Research and Application of Loitering Detection Based on Deep Learning

Yuyang Zhang 1, Jiangcheng Wang 1, Xiaobing Wang 1,*, Yingnan Wang 2, and Sijiang Yu 3

1Xidian University, Xi’an, Shaanxi, China
2Shaanxi Normal University, Xi’an, Shaanxi, China
3Xi’an University of Posts and Telecommunications, Xi’an, Shaanxi, China
zyy_xdu2021@163.com, 1239974410@qq.com, xbwang@mail.xidian.edu.cn, wyn_smu2021@163.com, ysj1026@163.com
*corresponding author

Abstract—Dangerous events in public places often have abnormal loitering behavior of pedestrians before they happen. It is significant to improve public safety to find out this kind of abnormal behavior. At present, the model of deep learning-based wandering detection algorithm is too complex and requires the support of devices with high computing power, otherwise it is difficult to be applied to scenes with dense pedestrians, so in-depth research on this problem is needed. In response to YOLOX’s low detection accuracy for small objects, YOLOA adds spatial and channel attention modules to learn discriminative deep features, and adds a path aggregation network to process both semantic and spatial information in the early stages of network. YOLOA also adds the measuring the similarity of width and height to IoU loss, which improve the prediction accuracy of object position. The experimental results show that the accuracy of YOLOA is 2% higher than that of YOLOX in small object detection of the COCO dataset. For scenes with dense pedestrians, JYTracker integrates object detection and pedestrian re-identification, improves prediction speed by sharing underlying features improving the training speed. JYTracker also increases tracking speed through efficient data association strategies. The experimental results also show JYTracker model’s MOTA on the MOT20 dataset is as high as 69.5. The experimental results on the PETS2007 dataset show that the JYLD can quickly identify (any) loitering behaviors.

Keywords—Deep Learning; Object Detection; Multiple Object Tracking; Loitering Detection

1. INTRODUCTION

Nowadays, video surveillance has been applied in all kinds of places, greatly increasing the number of monitors in the public, which plays a vital role in maintaining social stability. It is impractical to rely on manual processing of massive video data, so people urgently need intelligent video surveillance to replace manual processing of video streaming data in time. Abnormal loitering[1] refers to the continuous movement of pedestrians in a certain area without obvious displacement. It is a kind of basic abnormal behavior that generally accompanies many abnormal behaviors, such as suspicion of bank robbery, terrorist attack and pickpocketing. Therefore, early detection of loitering behavior can reduce the occurrence of crime. The application of loitering detection method to intelligent monitoring can reduce the time of manual monitoring and save a lot of manpower and material resources. Generally, the accuracy of loitering detection algorithm depends on the accuracy of extracting the trajectory of object tracking[2], in return, the latter depends on the result of target detection. Therefore, the research of the object detection and object tracking is exactly necessary.

Object detection is a fundamental and important technology in computer vision. With the improvement of deep learning, object detection algorithms based on deep learning can be divided into two main development directions: Anchor-Based and Anchor-Free. Anchor-based requires clustering or manual setting of several Anchor boxes with different sizes and aspect ratios on data sets. Two-step algorithm was adopted in Anchor-Based in the early stage. RCNN[3] is the first group of two-step detectors based on Anchor box. It first selects possible object boxes from the input image through selective search algorithm, and then takes these fixed size pairs of objects as the input of CNN to extract features. Finally, the extracted features are fed into the categorizer to predict foreground as well as background and further predict specific categories. Faster R-CNN[4] was the first end-to-end detector to achieve real-time performance. Region Proposal Network (RPN) was innovatively proposed for output of preliminary candidate bounding box, which greatly improved the precision of subsequent prediction boxes. Afterwards, one-step algorithm that meets the requirements of real-time monitoring emerged, YOLO v1[5] is the first one-step detection algorithm. This algorithm divides the image into multiple square grids with the subsampling scale, predicts the boundary box of the target with the center in the grid, and also gives the probability of the object contained in the grid and the category of the object. YOLO v3[6] uses DarkNet53 as the backbone network, uses Logistic function instead of Softmax function, and uses feature pyramid network to detect objects with different area sizes through three branches, which significantly improves the performance of YOLO. YOLO v4[7], YOLO v5[8] are very similar in that both heavily integrate State-of-the-art in computer vision, thus significantly improving YOLO's detection performance on targets. Anchor-free can directly predict the position and size of objects without Anchor box, greatly reducing network hyperparameters. YOLOX[9] used in this paper is the representative of Anchor Free, including decoupled head, SimOTA, Mosaic data enhancement and other methods.
Object tracking based on deep learning has good environmental adaptability and robustness. Qi et al. [10] proposed a tracking framework based on CNN, which makes several CNN-based trackers form a more powerful tracker through adaptive methods. Enlightened by Transformer, Chen [11] et al. proposed a novel feature fusion network using Self-Attention, which effectively combines the template and search area features. This is achieved through an ego-context enhancement module based on Self-Attention and a cross-feature augment module based on Cross-Attention.

Loitering detection is a significant research field in deep learning. Lim [12] proposed an intelligent framework for detecting multiple events, but the video content needs to be partitioned by attributes. Nayak [13] et al. proposed a Loitering Detection System (LDS) based on deep learning. The system includes object detection and feature extraction, camera switching and loiterer recognition. YOLOv3 is used as the detector, and DeepSORT [14] is used to track the pedestrian. Finally, starting with the pedestrian trajectory, the pedestrians meeting the time and displacement threshold are regarded as loiterers.

In this paper, a loitering detection algorithm JYLD is proposed, which improves the precision of small object detection by enhancing the YOLOX object detection algorithm. The object tracking algorithm JYTracker combines object detection and person re-identification, increases the prediction speed by sharing the low-level features for pedestrian intensive situations, and improves the tracking speed by using an efficient data association strategy.

2. THE NETWORK ABOUT YOLOA

Based on YOLOX, this study proposes the target detection algorithm YOLOA, which will be described from five aspects: attention mechanism, backbone network, FPN, and detection head objective function. Finally, the effectiveness of the YOLOA method is proved through comparative experiments. The previous evaluation found that on the COCO2017 dataset, the AP of YOLOX-S on the large, medium and small targets were 54.1%, 44.8%, and 23.3%, respectively. Based on the fact that YOLOX-S has a significant gap in the detection performance between small targets and large targets, this research improves the detection performance of small targets, which may greatly improve the overall detection accuracy.

2.1. Attention Module

This paper designs three modules based on attention mechanism, namely: Channel Attention Module (CAM), Spatial Attention Module (SAM) and Channel Attention Module Based on BN(CBM), as shown in Fig1, the full connection is represented by 1x1 convolution.

CAM acts on the channel dimension of the image. First, the input is passed through the max-pooling to obtain a feature map with a width and height of 1 and the number of channels is the same as the number of input channels, and then sent to the fully connected layer through the Flatten layer to obtain attention on the channel. The attention score is multiplied by the input to get the output.

SAM acts on the width and height dimensions of the image. First, the input is max-pooled in a channel direction to obtain a feature map with the same width and height as the input, but only one channel. Then after convolution, batch normalization and Sigmoid activation function, the weight value is between 0-1 to obtain the attention score, and finally multiplied by the input to obtain the output.

Batch normalization can stabilize the data distribution, thereby speeding up learning and making training more stable. The input feature image is standardized with a mean of 0 and a standard deviation of 1 on the batch channel. Its formula is as follows:

$$BN(x) = \frac{\gamma \cdot x - \mu}{\sigma_B} + \beta, x \in B$$ (1)

The scaling factor $\gamma$ is learnable and by this factor enables the data to recover as much of its own expressiveness as possible. CBM uses this coefficient as the input of the fully connected network to obtain the attention score on the channel, and finally multiplies it with the result of the batch normalization layer to obtain the output feature.

2.2. Backbone Network

- Improvement of DarkNet based on ConvNeXt: ConvNeXt exceeds the accuracy of the Transformer while maintaining the simplicity and effectiveness of the CNN.

2.3. Bottleneck

Figure 2. Comparison of the basic structure of CSPDarkNet and ConvNeXt Block
As can be seen from Figure 2, ConvNeXt improves the training speed through fewer activation functions and normalization layers, and uses $7 \times 7$ convolution kernels to convolve in groups, reducing the amount of calculation. The LN Layer refers to Layer Normalization, which is used for training RNNS regardless of sample batch size. This article modifies DarkNet with CovNeXt Block, and names the changed backbone network as DarkNeXt. Its configuration parameters are as follows:

- DarkNeXt: $C = (128, 256, 512, 1024)$, $B = (3, 3, 9, 3)$
- $C$ represents the number of channels and $B$ represents the number of blocks.

**Improvement of DarkNet based on ConvNeXt:** ConvNeXt exceeds the accuracy of the Transformer while maintaining the simplicity and effectiveness of the CNN.

**Improvement of DarkNet based on Focus:** In YOLO v5, the picture will be sliced after being input to the backbone network. The specific operation is to sample a pixel value every other pixel of the input picture, from left to right, from top to bottom, and get four complementary information, the same width and height features, without loss of information. The slice operation is shown in Figure 3.

2.3. Feature Pyramid Networks

Feature Pyramid Networks (FPN) [15] focuses more on the interaction between high-level semantic information and underlying airspace information. Compared with the backbone network, FPN is more important for target detection tasks. Figure 5 below shows the link aggregation (Path Aggregation, PA) FPN and two improved PAFPNs used in YOLOX. PAFPN1 and PAFPN2 hope that semantic information and spatial information can be processed with the same priority in the early stage of the network, making it more efficient for detection.

2.4. Head

The head is used to predict the type and position of the target. CAM has been added to the head, aiming to focus on the most relevant features depending on the target, and the structure of the detection head is shown in Figure 6.

2.5. Loss Function

IoU refers to the ratio of the intersection and union of the predicted bounding box and the real bounding box, as shown in Figure 7:
Bounding box regression uses the overlapping area between the predicted bounding box and the real bounding box as a loss function, called IoU Loss, and the formula is as follows:

\[
L_{IoU} = 1 - IoU^2
\]

(3)

Based on Formula 3, this study normalizes the distance between the predicted frame and the center point of the real frame, and considers the influence of aspect ratio on convergence, so the following formula is obtained:

\[
L_{CIoU} = 1 - IoU + \frac{d^2}{c^2} + \alpha v
\]

(4)

\(\alpha\) is the weight coefficient, \(v\) is the similarity of measuring aspect ratio, the formula is as follows:

\[
v = \frac{4}{\pi^2} \left( \arctan \left( \frac{w_{gt}}{b_{gt}} \right) - \arctan \left( \frac{w}{b} \right) \right)^2
\]

(5)

In this paper, Formula 4 is used to train the regression task.

2.6. Experimental results

CSPDarkNeXt, PAFPN1, detection head, and regression loss function were combined and named YOLOA, which surpassed YOLOX. The experimental comparison results of five YOLOX-based improved models and YOLOA are shown in Tables 1-4.

### Table 1. Comparison results of object detection algorithms (AP)

<table>
<thead>
<tr>
<th>Method</th>
<th>Size</th>
<th>Parm</th>
<th>(AP_5)</th>
<th>(AP_{50})</th>
<th>(AP_{95})</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOLOX-S</td>
<td>640</td>
<td>8.97</td>
<td>40.5</td>
<td>59.3</td>
<td>43.7</td>
</tr>
<tr>
<td>YOLOX-S +SAM+CAM</td>
<td>640</td>
<td>9.44</td>
<td>40.7</td>
<td>60.5</td>
<td>44.2</td>
</tr>
<tr>
<td>YOLOX-S +SAM+CBM</td>
<td>640</td>
<td>9.14</td>
<td>40.2</td>
<td>58.7</td>
<td>43.9</td>
</tr>
<tr>
<td>YOLOX-S +PAFPN1</td>
<td>640</td>
<td>9.11</td>
<td>41.0</td>
<td>59.9</td>
<td>44.7</td>
</tr>
<tr>
<td>YOLOX-S +PAFPN2</td>
<td>640</td>
<td>9.63</td>
<td>40.9</td>
<td>59.4</td>
<td>44.7</td>
</tr>
<tr>
<td>YOLOX-S +DarkNeXt</td>
<td>640</td>
<td>9.38</td>
<td>40.6</td>
<td>59.8</td>
<td>44.3</td>
</tr>
<tr>
<td>YOLOX-S +CSPDarkNeXt</td>
<td>640</td>
<td>8.10</td>
<td>40.3</td>
<td>59.5</td>
<td>44.0</td>
</tr>
<tr>
<td>YOLOA-S</td>
<td>640</td>
<td>9.58</td>
<td>41.1</td>
<td>60.1</td>
<td>45.1</td>
</tr>
</tbody>
</table>

The experimental results show that the AP of the object detection algorithm with the attention mechanism is close to that of YOLOX-S, but the detection accuracy of small targets is improved by 2%, and it has obvious advantages in the mAR index. The design of the two FPNs improve the overall performance of the network, especially the detection accuracy of small targets.

### 3. LOITERING DETECTION BASED ON OBJECT TRACKING

This Research uses YOLOA as the basic network architecture of JDE (Jointly learns the Detector and Embedding model), and finally proposes a loitering judgment algorithm and applies it to the target tracking algorithm JYTracker to identify abnormal loitering objects in video sequences. The overall architecture diagram is shown in Figure 8.
3.1. Network Structure

The object tracking network is designed as shown in Figure 9. The network makes predictions from multiple scales. The input video frame is firstly forward-transmitted through the backbone network to obtain feature maps of four scales, that is, subsampled at 32x, 16x, 8x and 4x, respectively and then upsampled from the smallest size. Finally, three smaller-scale feature maps are retained. The network first reduces the number of channels to 256 through 1x1 convolution of the input features. Then output the dense prediction map through three branches. The dense prediction map is divided into four parts:

- Classification results of size H×W×C;
- Position regression of size H×W×4;
- Object confidence of size H×W×1;
- A dense embedding graph of size H×W×D. D is the dimension of embedding.

3.2. Data Association

In order to match multiple objects from frame to frame, data association is required. Trajectory prediction is to predict the motion state of the current target in the next frame, aiming to track the target better. The Kalman filter is a method that uses the linear system of equations. Through the system input and output observation, the optimal estimation of the system is calculated by the Kalman Filter. It can predict the coordinates and velocities of objects from a limited set of noisy object position observation sequences.

The target in the previous frame can be associated with the currently detected target through the Hungarian algorithm for the feature distance matrix and the IoU distance matrix. The Hungarian algorithm is a combinatorial optimization algorithm that can solve task assignment problems in polynomial time.

The trajectory is represented as appearance feature $e_t$ and $m_t=(x_t, y_t, h_t, v_x, v_y, v_γ, v_h)$, where $x$, $y$ represent the center coordinates of the object, $h$ represents the height of the object rectangle, $γ$ represents the aspect ratio and speed of the object rectangular box.

Each frame in the video first predicts the detection box, score and appearance embedding through JDE, then according to the detection score thresholds $ε_{\text{high}}$ and $ε_{\text{low}}$, put the bounding boxes with scores higher than $ε_{\text{high}}$ into the high-scoring bounding box set $P_{\text{high}}$, put the bounding boxes lower than $ε_{\text{low}}$ into the low-score bounding box set $P_{\text{low}}$. Kalman filter is used to predict the motion characteristics of the target, and then the target in the trajectory pool is associated with the high-scoring prediction frame, and then the matching result of the cosine similarity matrix is obtained through the Hungarian algorithm, and then keep the prediction frame and trajectory that have not been matched. Then use the IoU distance to match the unmatched track with the reserved high-scoring detection frame and low-scoring detection frame, initialize the unmatched high-scoring detection frame, discard the unmatched low-scoring detection frame and mark the unmatched track. If there is no match for 30 consecutive frames, it will be deleted, as in Algorithm 1:

**Algorithm 1**: Data Association Algorithm

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>I←Input video sequence</td>
</tr>
<tr>
<td>JDE←Federated Learning Model</td>
</tr>
<tr>
<td>KF←Kalman Filter</td>
</tr>
<tr>
<td>$ε_{\text{high}}, ε_{\text{low}}$←detection score threshold</td>
</tr>
<tr>
<td>$ε$←tracking score threshold</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td>T←Track of object in the video</td>
</tr>
</tbody>
</table>

1: Init $T$
2: for frame $f$ in I do
3: $P_f$←JDE($f$)
4: Init $P_{\text{high}}, P_{\text{low}}$
5: for $p$ in $P_f$ do
6: if $p$. score > $ε_{\text{high}}$ then
7: $P_{\text{high}}←P_{\text{high}} \cup \{d\}$
8: else if $p$. score > $ε_{\text{low}}$ then
9: $P_{\text{low}}←P_{\text{low}} \cup \{d\}$
10: end
11: end
12: for $t$ in $T$ do
13: $t$←KF($t$)
14: end
15: Associate $T$ and $P_{\text{high}}$ using Embedding and update embedding
16: $P_{\text{remain}}$←Unmatched objects in $P_{\text{high}}$
17: $T_{\text{remain}}$←Unmatched objects in $T$
18: $P_{\text{remain}}$←Unmatched objects in $P_{\text{remain}}$
19: $T_{\text{remain}}$←Unmatched objects in $T_{\text{remain}}$
20: Associate $T_{\text{remain}}$ and $P_{\text{low}}$ using IoU distance
21: $T_{\text{remain}}$←Unmatched objects in $T_{\text{remain}}$
23: \( \text{T} \leftarrow \text{T} \setminus \text{T}_{\text{remain}} \) Delete Unmatched objects in \( \text{T} \)
24: for \( \text{p} \) in \( \text{P}_{\text{remain}} \) do
25: \( \) if \( \text{p} \). score > \( \epsilon \) then
26: \( \text{T} \leftarrow \text{T} \cup \{ \text{p} \} \)
27: end
28: end
29: end

3.3. Loitering Discriminant Rules

The time threshold \( \text{T}_{\text{threshold}} \) is determined manually. Calculate the stay time of the target in the destination area as follows:
\[
\text{T}_{\text{total}} = \frac{\text{N}_{\text{frame}} \times \text{fps}}{\epsilon} \quad (6)
\]
\( \text{N}_{\text{frame}} \) represents the total number of frames in which the object appeared, and "fps" represents the number of frames per second. Usually the time threshold is set at 2-3 times the time taken to pass through the area normally:
\[
\text{T}_{\text{total}} > \text{T}_{\text{threshold}} \quad (7)
\]
The loitering behavior is determined by comparing the pixel distance of the horizontal and vertical movement of the target in the picture with the displacement threshold:
\[
\text{D}_{\text{H}} > \text{D}_{\text{threshold}_{\text{H}}} \quad (8)
\]
\[
\text{D}_{\text{W}} > \text{D}_{\text{threshold}_{\text{W}}} \quad (9)
\]
Among them, \( \text{D}_{\text{H}} \), \( \text{D}_{\text{W}} \) represent the moving distance of the target in the vertical and horizontal directions, \( \text{D}_{\text{threshold}_{\text{H}}} \), \( \text{D}_{\text{threshold}_{\text{W}}} \) indicates the moving distance threshold in the vertical and horizontal directions, which is usually set between 1.5 and 3 times the width and height of the picture. An individual moves continuously with constant changes in direction of motion, it can be considered loitering. By calculating the angle between the current movement direction and the initial movement direction, if it is greater than 90°, add one to \( \text{N}_{\text{angle}} \), and if it is greater than 160°, then perform plus two. As shown in the formula:
\[
\text{N}_{\text{angle}} > \text{N}_{\text{threshold}_{\text{angle}}} \quad (10)
\]

3.4. Loitering Discriminant Algorithm

For the trajectory set that needs to be determined, first calculate the duration of the trajectory, the moving distance of the object in the horizontal and vertical directions, and the total value of the change in the direction of motion of the trajectory, and then judge whether it exists from the thresholds of the three dimensions of time, moving distance, and angle change. Output the set of loitering trajectories. As shown in Algorithm 2:

**Algorithm 2: Loiter Discriminant Algorithm**

**Input:**
- \( \text{Tracks} \) ← Track of object in the video
- \( \text{fps} \) ← The frame rate of a video sequence
- \( \epsilon \) ← Residence time threshold
- \( \text{d}_{\text{H}}, \text{d}_{\text{W}} \) ← Image Vertical and Horizontal Distance Threshold
- \( \theta_{\text{angle}} \) ← Direction change threshold

**Output:** \( \text{L} \) ← Loitering object track in video
1: init \( \text{L} \)
2: for track in \( \text{Tracks} \) do
3: \( \text{track}. \text{Time} = \frac{\text{track}. \text{num_frame}}{\text{fps}} \)
4: \( \text{track}. \text{dis}_{\text{x}}, \text{track}. \text{dis}_{\text{y}} = \text{track}. \text{last}_{\text{x}} - \text{track}. \text{curr}_{\text{x}}, \text{track}. \text{last}_{\text{y}} - \text{track}. \text{curr}_{\text{y}} \)
5: \( \text{track}. \text{d}_{\text{H}}, \text{track}. \text{d}_{\text{W}} = |\text{track}. \text{dis}_{\text{y}}|, |\text{track}. \text{dis}_{\text{x}}| \)
6: \( \text{track}. \text{curr}_{\text{direction}} = \sqrt{\text{track}. \text{dis}_{\text{y}}^2 + \text{track}. \text{dis}_{\text{x}}^2} \)
7: if \( \cos(\text{track}. \text{curr}_{\text{direction}}, \text{track}. \text{first}_{\text{direction}}) < \cos(160^\circ) \) then
8: \( \text{track}. \text{n}_{\text{angle}} += 2 \)
9: else if \( \cos(\text{track}. \text{curr}_{\text{direction}}, \text{track}. \text{first}_{\text{direction}}) < \cos(90^\circ) \) then
10: \( \text{track}. \text{n}_{\text{angle}} += 1 \)
11: else \( \text{end} \)
12: if \( \text{track}. \text{time} > \epsilon \) then
13: \( \text{track}. \text{is}_{\text{loitering}} = \text{True} \)
14: else if \( \text{track}. \text{d}_{\text{H}} > \text{d}_{\text{H}} \| \text{track}. \text{d}_{\text{W}} > \text{d}_{\text{W}} \) then
15: \( \text{track}. \text{is}_{\text{loitering}} = \text{True} \)
16: else if \( \text{track}. \text{n}_{\text{angle}} > \theta_{\text{angle}} \) then
17: \( \text{track}. \text{is}_{\text{loitering}} = \text{True} \)
18: \( \text{end} \)
19: if \( \text{track}. \text{is}_{\text{loitering}} \) then
20: \( \text{L} \leftarrow \text{L} \cup \{ \text{track} \} \)
21: \( \text{end} \)
22: \( \text{end} \)
23: return \( \text{L} \)

3.5. Experimental results

The following figure is a visual display of JYTracker target tracking.

![Visual Display of JYTracker Target Tracking](image)

Table 5 and Table 6 below are the target tracking results of JYTracker on MOT20.

<table>
<thead>
<tr>
<th>Input Size</th>
<th>MOTA</th>
<th>IDF1</th>
<th>FP</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>512 x 928</td>
<td>86.7</td>
<td>80.1</td>
<td>15774</td>
<td>64599</td>
</tr>
<tr>
<td>608 x 1088</td>
<td>88.1</td>
<td>82.8</td>
<td>15782</td>
<td>56414</td>
</tr>
<tr>
<td>736 x 1280</td>
<td>89.2</td>
<td>84.7</td>
<td>15770</td>
<td>49597</td>
</tr>
<tr>
<td>800 x 1440</td>
<td>89.3</td>
<td>84.1</td>
<td>16481</td>
<td>48005</td>
</tr>
</tbody>
</table>
Table 6. Experimental results of JYTracker at multiple scales (continued to Table 5)

<table>
<thead>
<tr>
<th>Input Size</th>
<th>IDs</th>
<th>FPS</th>
<th>MT</th>
<th>PT</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>512 x 928</td>
<td>1476</td>
<td>20.5</td>
<td>1080</td>
<td>240</td>
<td>98</td>
</tr>
<tr>
<td>608 x 1088</td>
<td>1174</td>
<td>19.4</td>
<td>1130</td>
<td>203</td>
<td>85</td>
</tr>
<tr>
<td>736 x 1280</td>
<td>1024</td>
<td>11.5</td>
<td>1150</td>
<td>186</td>
<td>82</td>
</tr>
<tr>
<td>800 x 1440</td>
<td>1034</td>
<td>10.8</td>
<td>1172</td>
<td>162</td>
<td>84</td>
</tr>
</tbody>
</table>

From the analysis in Table 5 and Table 6, it can be seen that multi-scale training can make the target tracking algorithm adapt to the resolution of different scales, and different input sizes affect the test results. Most of the four resolutions accounted for more than 76% of traces, while most of the losses accounted for only 5%.

JYLD operates in real time, processing at least 24 frames per second with the help of TensorRT.

The following figure shows the results of JYLD’s experiments on the PETS2007 dataset S01, where the left figure shows the initial position of the loitering object, and the right figure shows the first frame in which it is judged to be a loitering object.

Figure 11. Experimental results of JYLD at PETS2007

For pedestrians who stay in place, the number of angular changes is higher due to the offset of the detection frame, so the rule is set to determine that pedestrians who meet the time threshold or both the moving distance and angular change thresholds are considered to be wandering abnormally. Finally, the time threshold is set to 300 seconds, the angle threshold is set to 4 and the distance travelled threshold is set to 1.5 times the width and height of the image.

3.6. Experimental analysis

In the case of high interference and object deformation in the practical environment, we can use the fusion of infrared and visible images for target detection, or use some data enhancement methods such as image blurring or deformation to train the model, so as to improve the robustness of the algorithm.

JYLD performs anomaly loitering detection through trajectory analysis, which is independent of the behavior of the target, which may affect the accuracy of the target detection and tracking. In actual use, the model can be trained according to specific scenarios and constantly updated.

In future research, JYLD can be popularized by means of a series of compression models such as knowledge distillation, making it smaller in size and able to be deployed on equipment with limited resources. The accuracy of loitering detection can also be further improved by training large models.

4. CONCLUSION

This paper proposes an improved object detection algorithm YOLOA for YOLOX. In response to the problem that YOLOX is not accurate in detecting small targets, YOLOA proposes an improved method in terms of attention mechanism, backbone network, FPN, detection head and loss function. Through the test results on the COCO dataset, the mAP and mAR of the YOLOA-S sub-model surpass YOLOX-S, especially improving the detection of small objects, which makes YOLOA competent for the detection of complex scenes.

Based on YOLOA, this study further proposes a loitering detection algorithm based on object tracking. JYLD, which includes the tracking algorithm JYTracker and efficient loitering determination rules. JYLD is able to quickly identify suspicious targets in complex scenes and deal with interference and target deformation in the tracking process, with a small number of parameters and high accuracy and robustness.

ACKNOWLEDGMENT

This research is supported by National Natural Science Foundation of China Nos. 61972301 and 61672403, Key Research and Development Program of Shaanxi No. 2023-YBGY-229 and Xi’an Science and Technology Project No. 22GXFW0025.

REFERENCES


[2] Tiemei Huang, Qing Han, Weidong Min, Xiangpeng Li, Yunjun Yu, Yu Zhang. Loitering Detection Based on Pedestrian Activity Area Classification[J]. Applied Sciences, 2019, 9(9).


