Detection

Object detection is one of the most crucial tasks in computer vision and remote sensing to identify specific categories of various objects in images [3]. Target detection supports many visual tasks, such as instance segmentation, pose estimation, tracking, and motion recognition.

Almost all early target detection algorithms are based on manual features [4].

As the performance of manual characteristics tends to be saturated, numerous research has emerged using and proposing novel architectures and methods in computer vision such as R-CNN (Region-Convolutional Neural Network), Fast R-CNN, and YOLO (You Only Look Once) [5].

Deep learning is now widely used in image processing and object detection due to its powerful feature learning capabilities.

Broadly speaking, CNN-based object detection methods include two-stage detectors (e.g., Regions with CNN features (RCNN), Spatial Pyramid Pooling Networks, Fast RCNN, Faster RCNN, and Feature Pyramid Networks (FPN).) and one-stage detectors (e.g., You Only Look Once (YOLO), Single Shot Multi-Box Detector (SSD) and Retina-Net [3].

Among them, the YOLO models with a one-stage detector have advantages in real-time optical image detection but have relatively low accuracy for small objects [3].

YOLO is an advanced single-stage object detection framework, which has undergone the evolution from V1 to V5 [6], which combines traditional compressed sensing, and YOLOX [7]. As shown in Figure 1, the development history of the YOLO detection model is briefly shown [8]. Its greatest advantage is its speed - YOLO processes 45 frames per second and is incredibly fast. In addition, YOLO is capable of representing generalized objects. The performance of this algorithm is comparable to that of R-CNNs [9].

YOLO requires fewer resources than other CNN methods, such us, Faster R-CNN, Fast R-CNN [10], and the end-to-end network structure provides a detection rate that is higher than that of other networks.

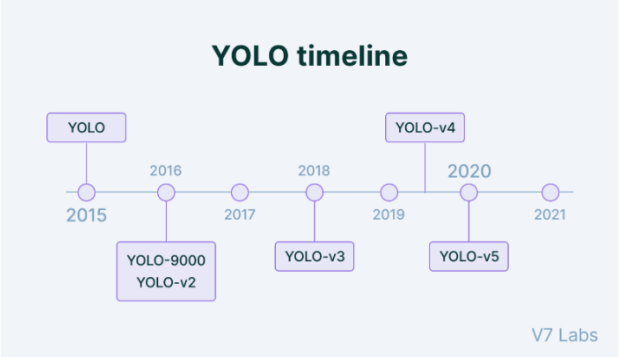

Fig. 1. The development of YOLO detection model [8]

The YOLO algorithm divides the pictures that need to be detected into SS grids, each of which has a unique detection task. Two complete connection layers and 24 convolution layers make up the entire network structure. The output of the entire connection layer is the tensor S S (B 5 + C), where B

stands for the number of anticipated targets in each grid and C stands for the number of categories. Regressing the detection box location and assessing the category likelihood of the tensor data will yield the final detection result. The training images in YOLO v5 now include the Mosaic data augmentation approach. Four distinct images are combined into one image by random scaling, random cutting, and random arrangement. Through these methods, training picture background information is enhanced, which is particularly helpful for tiny target detection. Each batch of training data is transmitted by the YOLO-v5 algorithm through the data loader while being improved. Additionally, the anchor mechanism of Faster R-CNN is used to improve the YOLO-v5 algorithm's capacity for small target recognition in the picture by way of a multi-scale mechanism during the image detection process. Additionally, it gives the YOLO-v5 algorithm considerable adaptability to various image sizes. Due to the following network structure advancements, YOLOv5 is more effective and capable of inferring. At the neck section, SPP is swapped out for SPPF compared to YOLOv3 and v4. By serializing the top pooling layer, SPPF can minimize memory requirements while still achieving the same level of pooling as SPP's first three pooling layers. This increases operational efficiency. The first layer, which was present in versions prior to YOLOv5 v5.0, was changed to a 6x6 convolutional layer. Despite having the identical outcome, in theory, the convolutional layer's actual impact is superior to the former's due to GPU optimization and the pertinent method. To increase the network's stability, CSP is also included and is based on the PAN structure of V4. Additional changes have been made to specifics such as choosing the activation function of the convolution module.

B. DeepSort

DeepSort introduces an apparent feature extraction network based on SORT(Simple Online Realtime Tracking) target tracking. The SORT is a straightforward and efficient multiple objects tracking algorithm that tracks an object using both the Hungarian algorithm and the Kalman filter. The four major parts of SORT are detection, estimation, data association, and creation and deletion of track identities. However, occlusions and various camera angles will reduce the SORT algorithm's effectiveness. To improve the SORT algorithm, a more effective association measure is used by DeepSORT, which integrates motion and appearance descriptors. The network is trained offline on the pedestrian recognition data set. In the process of target tracking, the apparent features of the target are extracted for nearest neighbor matching, which can effectively improve the target tracking effect under occlusion. The DeepSORT algorithm has main components as follows[17]:

- Track estimator, the original component of the SORT algorithm, forecasts the locations of the object bounding boxes using the Kalman filter. Based on earlier object velocities, a prediction has been made. The track estimator uses intersection over union (IoU) distance between the predicted and detected bounding boxes as its location metric.
- Appearance descriptor, using CNN, the appearance data is retrieved so that the features from the same object class are comparable and the features from various object classes differ in the feature space. The appearance metric uses the appearance descriptor.
- Data association uses location and appearance metrics to relate the observed bounding boxes to an object's actual track. Each active track serves as an item identifier.
- Track handling, a freshly detected bounding box will be placed into a tentative track if it cannot be connected to any track at frame f. In the next frames, the DeepSORT will attempt to connect the tentative track with other tracks. The track is updated if the association is a success. If not, the experimental track will be dropped.

C. Object tracking

Literature shows that much research has been performed on object tracking, and various surveys have been published. Arnold et al. [11] provide objective insight into the strengths and weaknesses of trackers. Li et al. [12] make a comprehensive review and exhaustive evaluation of existing deep visual trackers on large-scale benchmarks. Fiaz et al. [13] review the latest trends and advances in the tracking area and evaluate the robustness of different trackers based on the feature extraction methods.

Many research using YOLO series and DeepSort have been proposed in the literature, in conjunction with many application areas. For instance, Ref. [3] provides an experimental evaluation of unmanned aerial vehicle (UAV)-based thermal infrared (TIR) remote sensing multi-scenario images and videos. Whose tracking methods are designed for moving cameras and high viewpoint. Recently, a survey [14] focused on analyzing the physical distance between pedestrians in real-time, shows that YOLO using pre-trained CNN performs well in validated on video with simple scenes. Prepare Your Paper Before Styling

III. METHODS

A. Task1: Track Pedestrians

1) Task 1.1 Detect and track all pedestrian:

After data pre-processing, train processing with the pre-trained model[17] configurations involved 15 models based on the COCO dataset, the pre-defined network[19], and the number of key parameters used. The software and hardware platforms adopted in the training experiment were implemented on a computer with a 2.5 GHz CPU, 8 GB RAM, GeForce GTX 1080ti GPU, CUDA10.2, and cuDNN7.0. YOLOv3, YOLOv4, and YOLOv5 were implemented by Python and PyTorch. The key parameters of Wang et al.'s (2020) and Ultralytics's (2020a, 2020b) programs (available on GitHub) were used in training the models.

The YOLOv5 framework of pedestrian detection is based on logical regression using convolutional neural networks. With Deep-SORT and YOLOv5, this step mainly consists of three multiscale modules:

1. Feature extraction to detect pedestrians.
2. Count pedestrians.
3. Track the motion path.

By inputting the frames of the video into the frame network structure, YOLOv5 creates layers and extracts features, such as the boundary of the pedestrians and the centroid of the figure, and then performs feature mapping on the output object. The detector module uses YOLOv5 architecture to layer the deep neural network and generate the detection with different scales of kernels, followed by the counting and tracking module.

DeepSort is a DBT(Detection-Based Tracking) algorithm,

The usual workflow of the DBT algorithm is:

1. Each frame of the image is detected to get the bounding box of each target.

2. Calculate different features for each detected object.

3. The similarity calculation calculates the probability that the two objects in the front and back frames belong to the same target.

3. Associate assign each object ID

After detection and tracking using YOLO and DeepSort, we can get the result as shown in figure 2:

```
def tracking(self, image, total_face_bboxes, face_bboxes, centroids, select_roi_points, image_scale, show_group_flag):

    results = {
        'face_bboxes': face_bboxes,
        'centroids': centroids
    }

    image, faces, face_bboxes, centroids = update(self, image, total_face_bboxes, face_bboxes, centroids, select_roi_p

    results['image'] = image  1
    results['faces'] = faces  2
    results['previous_face_bboxes'] = [item for item in results['face_bboxes']]  3
    results['previous_centroids'] = [item for item in results['centroids']] 4
    results['face_bboxes'] = face_bboxes  5
    results['total_face_bboxes'] = total_face_bboxes  6
    results['centroids'] = centroids  7

    return results
```

Fig. 2. The results gained in Task1

1. Pictures after processing (drawing rectangles, trajectories, text annotations)

2. Information of new people entered in this frame

3. Information of people in the previous frame

4. The center of the bboxes in the previous frame

5. Information of people in this frame

6. Information of all characters from the beginning to the present

7. The center of the bboxes in the current frame

2) Task1.2

According to the center of the person in the previous tracking results, use deque to record the history of each person's center point, and use CV to draw the person's center point. After each frame is connected, it becomes a trajectory line, and the size of the point is not fixed. According to the current position of the point in the deque, the farther and thicker the trajectory line drawn is from the body, and the closer and thinner it is from the body.

3) Task1.3

According to the data obtained from 1, draw a box with cv2.rectangle, and assign colors according to different pedestrians' IDS recorded in 1.

B. Task2: Count Pedestrians

1) Task2.1

After Task1 is implemented, we use a list to save 6(Information of all characters from the beginning to the present). According to the length of this list, we can know the total number of people since the beginning of the video.

2) Task2.2

After detection and tracking using YOLO and DeepSort, we save 2(Information of new people entered in this frame) with a list. According to the length of this list, we can know the total number of people since the beginning of the video.

3) Task2.3

We use cv2.selectroi to frame the specified area and press the space to activate the selection. At this time, the video pauses jump out of the picture, and let you frame it. After selecting the area, press the space again, and return to the video. You can find a pink box appearing in the video. Press 'Q' to reset the selected area.

4) Task2.4

According to the coordinates of ROI and the coordinates of each pedestrian recorded in the bboxes, if the range of each coordinate is within ROI, the number of people in ROI +1.

All the number information is directly displayed in the upper left corner of the video with cv2.puttext.

C. Task3: Analyse Pedestrians

1) Task 3.1

We use the corresponding module to get the Euclidean distance between two directly from the centroids of the walkers Artificially set the range. If the distance between pedestrians is less than the threshold, it will be counted as a group. According to the pictures of two consecutive frames, if the distance between pedestrians is less than the threshold and remains more than 1 frame, we will count them as a group and add their character information to a group.

2) Task 3.2

traverse all currently existing groups, get the approximate coordinates of the most marginal position according to the centroid coordinates of the people in the group, and then draw a bold box with cv2.rect to circle a group. If someone leaves this group, the box will also change. The number of people in the group is obtained by adding the value 'len' of the person's information in each group. Because the bold lines block the line of sight very much, a switch is set, and pressing 'e' can hide/display the group information.

3) Task 3.3

Read the picture size with shape, and judge to enter/leave the current scene according to the distance between the centroid of the current personal information in the picture and the picture border. Artificially set, if the distance is less than 80, it is judged to enter/leave the current scene. At this time, use a CV to draw an exclamation point in the person's box to show that he is about to enter/leave the scene.

IV. EXPERIMENTAL RESULT

A. Running environment:

Windows10, cuda V10.2.89, cudnn V8.10, GPU: RTX3060.

B. *Results of Task1*

1) *Evaluation metrics*

Accuracy, precision, recall, and f1-score are used to evaluate the performance of our model. We capture the last 20% of processed images to calculate the metrics.

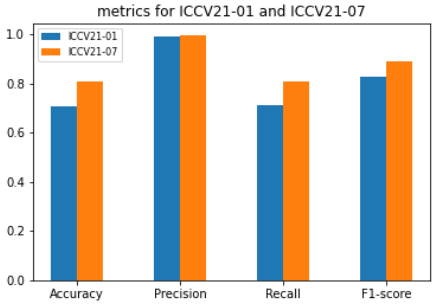

Fig. 3. Metrics

TABLE I. DETAILED METRICS OF TESTING VIDEO

	ICCV21-01	ICCV21-07
TP	972	1742
FP	10	6
FN	395	409
TN	0	0
Accuracy	0.706	0.807
Precision	0.990	0.997
Recall	0.711	0.810
F1-score	0.828	0.89

Our model has high precision, the false negative number is the main problem that reduces the accuracy.

2) *Ablations study*

We modify the candidate threshold and IoU threshold of the non-maximum suppression in YOLO to limit the candidate number and remove duplicated bounding boxes. A higher candidate threshold provides fewer candidates. A Higher IoU threshold remains more bounding boxes. After many attempts, we find that dataset ICCV21-07 is more sensitive to the candidate threshold, and dataset ICCV21-01 is insensitive to the candidate threshold. Finally, we set the candidate threshold and IoU threshold as 0.1 and 0.4.

We also try some different pre-processing to improve the performance. We use torchvision.transforms to implement the transforms. Its purpose is to is to enhance the image data and to improve the generalization ability of the model. After modifying the brightness and contrast of the image, there is no significant improvement. The crop, blur and flip operations even reduce the performance. Considering the real-time performance, the pre-processing should be as less as possible, by comparing the results of with transforms and without transform, we finally decide not to use the transform method.

Fig. 4. with transforms

Fig. 5. without transforms

3) *performance*

Based on the observation, the trajectory of each pedestrian is drawn very precisely at the beginning, but some trajectories are mixed up in the latter frames. All the trajectories are continuous lines, the model tracks every pedestrian accurately.

Fig. 6. till 90th frame

Fig. 7. till 438th frame

C. *Results of Task2*

Counting the number of pedestrians is based on the detection in Task1. As the number of false positive and true

negative is far less than the number of true positive and false negative, the accuracy in Table1 can also be used to evaluate the performance of pedestrian counting. As shown in Figure8, there are actually 13 pedestrians in that frame and 3 pedestrians in the selected region, our program report there are 9 pedestrians in the frame and 3 pedestrians in the selected region. The reasons that cause the missing detection will be covered in the discussion part. You can press 'space' to select a region and press 'space' or 'enter' again to tell the program the region is selected. Then, you can press 'Q' if you want to select another new region.

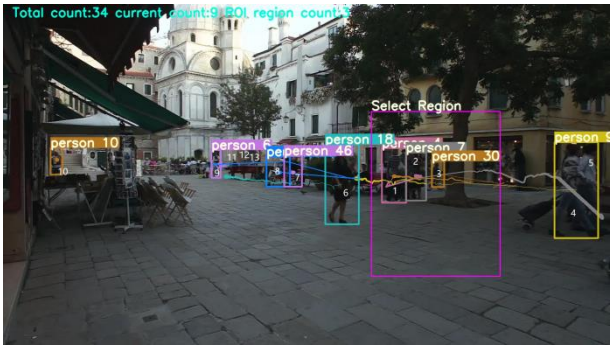

Fig. 8. Case to show counting

D. *Results of Task3*

Based on the Euclidian distances of centroids of pedestrians, we split the pedestrians into different groups. If the Euclidian distance between 2 pedestrians stays less than 80 pixels (test with STEP-ICCV21-01) for over 1 frame, the program put them into one group. We tried 3 methods to avoid duplicated groups.

1. Using list(set(list)) to only keep the unique pedestrian information.

2. Checking whether the pedestrian information is already in one group.

3. Using the NMS in imutils.object_detection.

However, the duplicated groups still appear in some frames. If a group is detected, a rectangle with thick lines will surround the group members. The existence of the rectangles will block the view, you can press 'E' to hide or show the group rectangles on the scene.

Fig. 9. Group detection (good performance)

Fig. 10. Group detection (with duplication)

When a pedestrian enters or leaves the scene, the program will draw a big white exclamation mark on this pedestrian. The distance between the centroid of a pedestrian and the border of the scene is a simple but effective way to detect the entering or leaving.

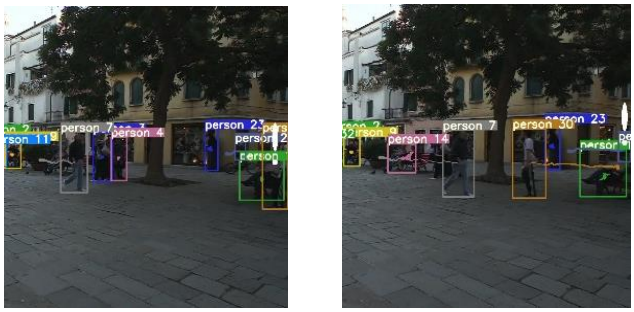

Fig. 11. (a) enter the scene (b)leave the scene

V. Discussion

Before using YOLO and the deep-sort algorithm, we tried two other methods, the detection of pedestrians of those two methods is not good enough. In the first method, the inter-frame difference method [15], we use the background segmentation function in OpenCV to create a mask. After subtracting the background from the original image, the pedestrians and some noise will be remained in the image. Then using dilation and erosion to identify blobs and contours. Each unique contour stands for a pedestrian. While the accuracy and precision of method 1 are both very poor.

In the second attempt, we use the Histogram of Oriented Gradient [16] to extract the features in frames. Then use the linear Support Vector Machine to make the classification. Finally, use the Non-Maximum Suppression to get the best candidates. Although we used NMS, the performance was still not good enough. Compared with method one, method two has higher precision. However, many pedestrians are not detected.

Fig. 12. Attempt 1

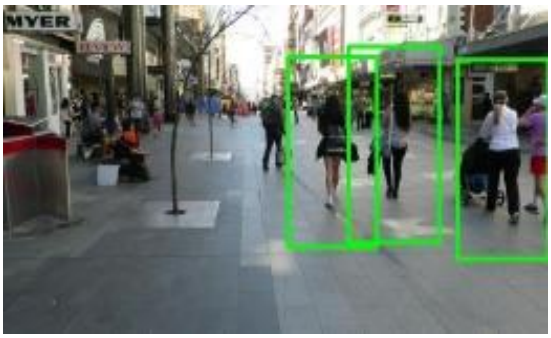

Fig. 13. Attempt 2

Finally, we decide to use the YOLO and deep-sort algorithm. It provides good performance, but there are still some limitations.

We find 6 cases that will affect the accuracy.

1, The surrounding environment. If the environment is very dark or contains too many distractions, the features of the pedestrians cannot be extracted accurately. If a pedestrian's body is occluded by some non-pedestrian objects in the scene over half, the features of that pedestrian may not enough for the model to classify him as a pedestrian.

2, Resolution. If a pedestrian is very small in the scene, in other words, he is far away from the camera, his features may not enough for the model to classify him as a pedestrian.

3, Posture. No matter whether a pedestrian is sitting or running or lying on the ground, he is always a pedestrian for us humans. However, if the training data does not contain enough postures, the model may not recognize the pedestrians who are in a weird pose.

4, Wearing. The wearing will change some features of the pedestrians. For example, if a pedestrian wears a hat, his face will be very dark.

5, Overlapping. If one pedestrian is obscured by another pedestrian, the detection of the pedestrian behind will be affected.

6, Some tricks. There exist some objects whose appearances are very similar to pedestrians, for example, the statue in our test dataset. The model cannot tell the difference between those objects and real pedestrians.

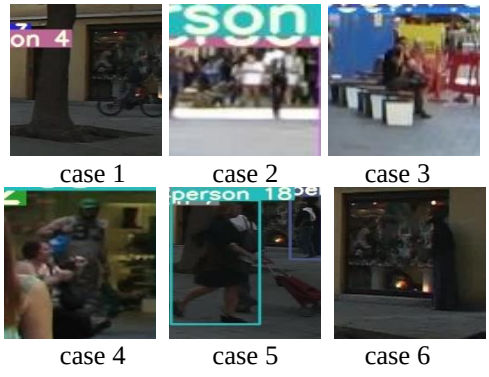

Fig. 14. cases that affect the result

VI. Conclusion

We started with model selection and tested a series of models based on the detection. In addition, we consider the reasoning time and model architecture and choose YOLOv5.

The pedestrian object detection framework based on YOLO and Deep Sort algorithm in this project was proven to be efficient in detecting real-time videos or time-lapse image sequences with multiple objects.

The model still has room for improvement in mapping and delay. When the scene is crowded, there are some missed inspections, resulting in a slightly smaller number of pedestrians than the actual number. The selected super parameters are given by YOLOv5 by default. We can use optional and other super parameter search libraries to optimize them. When there is a difference between the training distribution and the test distribution, domain adaptation is another technology that can be used. Similarly, this situation may require a continuous training cycle, including additional data sets, to ensure the continuous improvement of the model.

Due to the weather, the angle, and the range of future testing data are not fixed, we can implement more pre-processing. For example, the program will automatically check whether a frame is overexposed or underexposed and do some pre-processing to make the frame more balanced. Then increase the brightness and contrast in some special region, like the face under a hat or a pedestrian in the shadow.

Additional data helps to make the model more robust to background interference, but the amount of data collected is still very small compared with the overall data set size, and the model still has some false positives.

For the group detection part, although we use 3 different ways to avoid the duplicated groups, the overlapping still exists. A simple list maybe not be the best container to store the group information.

Disjoint-set data structure will be used to store the group information. This structure is to store a collection of disjoint (non-overlapping) sets. Euclidian distance will still be the basis for judging whether pedestrians are in a group. We will iterate over the pedestrians in a frame and assign a parent node randomly. One parent node stands for one group. When a new frame is passed to our program, the program will do the iteration again, and check the difference between current group information and previous group information.

References

[1] Idrees H, Shah M, Surette R. Enhancing camera surveillance using computer vision: a research note[J]. Policing: An International Journal, 2018, 41(2): 292-307.*)*

[2] https://blog.csdn.net/qq_44554428/article/details/122597358

[3] Jiang C, Ren H, Ye X, et al. Object detection from UAV thermal infrared images and videos using YOLO models[J]. International Journal of Applied Earth Observation and Geoinformation, 2022, 112: 102912.

[4] Soliday S, Perona M T, McCauley D G. Hybrid fuzzy-neural classifier for feature level data fusion in ladar autonomous target recognition[C]//Automatic Target Recognition XI. SPIE, 2001, 4379: 66-77.

[5] Wang Z, Jin L, Wang S, et al. Apple stem/calyx real-time recognition using YOLO-v5 algorithm for fruit automatic loading system[J]. Postharvest Biology and Technology, 2022, 185: 111808.

[6] Ting L, Baijun Z, Yongsheng Z, et al. Ship detection algorithm based on improved YOLO V5[C]//2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE). IEEE, 2021: 483-487.

[7] Ge Z, Liu S, Wang F, et al. Yolox: Exceeding yolo series in 2021[J]. arXiv preprint arXiv:2107.08430, 2021.

[8] https://www.v7labs.com/blog/yolo-object-detection

[9] Gkioxari G, Hariharan B, Girshick R, et al. R-cnns for pose estimation and action detection[J]. arXiv preprint arXiv:1406.5212, 2014.

[10] Jiménez-Bravo D M, Murciego Á L, Mendes A S, et al. Multi-Object Tracking in Traffic Environments: A Systematic Literature Review[J]. Neurocomputing, 2022.

[11] Smeulders A W M, Chu D M, Cucchiara R, et al. Visual tracking: An experimental survey[J]. IEEE transactions on pattern analysis and machine intelligence, 2013, 36(7): 1442-1468.

[12] Li P, Wang D, Wang L, et al. Deep visual tracking: Review and experimental comparison[J]. Pattern Recognition, 2018, 76: 323-338.

[13] Fiaz M, Mahmood A, Javed S, et al. Handcrafted and deep trackers: Recent visual object tracking approaches and trends[J]. ACM Computing Surveys (CSUR), 2019, 52(2): 1-44.

[14] Himeur Y, Al-Maadeed S, Almadeed N, et al. Deep visual social distancing monitoring to combat COVID-19: A comprehensive survey[J]. Sustainable Cities and Society, 2022: 104064.

[15] Cheng Y H, Wang J. A motion image detection method based on the inter-frame difference method[C]//Applied Mechanics and Materials. Trans Tech Publications Ltd, 2014, 490: 1283-1286.

[16] Datal N. Histograms of oriented gradients for human detection[C]//Proc. 2005 International Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2005, 2: 886-893.

[17] Dang, Linh & Nguyen, Gia & Cao, Thang. (2020). Object Tracking Using Improved Deep_Sort_YOLOv3 Architecture. ICIC Express Letters. 14. 961-969. 10.24507/icicel.14.10.961.

[18] https://github.com/HowieMa/DeepSORT_YOLOv5_Pytorch

[19] https://github.com/Sharpiless/Yolov5-deepsort-inference