Human Detection and Tracking with YOLO and SORT Tracking Algorithm

Tanveer Kader[1], Ahmad Fakhri Ab. Nasir[2]*, M. Zulfahmi Toh[3], Muhammad Nur Aiman Shapiee[4], Amir Fakarullsroq Abdul Razak[5]

Faculty of Computing, Universiti Malaysia Pahang Al-Sultan Abdullah (UMPSA), 26600 Pekan, Pahang, Malaysia[1, 2, 3]
Centre for Artificial Intelligence & Data Science (CAIDaS), Universiti Malaysia Pahang Al-Sultan Abdullah (UMPSA), Lebuh Persiaran Tun Khalil Yaakob, 26300 Gambang, Kuantan, Pahang, Malaysia[2]
Faculty of Manufacturing and Mechatronic Engineering Technology, Universiti Malaysia Pahang Al-Sultan Abdullah (UMPSA), 26600 Pekan, Pahang, Malaysia[4, 5]

***Abstract*—Human tracking is often performed on publicly available well annotated datasets, where the dataset development is always avoided because of the tiring process. Publicly available well-annotated datasets are ideal for training because those generate higher tracking accuracy. This study performs human tracking on videos recorded manually using optimized detectors following the tracking by detection framework. Manually recorded videos were used to develop a dataset which comprises more than 8k image sequences. Both indoor and outdoor scenarios were chosen to maintain different lighting conditions which make tracking difficult. All these image frames are labelled with bounding boxes for humans. The dataset is prepared by following the MOT15 dataset structure. A unique annotation process was performed that reduced the annotation labor by almost 80% which was a combination of manual annotation and prediction from pretrained models. Different sizes of You Only Look Once (YOLO) detection model (n/s/m) were trained using the train dataset focusing on humans and coupled with two most popular tracking algorithms: Simple Online Realtime Tracking (SORT) and DeepSORT. The YOLOv8 and YOLO11 models were optimized with proper hyperparameter values followed by tracking, using SORT and DeepSORT. The results were observed with those models on different confidence and Intersection over Union (IoU) threshold values. This study finds a proportional relation with the optimization of detection models and tracking accuracy. YOLO11m with DeepSORT tracker performed best on the test data with 74% Multiple Object Tracking Accuracy (MOTA) also the other optimized YOLO models tend to perform better with the trackers than the unoptimized ones.**

Keywords*—Human tracking; multiple object tracking; tracking-by-detection; you only look once (YOLO); simple online and realtime tracking (SORT)

I. Introduction

Human tracking is a popular research problem in the computer vision field which has a broader application in surveillance systems, human computer interaction and activity recognition. The main goal is detecting individuals across video frames with an assigned identity. This detection driven tracking system is known as Tracking by Detection (TBD). The problem becomes harder when it comes to track Multi-Object Tracking (MOT), where multiple individuals must be simultaneously tracked in a crowded environment. This throws challenges like handling occlusion referring to people overlapping or partially hidden, maintaining identity switches in crowded scenes, different lighting conditions and processing video efficiently for real-time applications. Some public datasets are available with these challenging scenarios, where most of the preprocessing like preparing frames, annotation, split of train and test set are already settled.

MOT based human tracking typically consists of several stages which are object detection, motion prediction, feature extraction, similarity calculation and data association. The detection stage identifies humans from frames and creates bounding boxes. Based on different tracking algorithms some predict motion using motion models and some uses different feature extractors to extract appearance features. From the predicted bounding boxes, it uses different metrics to calculate similarity between objects in the consecutive frames. The data association step links the detected humans across the frames with a consistent id assignment despite challenges such as occlusion, appearance changes and motion variations etc. One of the most common approaches for the association task is using a tracking by detection framework, where a detector identifies an object first, then a tracker associates the detection frame by frame using motion or appearance features. Many approaches are available for human tracking within MOT like object representation, motion modeling and feature extraction. Kalman filtering is widely used to predict object positions between frames and deep feature extractors which provide effective solutions for re-identification of object during occlusion and re appearance of humans. Observing the TBD framework this study poses the following research questions (RQ):

RQ1: Do manually developed dataset with the proposed annotation process provide optimal training for the detectors?

RQ2: To what extent does the optimization of the hyperparameters improve the detection along with tracking?

Nowadays a lot of deep learning-based detection models are available for the detection task which provides remarkable results in detection such as Faster R-CNN, You Only Look Once (YOLO) etc. Simple Online Realtime Tracking (SORT) [1] uses motion models and DeepSORT [2] adds appearance cues to the motion models to provide a better solution of the association task. In this study the detection models impact on the tracking performances is observed on our dataset. The

*Corresponding Author.

preliminary works on human detection can be referred in [3], [4], [5]. The dataset was developed from recorded videos by extracting the video frames and proper annotation process. This development follows the MOT15 [6] dataset standards by organizing the image frames as well as the annotated labels and ground truth files. From a total of twenty-eight minutes and five second videos, 8427 frames were prepared and annotated. The most popular YOLOv8 and YOLO11 object detection models with SORT and DeepSORT are used for tracking the association of the detected humans. This study will cover the details of the whole tracking by detection process followed in the experiments. First, the detection models were properly tuned and trained for humans on the train set. Then the trained models were used for the detection stage and lastly the tracking algorithms were plugged on top of that. All the results are displayed to observe the tracking performances on the optimized detection models.

The rest of the study is structured as follows: Section II summarizes and analyses the work that has been done till now in the tracking by detection paradigm. The development of the dataset and analysis is described in Section III. Section IV shows the whole methodology that includes detection models, tracking algorithms, training and testing. The results are discussed in Section V and finally conclude with the outcomes of the study in Section VI.

II. Literature Review

This section will briefly describe the previous work done related to human tracking on different scenarios and applications.

Foreground extraction that distinguishes between foreground and background is used to improve the tracking for an indoor environment [7]. This was done for home safety by building a system to track and detect people and analyze their behavior from indoor videos. Adaptive hybrid Multiple Human Tracking (AHMHT) [8] a technique which combines the concept of Gaussian Mixture Model (GMM) and Improved Adaptive K GMM (IAKGMM) into a single framework hence improving the tracking system in crowded scenes from public dataset PET2009. DeepSORT algorithm on top of different YOLO variants is a hotcake in the tracking by detection genre.

TABLE I Video File Properties

Sequences	Length (min:sec)	Frame Width	Frame Height	Frame Rate (fps)	Extracted Frames
Indoor Lobby 1	3:25	2688	1512	4.95	1013
Indoor Lobby 2	3:52	2688	1512	5.02	1162
Indoor Lobby 3	5:00	2688	1512	5.02	1502
Indoor Lobby 4	3:48	2688	1512	5.02	1145
Indoor Lobby 5	6:00	2688	1512	5.02	1801
Outdoor Entrance	3:00	2688	1512	5.02	902
Outdoor Parking	3:00	2688	1512	5.02	902
Total	28:05				8427

Azhar et al. uses YOLOv3 and DeepSORT for realtime people tracking system [9]. They used 3 different datasets YOLOv3, YOLOv3 tiny, YOLOv3 custom which they filtered for the person class only to detect human and concludes that providing an accurate dataset improves the tracking results. In a 5G infrastructure, multiple human tracking is performed using YOLOv3 and DeepSORT to track people from top view perspective [10]. Transfer learning is used by integrating a trained layer using a top view dataset. This approach improves overall tracking performance. Another people tracking is done by using YOLOv3 and YOLOv3-TINY models that extends to real time gender detection and tracking [11]. The models were trained on OpenImagesv5 dataset and a trade of between speed and accuracy is observed. The model was deployed on flask framework and tested the system on real world scenarios that achieved higher detection accuracy. Fang Yang et al compares YOLOv5 and YOLOv7 detectors for tracking with DeepSORT [12]. They tested the performance on MOT15 challenge dataset using DeepSORT on top of different sizes of YOLOv5 like small, medium, large and YOLOv7. Another comparison is done on tracking vehicles and humans on open access dataset of highway videos using SORT, DeepSORT and ByteTrack trackers on top of YOLOv5 [13]. They showed that ByteTrack surpasses other trackers when plugged with YOLOv5.

Some works improved their tracking system either by enhancing the YOLO detection model or modifying the tracking algorithm on MOT datasets.

Dimitrios et al. worked with the modified version of the DeepSORT for their real time multiple tracking of vehicles and pedestrians [14]. The YOLO models were trained on MSCOCO and UA-DETRAC datasets. The model was tested with modified DeepSORT on some mixture of scenes taken from MOT16 and MOT20. Besides, they provided a vehicle dataset of seven scenes. Mingwei Lei et al. follows the tracking by detection method for pedestrian detection and tracking YOLOv5-Lite and DeepSORT is used for the detection and tracking respectively [15]. They performed tracking using MOT16 and VERI-Wild datasets, where they obtained better results by enhancing the DeepSORT tracker. Another DeepSORT tweak was done for multi pedestrian tracking on top of YOLOv8 object detection model [16]. MOT16 and MOT17 datasets were used for testing. The method used omni-scale network (OSNet) for feature extraction and replaced intersection over union (IOU) with complete-intersection over union (CIOU) for association matching. Xueiting Jiang et al. performed multi pedestrian tracking on MOT15 and MOT16 datasets using YOLOX for enhanced detection [17]. Unscented Kalman Filtering (UKF) was integrated with FairMOT to track the detections across frames and provide better accuracy.

Most of these methods are performed on public datasets which are very well annotated. These nearly perfect datasets always tend to provide better results. The process of dataset development is skipped because of the complexity of the proper annotation. Also, optimization of hyperparameters may improve the detection component of the TBD system which may result in better tracking. Optimization is cost effective than developing or modifying a model. Hence, the role of optimization of object detectors should be more emphasized.

III. Dataset Development and Analysis

A. Recording

The dataset is prepared manually by recording videos, where people gather and walk frequently. In this case, the lobby, entrance and parking area of Faculty of Computing, UMPSA were chosen to record both indoor and outdoor scenarios (see Fig. 5). The focus is to find solutions for the human detection and tracking problem in campus. Video shots were taken by placing cameras in one corner to get full coverage of the indoor scenarios. For the outdoor footage camera was placed to get a wide view of the entrance and parking area.

A total of seven recorded videos were chosen for the human tracking work which sums up to 28 minutes and 05 seconds of videos. These videos were picked based on containing the challenges like groups, frequent passing of people, occlusions etc. All the videos were recorded in .mov video format. From these seven videos one video is taken from the entrance of the lobby area, another is from the parking area and the rest of the videos are taken from the lobby area. Table I shows the properties of the recorded videos. The resolution is 2688x1512 meaning the frame width and height are 2688 and 1512 respectively. The total frames exist in one second video known as frame rate in short fps. All the videos are taken in 5.02 fps except one which is 4.95 fps.

B. Frame Extraction

Extracting image sequences is an early essential part of tracking. This is a prerequisite task for the annotation process. Fig. 1 (top) shows the process of managing the image frames directory. The cv2 package from well-known python open cv library is used. Starts with opening a video file from the beginning and processes each frame then saves it with the corresponding frame number. For the management of all these extracted frames, the frames extracted from one video were saved in a subdirectory named after the video title inside of a parent directory that is named as image frames. By following this approach all the image frames were organized and were readily accessible for the later annotation process.

Fig. 1 (bottom) shows that from the videos, which got a minimum of 902 image frames to a maximum of 1801 image frames. From the selected sequence, a total of 8402 image frames were extracted which is suitable for the work. It also depicts the distribution of image frames over length of a video, where the six minutes video has the maximum image frames, and the three minutes videos have the minimum image frames which indicate the frames are extracted successfully from the videos and are ready for further processing.

C. Data Annotation

A unique approach to make the tiring annotation process interesting, easier and faster is selected by combining the manual annotation and prediction to make the annotation process faster. Fig. 2 demonstrates the annotation process done in ten steps. The latest pretrained YOLO model, YOLO11x, was executed. This is one of the latest and largest object detection models available till date for YOLO and it has the highest accuracy among the available YOLO models at the time of this research was conducted and directly used to predict humans from the image sequences. In YOLO, the object class for detecting humans is persons is present. This is the only class that has been defined as our goal is to detect and track humans. The labels are saved in .txt format for each image frame with its corresponding title. Then, the bounding boxes are generated on the images and saved them in a directory named as detection. From there each image frame is checked manually to identify the wrong and missed detections. If the human is not fully inside the bounding box or any human is not detected from the prediction, it is considered as error prediction. The popular image annotation tool labelImg was used to fix the wrong and missed detection. For fixing these errors, the original image frames from the image frames directory were opened and drew the bounding box around the humans carefully and saved it in YOLO detection format. Only frames that were correctly detected from each image sequence were selected through a process called sample selection, with any errors manually fixed using labelImg. Then with these few images a smaller YOLO11n model was trained and saved as best trained model.

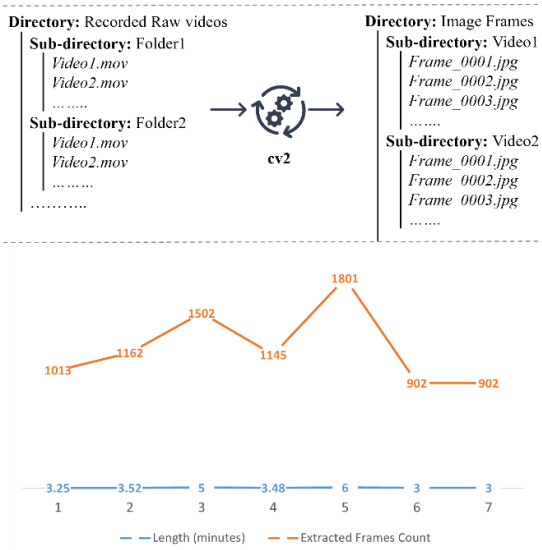

Fig. 1. Image frame extraction.

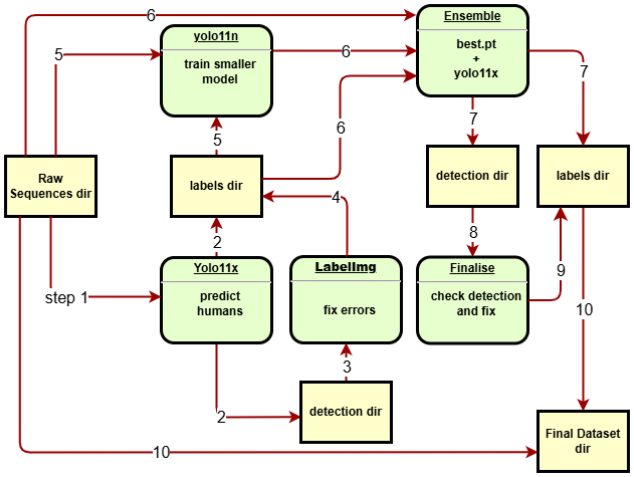

Fig. 2. Data annotation.

TABLE II DETECTION AND GROUND TRUTH FORMAT

Dataset	Detection	
MOT15	Format	frame, id, bb_left, bb_top, bb_width, bb_height, conf, x, y, z
	Example	1, -1, 1097, 463, 71, 124, 1, -1, -1, -1
ours	Format	class, x_center, y_center, width, height
	Example	0, 0.583449, 0.479252, 0.036709, 0.114276
	Ground Truth	
MOT15	Format	frame, id, bb_left, bb_top, bb_width, bb_height, conf, x, y, z
	Example	1, 1, 1097, 463, 71, 124, 1, -1, -1, -1
ours	Format	frame, id, bb_left, bb_top, bb_width, bb_height, conf, x, y, z
	Example	1, 1, 1097, 463, 71, 124, 1, -1, -1, -1

After that, an ensemble learning method was conducted by combining the trained model with YOLO11x and running predictions on the image frames again. Analyzing the detected frames, it was observed that the detections had improved this time. The detected frames were reviewed, and the dataset was finalized with proper annotation. By this approach, only approximately 250 frames per sequence had to be manually annotated on average. This means the tiring manual annotation was reduced by 80%, making the process more efficient and interesting. The detection labels were placed in the label's directory, separated for each image sequence with the corresponding sequence name. For example, for frame_0003.jpg image the detection label file was saved as frame_0003.txt.

D. MOT Standardization

For the clean workflow, the dataset was organized by following the MOT15 dataset structure with a little bit modification. The MOT15 contains a train and a test folder which contains image sequences. Each image sequence directory contains the image frames in img1 folder along with a det folder that contains the detections as det.txt. The path is "train/sequence_name/det/det.txt". Table II shows the detection and ground truth data format compared to MOT15. The MOT15 detection files contains ten values that are frame number, id number, bounding box (bb) left, bb top, bb width, bb height, and detection confidence. As the MOT15 is a 2D dataset the x, y, z values are indicated -1 also the id is not assigned in the det.txt and its set to -1. Every single line in the text file refers to an object in a frame. Our structure is a bit different than MOT15. The labels folder contains labels for each image in a txt file named according to the frame title. The path is "train/labels/sequence_name/frame_number.txt". Each file contains the detections for the objects in the image frames.

Our detection format aligns with YOLO detection format which is object class, x and y co-ordinates of the object center, width and height of the object. MOT15 contains ground truth file in the path "train/sequence_name/gt/gt.txt". The gt file contains the same values as the detection file along with the id number which is unique to a particular object. Our dataset also has a gt file for each sequence in the gt directory. Each line in the sequence refers to an object in a frame exactly like MOT15.

Unlike MOT15 dataset our dataset is divided into train, validation and test sets whereas MOT15 contains only train and test set. Four sequences were kept for training, two for validation, and the remaining one for testing. The dataset was split in a way so that both training and validation set contain both indoor and outdoor image sequences. Fig. 4 (left) depicts that the train set contains 3 indoor sequences and an outdoor sequence that contains 1013, 1145, 1502 and 902 frames respectively which sums up to 4562 frames. The validation set is a bit larger as it also contains a mixture of indoor and outdoor footages that have 1162 and 902 frames respectively. The remaining 1801 indoor video frames are kept for testing the trackers. Fig. 3 (right) demonstrates the percentages for each train, validation and test slices, which are roughly 54:24:21. Though a standard dataset split is considered as 70:20:10, ours is different for some good reasons. Firstly, the image sequence from one video is kept together to maintain the sequence of object movement. Some videos are lengthy so that the frame count is higher. A mixture of indoor and outdoor videos was kept in a set to maintain diversity and different lighting conditions. Another reason is to prevent overfitting during training. The validation set also contains indoor and outdoor video frames. The larger validation set will help the model to tune the parameters so that it does not memorize the train data rather than learning it. The longest video in the dataset is for testing the trackers performance because longer videos make it challenging to maintain object tracking throughout time, considering a good choice for testing the trackers.

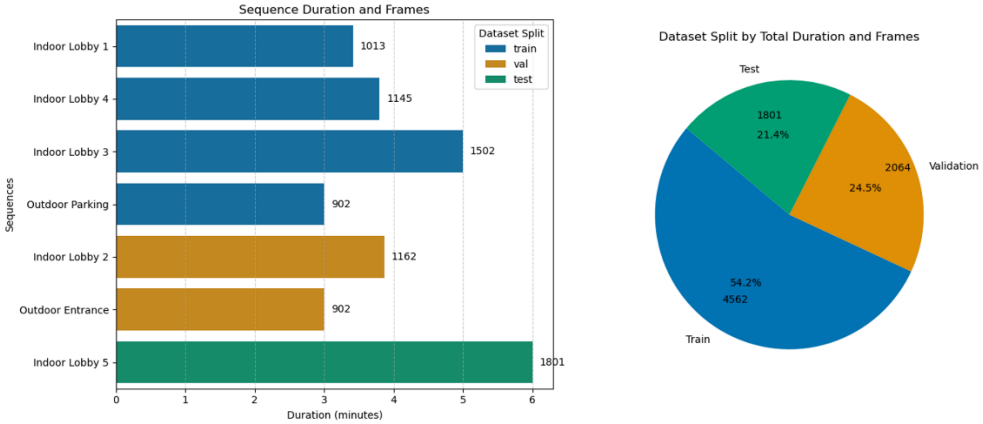

Fig. 3. Sequence distribution.

IV. METHODOLOGY

A. Detection Models

YOLOv8 is an improved object detection model compared to their previous versions. The advanced backbone and neck architecture improved the performance of feature extraction and object detection. Previous versions are mostly anchor based models meaning they use predefined bounding boxes. In contrast it uses an anchor-free split head that directly predicts object location, sizes and categories. This anchor free approach made the model's architecture simpler and computationally efficient. It modifies some key components of the CSPDarkNet backbone that helps to extract feature with fewer parameters and requires less computation while detecting larger objects. Also, the neck architecture with CSP Bottleneck with fusion (C2f) blocks made it a lightweight model.

YOLOv11 is the newer versions of YOLO variants that surpasses the previous ones in terms of feature extraction, optimized efficiency and speed with reduced parameters. The C2f blocks were replaced with C3K2 which implements Cross-Stage Partial (CSP) networks more efficiently. The Cross Stage Partial with Spatial Attention (C2PSA) networks which use a spatial attention mechanism and improve feature selection that helps in precise object localization. In comparison with YOLOv8, it uses 22% fewer parameters and achieves higher mAP on COCO dataset. These architectural advancements help to detect objects more accurately by improving focus on critical image regions.

YOLOv8 and YOLO11 both offer different sizes which are nano, small, medium, large, extra-large that are denoted as n, s, m, l and x. Each of these models offers trade of between speed and accuracy, providing better utilization of the resources. The smaller the model is the faster the speed is and uses less computations. The nano variant is optimized for speed and suitable for real-time applications using limited resources. The small variant is balanced between speed and accuracy, and the medium one prioritizes detection quality by spending more computational resources. The remaining two are the largest models that require high computational resources. The optimal size can be chosen according to the speed, size of the dataset and the computational resources.

B. Tracking Algorithms

SORT is a detection-based tracking system, where it leverages the power of CNN models to detect objects with more accuracy, hence improving the power of the tracking system. Faster region CNN (FrRCNN) is used here making this an end to end two state frameworks, where the feature extraction and proposed region passed to the second stage for classifying the object utilizing the power of parameter sharing. This is a motion-based tracking system and uses a linear constant velocity model, where velocity components are solved by Kalman filtering. Hungarian algorithms help to solve the id assignments optimally. A threshold value of Intersection over Union (IoU) filters out the redundant assignments. SORT uses a minimum frame count to assign id to detected object.

DeepSORT is an advancement of SORT that includes appearance-based feature extractor to generate more stable tracking. The combination of both motion and appearance cues for track association improves tracking in longer periods of occlusions. It utilizes a CNN pretrained on a large person reidentification dataset. This deep neural network generates feature vectors which then combines with IoU resulting in better tracking performance. The similarity metrics measure how similar two vectors are, and the max distance value ensures only objects with that minimum provided similarity are linked across frames. Identified objects are represented as feature vectors which helps to improve the reidentification problem. This integration of appearance features reduces id switches in cases like occlusion or object disappearance for a short time.

C. Tuning Detection Models for Humans

The YOLOv8 and YOLO11 models (n, s, m) were trained on the train set that consists of 4562 images as mentioned above. YOLO object detection models are multi object detectors that can detect a variety of objects. This study focuses on human tracking system so it will be unnecessary for YOLO models to detect all the objects rather the models will be set to only one class of object to detect humans which is persons. The training was conducted with optimized hyperparameters. Table III displays all the values for these optimized parameters in comparison with the baseline unoptimized parameters, where most of the parameters are not in use. As the training set is not heavy, several augmentation techniques were applied to make the models learn better. Mosaic and mixup augmentation techniques were applied. Mosaic takes four images and combines them into a single image. It resized each image, adds them together and takes random cutout of that image. And the mixup augmentation averages two images together and the bounding boxes are combined into the same list. More augmentation was applied to the frames such as rotation, flips and distortion. In terms of flipping, horizontal flipping was used, the vertical flip is off because our task is to focus on humans, where vertical flipping is not realistic at all. The perspective parameter helps to simulate real world scenario, where an object might be viewed from different angles and minimal value prevents distorting the image drastically. Then the models are trained on default optimizer Stochastic Gradient Descent (SGD) with a learning rate of 0.01 because it converges more slowly and generalizes better. The models are set to train for 100 epochs with a batch size of 32, but patience parameters are used to stop the training if performance does not improve, which is essential for tackling the overfitting problem as well as making the training cost efficient. Then the post processing parameters like confidence and intersection over union (IoU) threshold comes in play to filter and refine the detections, where the optimal values are 0.2 and 0.5 respectively. In contrast with the baseline settings no patience parameters were activated to minimize overfitting as well as the computational cost. The default confidence and IoU threshold values were 0.25 and 0.45 respectively that does not provide good recall which is necessary for tracking.

TABLE III HYPERPARAMETERS

Types	Parameters	Values Optimized	Values Baseline
Basic	epochs	100	100
	imgsz	640	640
	batch	32	16
	workers	4	2
Augmentation	mixup	0.2	0.0
	mosaic	0.5	1.0
	degrees	5.0	0.0
	translate	0.15	0.0
	scale	0.3	0.0
	fliplr	0.6	0.5
	flipud	0.0	0.0
	perspective	0.0005	0.0
L2 regularization	weight_decay	0.0005	0.0
	droupout	0.1	0.0
Post Processing	conf	0.2	0.25
	iou	0.5	0.45
Others	patience	10	0
	half	true	false

D. Tracking Pipeline

Fig. 4 shows the full tracking process divided into four major stages. All the stages are described below one by one.

1) Stage 1: Setup and configuration: The project structure was set up with all the required items that have been prepared. The final dataset containing image sequences, labels and ground truth file added to the main project pipeline. Then, all the custom trained optimized models were placed in one place so that the models can be switched easily for the experiments. The tracking was executed from a notebook named as main.ipynb, where all the file paths for the necessary inputs and outputs were defined. It was also responsible for passing the post processing parameters to the main tracker file. A run_tracker.py file contained all the custom setup for initialization and customization for utilizing the trackers. The tracking results were organized in the results directory with separate files for easing the later evaluation process.

The YOLO custom trained optimized models were provided from the detection model's directory by the path defined in the main file. Also, the paths for the test sequences and output results are loaded in the detection pipeline. Values of the key parameters like confidence threshold and (IoU) are initialized and passed to the detection model to predict humans from the frames. The models were tested for different confidence and IoU threshold values to get optimal performance. All these operations were performed from a single common notebook which initiates the detection with the different parameters and performs the tracking.

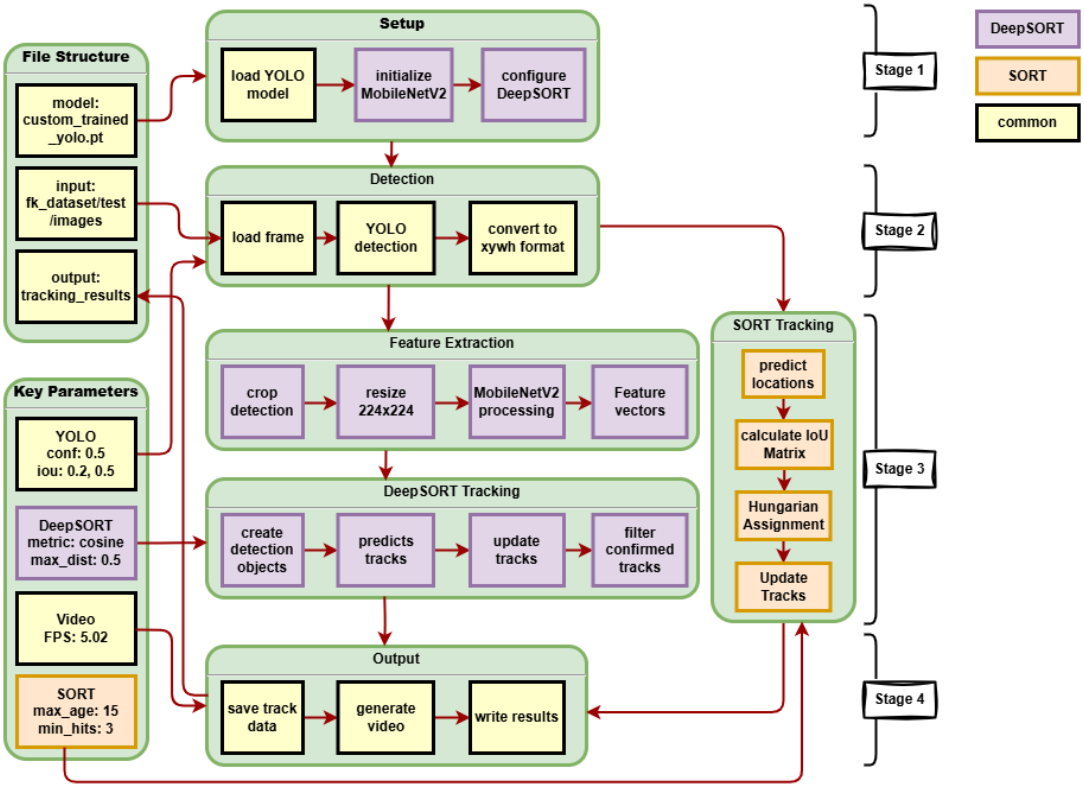

Fig. 4. Tracking pipeline.

2) Stage 2: Human detection: The custom trained YOLO detection models were loaded to identify humans from the test sequences. It started with loading the frames sequentially from the input directory to process each frame through the custom trained YOLO models. The models generated detections in 'xyxy' format which represents bounding boxes with two points, (x1, y1) and (x2, y2) that denotes the top-left and bottom-right corners respectively. For a bounding box with values [100, 200, 150, 300] the top-left corner is (100, 200) and bottom-right corner is (150, 300). These detections were filtered by the key parameters like confidence and the IoU threshold to eliminate the weak and redundant detections. The detections were then changed to YOLO's 'xyxy' format to 'xywh' format for the tracker compatibility, where (x, y) represents top-left corner and the (w, h) represents the width and height of the bounding box. The conversion was performed by calculating w =x2 - x1 and h = y2 - y1, resulting in the format [100, 200, 50, 100].

3) Stage 3: Tracking detected humans

a) YOLOv8/YOLO11(n/s/m) with SORT: Total six combinations of YOLO and SORT trackers were tested. Three key parameters were passed to maintain the tracking function: max_age = 15 to delete the unmatched tracks after fifteen frames, min_hits =3 which establish a track after three consecutive detections and IoU threshold for associating detection with prediction. Assigned detection updates the Kalman filter state. The unmatched detections start new tracks and the remaining tracks without detection for several frames were deleted.

b) YOLOv8/YOLO11(n/s/m) with DeepSORT: With an exception to the SORT, DeepSORT went through the feature extraction model to associate appearance feature. DeepSORT uses deep neural networks for feature embeddings. In our case the MobileNetV2 pre-trained model was used for this feature extraction task. First, the algorithm cropped the bounding box region from the frame. Then resized it to 224x224 dimension. These cropped images were passed through the deep CNN model which is MobileNetV2 to generate feature vectors. This is an additional layer of tracking that adds appearance feature beside IoU matching. Then it computes cosine similarity metric to measure how similar two vectors are, and the max distance value ensures only objects with that minimum similarity are linked across frames. Thus, DeepSORT improves the re-identification model.

4) Stage 4: Outputs: The tracking components take the processed detections and maintain consistent identification of detected humans across frames. The trackers mainly take the detections as inputs for each frame as mentioned above in the 'xywh' format for each frame. Then the tracker assigns ids for each unique human and outputs the bounding box values with the additional track ids. Finally, the tracking result contains frame number, assigned id, and bounding box. For example, with a given 'xywh' bounding box format for a frame [100, 200, 50, 100] along with object class, confidence score, the tracker assigns a track id for this bounding box as the output. Then the final tracking result is saved in the MOT15 format that was displayed earlier in Table II, where it contains frame number, track id, x, y, width, height and confidence score so that it can be evaluated for the performance. This pipeline also provided these data along with videos of the tracked humans for each model in the output directory named after the corresponding detection model and tracking algorithm.

E. Evaluation Metrics

The results were saved in the tracking_results directory by the sequence and model name for evaluation. There are many metrics available for evaluating tracking results.

1) MOTA (multiple objects tracking accuracy): It is the principal metric to measure the performance of a tracker. Eq. (1) shows that it combines three error sources like false positive (FP): Tracked object that doesn't match any ground truth, false negative (FN): Ground truth object that were not tracked, identity switches (IDSW): The number of times the tracked object switch incorrectly.

$$MOTA = 1-(FN+FP+IDSW)/GT \quad (1)$$

2) MOTP (multiple objects tracking precision): Measures the precise localization of the tracked objects by calculating average overlap between ground truth and tracked bounding boxes. It only takes true positive detection into consideration. Higher value indicates better bounding box precision. ID F1 score (IDF1): It indicates how long the tracker correctly identifies an object. It measures the assignment between prediction and ground truth objects across the video. Mostly Tracked (MT): It measures the objects that are tracked for more than 80 per cent of the time. Partially Tracked (PT): It measures the objects that are tracked for 20 to 80 per cent of the time. Mostly Lost (ML): It indicated the objects that are tracked less than 20 per cent of the time. The results were evaluated using all these metrics and compared the results to find out the best combination of detection model and tracking algorithm with proper parameters.

V. Result and Discussion

A. Detection Models

Fig. 5 illustrates the detected human on several images. Fig. 6 shows performance of the yolo models of different sizes. The ones marked as "_uo" indicates the unoptimized ones. It displays all the main performance metrics like precision, recall, mAP50 and mAP50-95 to portrait a better picture of the model performance. All these versions of YOLO model were trained with the described hyperparameters. The results show that the unoptimized versions, v8m_uo and 11m_uo gained the highest precision of 98% and 97% but lower recall of 64%. Where the optimized v8m and 11m gained 67% and 83% recall maintaining a decent precision of 91% and 97% respectively.

The well annotation and organization of dataset helped the models learn earlier and due to the proper hyperparameter settings the models did not overfit the training data resulting balanced performance in both precision and recall. On the other hand, unoptimized models were overtrained for all the

100 epochs which made them memorize the data rather than learning. This situation caused these models to increase in precision but poor score in recall. The YOLO11m gets the best scores for all the metrics, indicating this is the best trained detection model for our dataset. As our pipeline follows the TBD method, the assumption is to provide better results with the well-trained detection models.

Fig. 5. Sample of raw data at left and detected person at right, (a) Entrance, (b) Lobby, and (c) Parking area.

B. Human Tracking

The human tracking performed on our dataset by applying SORT and DeepSORT trackers on top of different sizes of YOLOv8 and YOLO11 which are nano, small and medium. Table IV shows a comparison of the tracking performance on different confidence and IoU threshold values for both YOLO models of different sizes. Observing the results, the medium sized YOLO models performed better than the small and nano ones. Fig. 7 presents successful recognition of identity of the tracked human between frames even after long time full human occlusion.

The YOLOV8m and 11m provided higher MOTA for both SORT and DeepSORT trackers. The higher IoU threshold of 0.5 provided the best results according to MOTA of 34.5% and 74% for both SORT and DeepSORT respectively. These models increased the MOTA, MOTP, IDF1, MT values and decreased the IDSW, FN and ML values.

In contrast, the unoptimized models obtained lower scores in all these matrices. With the similar IoU of 5, the unoptimized YOLOV8m and 11m models show only 20.9% and 33.4% MOTA for SORT and DeepSORT individually. Improvements in both tracking algorithms can be observed with the other optimized models like v8m, 11m compared to the unoptimized ones. In IoU of 0.2 optimized models increased MOTA from 12.2% to 28.3% (v8m), 21.2% to 30.8% (11m) for SORT and 23.4% to 54.6% (v8m), 33.3% to 71.5% (11m) for DeepSORT. Similarly, for IoU of 0.5 the increment was from 12.3% to 28.1% (v8m), 20.9% to 34.5% (11m) for SORT and 33.4% to 74% (11m) for DeepSORT. Moreover, in some cases the optimized nano models can beat all other unoptimized medium ones e.g., 11n with DeepSORT using IoU value 0.5 surpassed all the other unoptimized medium models by achieving 32.7% MOTA. The increased IoU threshold helps to improve the overall performance of the trackers. All combinations are ranked in the table based on their performance.

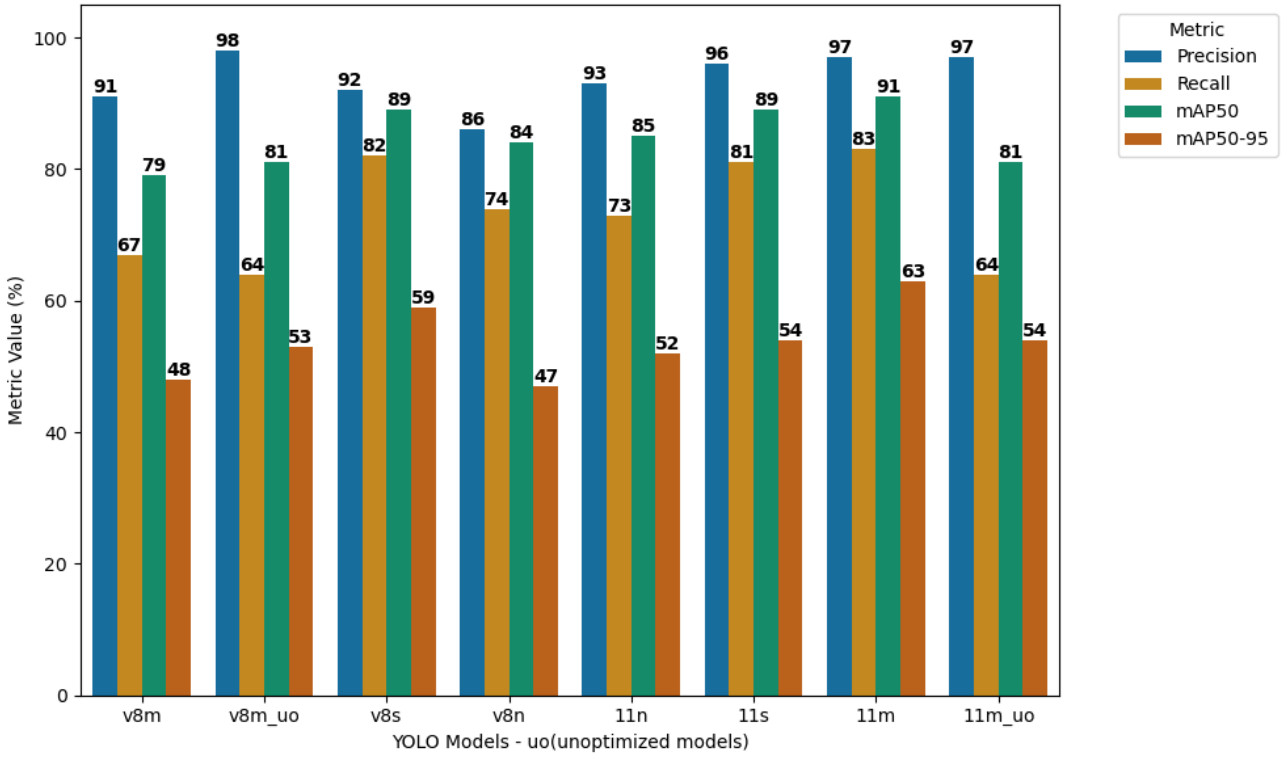

Fig. 6. YOLO validation results.

TABLE IV TRACKING PERFORMANCE

YOLO	MOTA ↑	MOTP ↑	IDF1 ↑	IDSW ↓	FP↓	FN↓	MT ↑	PT	ML↓
SORT (confidence threshold = 0.5, IoU threshold = 0.2)									
v8n[5]	22.1	19.4	26.8	**6**	86	**2103**	12	38	145
v8s[4]	22.6	25.2	25.6	18	218	3034	14	82	**134**
v8m[2]	28.3	21.0	31.0	14	58	3112	8	67	167
v8m_unoptimized*	12.2	19.2	19.1	8	446	3852	12	85	146
11n[3]	22.9	17.4	26.8	9	77	2444	16	44	155
11s[6]	20.3	**26.7**	23.5	12	264	3292	**23**	76	142
11m[1]	**30.8**	22.2	**33.7**	13	**51**	3573	10	93	143
11m_unoptimized*	21.2	21.3	24.6	6	301	2817	23	76	131
11s_100[7]	20.3	26.7	23.5	12	264	3292	23	76	142
SORT (confidence threshold = 0.5, IoU threshold = 0.5)									
v8n[7]	22.1	19.4	26.8	**6**	86	**2103**	12	38	145
v8s[3]	23.6	25.5	25.5	17	239	3176	**20**	79	139
v8m[2]	28.1	21.0	30.5	12	58	3171	9	66	169
v8m_unoptimized*	12.3	19.4	19.1	10	449	3845	11	85	147
11n[4]	23.0	17.4	26.8	10	78	2428	15	45	155
11s[5]	22.7	**27.2**	25.4	18	263	3170	**20**	83	**136**
11m[1]	**34.5**	22.5	**35.7**	10	**51**	3192	12	95	140
11m_unoptimized*	20.9	21.3	23.9	11	302	2855	18	85	126
11s_100[6]	22.7	27.2	25.4	18	263	3170	20	83	136
DeepSORT (confidence threshold = 0.5, IoU threshold = 0.2)									
v8n[7]	29.5	19.8	32.0	**4**	386	4306	25	35	206
v8s[3]	41.8	25.4	31.7	28	826	3168	56	52	169
v8m[2]	54.6	21.4	45.1	11	286	2840	58	42	177
v8m_unoptimized*	23.4	19.9	29.7	35	1545	3712	71	53	153
11n[6]	32.6	19.0	34.0	20	397	4243	37	41	199
11s[4]	39.3	**26.3**	36.7	22	916	3258	59	58	**160**
11m[1]	**71.5**	22.5	**52.5**	21	**286**	**1661**	**63**	50	164
11m_unoptimized*	33.3	21.1	33.5	22	1079	3507	66	45	166
11s_100[5]	39.0	26.3	36.5	23	927	3267	59	57	161
DeepSORT (confidence threshold = 0.5, IoU threshold = 0.5)									
v8n[7]	29.5	19.8	32	**4**	386	4306	25	35	206
v8s[3]	46.8	25.3	36.2	28	871	2782	60	55	162
v8m[2]	54.8	21.5	45.6	12	295	2818	57	43	177
v8m_unoptimized*	23.4	19.9	30.4	32	1561	3697	72	53	152
11n[6]	32.7	19.1	34.5	18	392	4237	38	40	199
11s[4]	43.2	**26.9**	37.5	37	948	2942	63	61	153
11m[1]	**74.0**	22.5	**54.2**	29	**293**	**1477**	**71**	46	160
11m_unoptimized*	33.4	21.1	34.5	23	1083	3496	65	45	167
11s_100[5]	43.1	26.9	37.4	39	962	2936	63	62	**152**

*Baseline default detectors

The fine-tuned models with optimal hyperparameters always provide better results. Despite showing higher precision, the unoptimized overtrained detection models fall behind compared to the optimized one. Unoptimized models fail to learn rather than the memorization of the data provides higher precision but in terms of recall they fall behind. That causes tracking failure and poor performance. On the other hand, optimized detection models obtain good balance in precision and recall resulting in higher tracking accuracy.

Fig. 7. Sample of tracked human before, during, after (from top) occlusion.

VI. Conclusion

This study validates the tracking by detection system by observing the tracking results with multiple detection models fine-tuned on custom recorded datasets with optimal hyperparameters. The training process with custom-developed dataset provided good detection results. The longer test video sequences put stress on the detection and tracking algorithm makes the task challenging for frequent occlusion and reappearance of the human. The results show that the nano models provide the minimum IDSW, but it doesn't detect many humans thus providing higher FN. By analyzing all the performance scores with different hyperparameters and different sized models, the medium sized model fits better for our dataset and the DeepSORT tracker stays ahead of all as it leverages the pretrained feature extractor model MobileNetV2 which helps to add the appearance features with the IoU matching. Moreover, the tracking results are always better for the optimized detection models. Even optimized nano models can perform better than unoptimized larger ones. Hence an improved optimized detection system makes higher tracking accuracy.

Fully manual process of dataset annotation (labelling each object manually using only annotation tool e.g. labelImg) may provide better bounding boxes which can lead to better training for the detection models hence improving detection as well as tracking performance. Besides, it might be possible to find better hyperparameters values through long range of hyperparameter tuning. All the experiments for this study were performed in Google Colab which limits the experiments from going to long range of values for the parameters due to short time connectivity and computational complexity. Only short-range limited values were tested for the optimization.

Acknowledgment

The authors would like to thank the Ministry of Higher Education for providing financial support under Fundamental Research Grant Scheme (FRGS) No. FRGS/1/2023/ICT02/UMP/02/3 (University reference RDU230117).

References

[1] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in 2016 IEEE International Conference on Image Processing (ICIP), Sep. 2016, pp. 3464–3468. doi: 10.1109/ICIP.2016.7533003.

[2] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in 2017 IEEE International Conference on Image Processing (ICIP), Sep. 2017, pp. 3645–3649. doi: 10.1109/ICIP.2017.8296962.

[3] T. J. Cheng, A. F. A. Nasir, A. P. P. A. Majeed, M. A. M. Razman, and T. L. Lim, "Vision-based Human Detection by Fine-Tuned SSD Models," Int. J. Adv. Comput. Sci. Appl. IJACSA, vol. 13, no. 11, Art. no. 11, 30 2022, doi: 10.14569/IJACSA.2022.0131143.

[4] T. J. Cheng, A. F. Ab. Nasir, A. P. P. Abdul Majeed, L. T. Li, and I. Mohd Khairuddin, "CenterNet: A Transfer Learning Approach for Human Presence Detection," in Advances in Intelligent Manufacturing and Robotics, A. Tan, F. Zhu, H. Jiang, K. Mostafa, E. H. Yap, L. Chen, L. J. A. Olule, and H. Myung, Eds., Singapore: Springer Nature, 2024, pp. 41–51. doi: 10.1007/978-981-99-8498-5_4.

[5] J. C. Tang, A. F. B. A. Nasir, A. P. P. A. Majeed, L. L. Thai, M. A. M. Razman, and I. M. Khairuddin, "Fine-tuned RetinaNet models for Vision-based Human Presence Detection," Mekatronika J. Intell. Manuf. Mechatron., vol. 4, no. 2, Art. no. 2, Nov. 2022, doi: 10.15282/mekatronika.v4i2.8850.

[6] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking," Apr. 08, 2015, arXiv: arXiv:1504.01942. doi: 10.48550/arXiv.1504.01942.

[7] C.-J. Yang, T. Chou, F.-A. Chang, C. Ssu-Yuan, and J.-I. Guo, "A smart surveillance system with multiple people detection, tracking, and behavior analysis," in 2016 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Apr. 2016, pp. 1–4. doi: 10.1109/VLSI-DAT.2016.7482569.

[8] P. Karpagavalli and A. V. Ramprasad, "Automatic multiple human tracking using an adaptive hybrid GMM based detection in a crowd," Multimed. Tools Appl., vol. 79, no. 39, pp. 28993–29019, Oct. 2020, doi: 10.1007/s11042-019-08181-0.

[9] M. I. H. Azhar, F. H. K. Zaman, N. Md. Tahir, and H. Hashim, "People Tracking System Using DeepSORT," in 2020 10th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Aug. 2020, pp. 137–141. doi: 10.1109/ICCSCE50387.2020.9204956.

[10] I. Ahmed, M. Ahmad, A. Ahmad, and G. Jeon, "Top view multiple people tracking by detection using deep SORT and YOLOv3 with transfer learning: within 5G infrastructure," Int. J. Mach. Learn. Cybern., vol. 12, no. 11, pp. 3053–3067, Nov. 2021, doi: 10.1007/s13042-020-01220-5.

[11] Z. M. Peerun and R. K. Moloo, "Real-time gender and people tracking using YOLO," in 2024 Sixth International Conference on Computational Intelligence and Communication Technologies (CCICT), Apr. 2024, pp. 448–454. doi: 10.1109/CCICT62777.2024.00079.

[12] F. Yang, X. Zhang, and B. Liu, "Video object tracking based on YOLOv7 and DeepSORT," Jul. 25, 2022, arXiv: arXiv:2207.12202. doi: 10.48550/arXiv.2207.12202.

[13] M. Abouelyazid, "Comparative Evaluation of SORT, DeepSORT, and ByteTrack for Multiple Object Tracking in Highway Videos," Int. J. Sustain. Infrastruct. Cities Soc., vol. 8, no. 11, Art. no. 11, Nov. 2023.

[14] D. Meimetis, I. Daramouskas, I. Perikos, and I. Hatzilygeroudis, "Real-time multiple object tracking using deep learning methods," Neural Comput. Appl., vol. 35, no. 1, pp. 89–118, Jan. 2023, doi: 10.1007/s00521-021-06391-y.

[15] W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang, and Y. Wang, "End-to-end Active Object Tracking via Reinforcement Learning," in Proceedings of the 35th International Conference on Machine Learning, PMLR, Jul. 2018, pp. 3286–3295. Accessed: Oct. 28, 2024. [Online]. Available: https://proceedings.mlr.press/v80/luo18a.html

[16] W. Sheng et al., "Multi-objective pedestrian tracking method based on YOLOv8 and improved DeepSORT," Math. Biosci. Eng., vol. 21, no. 2, Art. no. mbe-21-02-077, 2024, doi: 10.3934/mbe.2024077.

[17] X. Jiang, J. Li, H. Jia, W. Xu, and R. Li, "Improved FairMOT for multi-pedestrian tracking in complex environments," in 2024 IEEE 6th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), May 2024, pp. 287–291. doi: 10.1109/IMCEC59810.2024.10575359.