# Multi-video Supervisory Target Tracking Improved by Interactive On-line Learning

Chang Wang[1] Yuting Zhu Xupeng Wen Yifeng Niu[1] Lizhen Wu
College of Intelligent Systems, National University of Defense Technology
De Ya Lu 109, Changsha, Hu'nan, China, 410073
*[1]wangchang07@nudt.edu.cn niuyifeng@nudt.edu.cn*

**It is difficult for a human operator to monitor multiple video streams in real-time surveillance tasks. A variety of machine learning algorithms have been proposed for target tracking, but human intervention is still necessary because the algorithms have limitations to detect and recover from tracking failures. In this paper, we propose a supervisory target tracking method for multi-video surveillance tasks based on hTLD (human-in-the-loop Tracking-Learning-Detection). The human operator can monitor both the original video streams and the calculated confidence values of the tracking results. Human intervention allows to update the training data for TLD whenever a tracking failure is observed, by manually drawing a bounding box or using an eye-tracker to relocate the target. Meanwhile, an interactive on-line learning algorithm is used to learn the relations between the confidence values and the tracking results. In this way, the confidence values help avoid unnecessary intervention caused by false alarms of tracking failures. Experimental results have demonstrated that the proposed method can reduce the task completion time.**

**Interactive learning, eye-tracker, target tracking, TLD.**

## 1. INTRODUCTION

Monitoring multiple video streams by a single human operator is a typical surveillance task for the Control Center of public places such as supermarkets, airports or train stations. Watching these videos can be tedious for the human operator after several hours of work. Nevertheless, the surveillance task can be more challenging in dynamic and uncertain environments, e.g., in the scenarios of using multiple UAVs to search for survivors after disasters, patrol coastal borders, or locate and attack terrorists.

In the literature, a variety of surveillance systems have been developed for monitoring people and vehicles (Wren et al., 1996; Olson et al.,1997; Collins et al., 2000; Haritaoglu et al., 2002; Matei et al., 2011; Unzueta et al., 2012). However, these systems have made strong assumptions of the task environments. For example, the Pfinder system (Wren et al., 1996) assumed that there was only a single person in an upright standing posture. The TI system (Olson et al.,1997) could not handle the motions of background objects, or the cameras and the veiwing angles were fixed (Matei et al., 2011; Unzueta et al., 2012).

Various machine learning algorithms have been chosen to detect and track targets of interest, e.g., using deep learning (Wang et al., 2017; Wei et al, 2017), Support Vector Machines (SVMs) (Chen et al., 2013), or Dynamic Bayesian Networks (DBNs) (Cheng et al.,2012). However, these algorithms may not work well in real-time tasks where on-line learning is needed to solve the target's appearance changing problem caused by motion blur or lighting change. In addition, these algorithms typically are data hungry that require a large data set for model training before the learned models can be used for task execution, e.g., the deep learning approach.

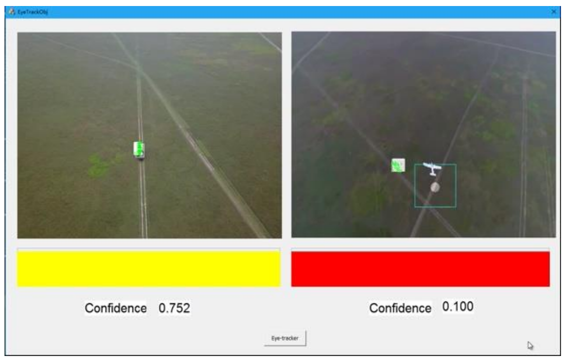

***Figure 1**: An illustration of multi-video surveillance task supported by TLD. The videos were taken by on-board cameras of UAVs. A ground vehicle and an aerial vehicle were being tracked, respectively. In each video stream, the rectangular with feature points (marked in green) indicated the location at which TLD believed the target was. In the right video, the circle (marked in white) indicated the gazing points obtained from the eye-tracker.*

In our previous work, hTLD (human-in-the-loop TLD) (Zhu et al., 2018) includes human into the loop of the original TLD algorithm, which supports on-line learning of the new appearances of the target of interest. However, monitoring several video streams simultaneously remains a challenge for a human operator assisted by hTLD. We illustrate the limitations in Figure1. First, the human has to initialize the TLD tracker by selecting the interested target in the video, typically drawing a bounding box around it. Then, hTLD uses the selected region to provide samples for training the TLD learner. This process usually takes several seconds before the TLD tracker starts working. As a result, the human operator could be too busy to take care of all the video streams, so that some of the tracking failures cannot be handled in time. Therefore, a more efficient way of human-computer interaction has to be designed to meet the real-time performance requirement of the surveillance task. Second, it is not reliable of TLD to decide by itself whether a tracking failure has happened or not. Therefore, it is necessary to find a reliable way of calculating the confidence values to assist the human operator to interpret the tracking result of each video stream.

Compared with our previous work (Zhu et al., 2018), the main contributions of this paper are as follows:

- We have proposed a novel framework for solving the multi-video target tracking task.
- We have proposed an on-line learning algorithm to discover the relations between the confidence values and tracking results that can assist the human operator for decision making.
- We have sped up the target tracking task by using eye-tracker to provide training data for TLD compared with the old way of using mouse to draw a bounding box.

This paper is organized as follows. Section 2 briefly introduces the framework. Section 3 describes the the framework in details, followed by experiments in Section 4. Finally, section 5 concludes the paper and outlines our plans for future work.

## 2. THE SUPERVISORY TARGET TRACKING FRAMEWORK

The proposed framework is shown in Figure 2. We assume that the human operator simultaneously performs several surveillance tasks. Due to the limitation of the paper layout, we only illustrate task *N* in the framework to explain how the human operator interacts with the original TLD module and the confidence learning module. Each other task is exactly the same with the case of task *N*.

Refer to our previous work for details of hTLD (Zhu et al., 2018). However, the human operator is fully responsible for the judgement of a tracking failure in

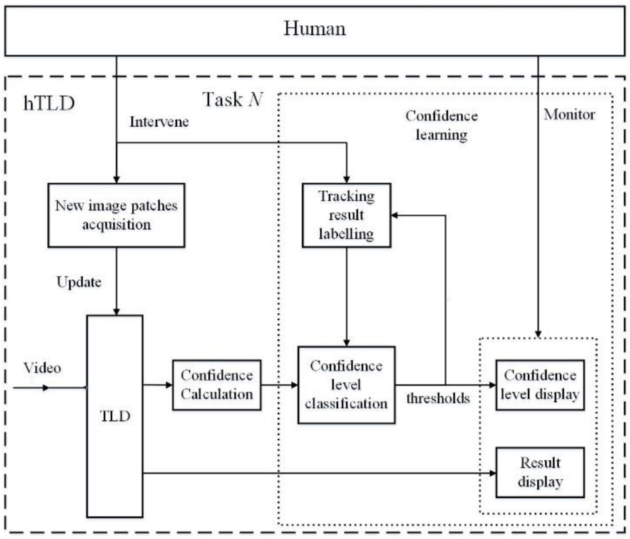

***Figure 2:*** *The framework of multi-video supervisory target tracking supported by interactive on-line learning.*

hTLD. This may work for a single task, but is not user-friendly in multi-task scenarios. Because the human operator can easily get bored or fatigue after long time of gazing at the targets (usually small) in the videos. In this paper, we use a confidence value to assist the human operator for deciding whether a tracking failure happens or not. However, the calculation of the confidence value is based on the distribution of visual features extracted by TLD, which can be inconsistent or unreliable in changing environments. Therefore, we add a confidence learning module to learn the relations between the confidence value and the tracking result. Specifically, we choose Support Vector Machines (SVMs) as the supervised learning algorithm to classify the confidence values into three levels, i.e., *failure*, *success* or *unsure*. We note that *unsure* means a tracking drift. We have added a coloured bar in red, green and yellow (see Figure 1) corresponding to the three classes, respectively. Whenever a tracking failure has happened, the human operator selects the target of interest by drawing a bounding box or using the eye-tracker, and a sample is automatically generated by labelling the current confidence value as *failure*. If the human does not intervene, the current confidence value is automatically labelled as *success* or *unsure*, depending on the current thresholds given by the SVMs.

## 3. INTERACTIVE LEARNING

### 3.1 Use eye-tracker to select target for learning

Compared with hTLD, we use an eye-tracker to select the target of interest in order to speed up the tracking process. We assume the initial target size is $W \times H$ pixels, and it does not change much in the task. The eye-tracker captures the human's gazing points and records the corresponding coordinates with a specific frequency. First, we sample a set of gazing points for a fixed period of time t, denoted as $P = \{p_1, \ldots, p_n\}$. Then, we calculate the centroid of $P$ to estimate the target center. Considering that there exists errors between the estimated location and the true location of the target (see Figure 3), we use a bigger bounding box with the size of $nW \times nH$ pixels to cover the target, where $n>1$. As the size of the bounding box influences the tracking result, the bounding box should be as accurate as possible to exclude irrelevant features (Zhu et al., 2018). Based on the bounding box obtained from the eye-tracker, we propose an adaptive algorithm to separate the target from the background. We use the *k*-means clustering algorithm (Hartigan, 1979). We choose the maximum and minimum pixel values as the initial centroids of the target cluster and the background cluster, respectively. Then, we classify all pixels into the two clusters according to the distances between each pixel's value and the two cluster's centroids. The clustering process is iterative until converged.

After the optimal bounding box is obtained, we can use it to update TLD. We use the bounding box to initialize the tracker, and we use it to generate new training samples that include both positive samples and negative samples using PN learning. Then, the TLD learner trains the classifier to update the TLD detector. As a result, TLD tracker is updated and adapted to the target accordingly (see Figure 3).We summarize the above process in Algorithm 1.

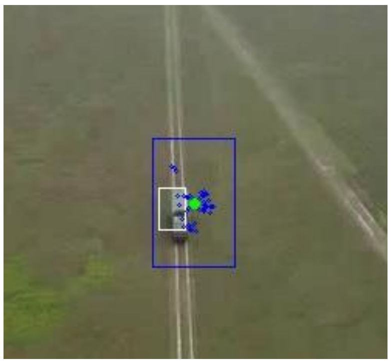

***Figure 3**: An illustration of deciding the bounding box using the gazing points (marked in blue) obtained from the eye-tracker. The calculated centroid is marked in green, the blue box is the bigger one, and the white box is the resulted bounding box that represents the target.*

## 3.2 Interactive learning of confidence levels

The original confidence values calculated by TLD is not reliable for judging whether a tracking failure actually happens or not. For example, in case that the confidence value drops low and alarms the human operator of a tracking failure, actually it can be only a tracking drift and not a failure yet. In this case, TLD may recover by itself and needs no human intervention. However, this kind of false alarm will increase the workload of the operator. In this paper, we calculate the confidence values as

***Algorithm 1:** Learning in hTLD using eye-tracker*

```
if a tracking failure is observed then
   human initiates the sampling of gazing points
   $P=\{p_1,\ldots,p_n\}$ for a fixed period of time $t$;
   Calculate the centroid $Cp$ of $P$ and generate a
   bounding box eBox with the size of $nW \times nH$ pixels that
   covers $T$, where $n>1$;
end if
   calculate the max value $a$ and min value $b$ of the
   image, then take them as the initial cluster centroids of
   the target and the background, respectively;
   classify all points into the two clusters according to
   each point's distance to $a$ and $b$, then compute centroid1
   and centroid2;
while($a$ is not equal to centroid1 OR $b$ is not equal to
centroid2)
   $a \leftarrow$ centroid1, $b \leftarrow$ centroid2;
   classify again as described above;
end while
   get threshold: threshold  max(point$\in$cluster1);
   separate the object from background according to
   threshold;
   adjust the bounding box to an appropriate size.
Update TLD.
```

follows. We assume that the feature points of the target should be concentrated with a small variance. We calculate the tracked points' variance to evaluate the confidence, and we calculate the ratio of the number of tracked points to the number of all feature points. The higher the ratio, the more confident the tracking result is. We propose an interactive learning method to learn the relations between the confidence values and the alarm levels. We collect training samples and label them as *failure*, *success* or *unsure* (see section 2). Accordingly, we train SVMs to classify the confidence values into three classes. As a result, we obtain two support vectors, i.e., two thresholds $thr_1$ and $thr_2$ to classify the confidence values into three levels. Then, we use three coloured bars (i.e., green, yellow, red) to represent the three levels, respectively. We summarize the above process in Algorithm 2.

***Algorithm 2:** Learning confidence levels using SVMs*

```
Calculate the current confidence value Cf.
1 : Obtain labelled data.
 if human intervenes then
     label Cf as failure, and add (Cf, failure) to D;
 end if
if human does not intervene then
     if $Cf > thr_2$ then
          add (Cf, success) to D;
     else if $thr_1 < Cf < thr_2$ then
          add (Cf, unsure) to D;.
     else
          add (Cf, failure) to D;
     end if
end if
2: Train two SVM classifiers SVM1 and SVM2
Train classifier1 to classify success and unsure;
Train classifier2 to classify unsure and fail;
$thr_1 \leftarrow$ classifier1's support vector;
$thr_2 \leftarrow$ classifier2's support vector.
```

## 4. EXPERIMENTS

We used two videos taken by on-board cameras of UAVs, tracking a ground vehicle and an aerial vehicle, respectively (see Figure 1). The video of task 1 lasted 78 seconds and had 2340 frames. The video of task 2 lasted 13 seconds and had 390 frames. Both videos had the frame size of 640x480 pixels. One of the authors did the experiments using both the mouse and the eye-tracker.

### 4.1 Results of respond time

We compare the results of the respond time between using the eye-tracker and the traditional way of using the mouse (see Table 1). When using the eye-tracker, we set *n* = 1, 2, 3 as explained in Algorithm 1. The results indicate that using eye-tracker is faster than using the mouse in all cases. The respond time of the case *n* = 2 is only half of the case *n* = 1, but the respond time of *n* = 3 is almost the same with the case of *n* = 2. The reason is that if *n* is too small, i.e., *n* = 1, the generated bounding box may be too small to cover the target, therefore it requires the human operator to gaze for a longer time for a successful tracking. By using a larger bounding box, it is easier to select the target. However, the bounding box does not have to be too big, because a bigger bounding box has a higher calculation cost. Thereafter, we choose *n* = 3.

***Table 1:*** *Comparison of the respond time (s)*

| | | Task1 | Task2 |
|---|---|---|---|
| Mouse | | 3 | 2.5 |
| Eye-tracker | *n* = 1 | 1.5 | 1.9 |
| | *n* = 2 | 0.8 | 0.9 |
| | *n* = 3 | 0.8 | 0.8 |

### 4.2 Results of task completion

We compare the tracking accuracy, time spent and the numbers of intervention in Table 2 and 3.

***Table 1:*** *Comparison of tracking accuracy and time*

| | Task1 | | Task2 | |
|---|---|---|---|---|
| | Tracked frames | Total time(s) | Tracked frames | Total time(s) |
| Mouse | 1690 | 252 | 257 | 73 |
| Eye-tracker (*n*=3) | 1831 | 214 | 331 | 69 |

***Table 2:*** *Numbers of human intervention*

| | Task1 | Task2 |
|---|---|---|
| Mouse | 13 | 8 |
| Eye-tracker (*n*=3) | 19 | 7 |

In both tasks, using the eye-tracker saves time as well as increases the number of tracked frames. In task 2, the numbers of human intervention are almost the same. However, in task 1, the number of human intervention using the eye-tracker is higher than using the mouse, because some of the interventions failed to correctly select the target.

### 4.3 Results of confidence level learning

The results are given in Table 4. We have a record of the numbers of missing alarms and false alarms in both tasks using or not using SVMs. The results indicate that using the SVMs reduces the number of missing alarms in task 1, but does not make a much difference in general. It might be caused by the biased collection of training samples, in other words, only the tracking failures were correctly labelled whenever the human intervened, and the *success* and *unsure* data were not distinguished clearly. Nevertheless, using the SVMs makes the thresholds interpretable and easier for the human operator to make decisions.

***Table 4:*** *Comparison of the numbers of alarms*

| | Task1 | | Task2 | |
|---|---|---|---|---|
| | missing alarms | false alarms | missing alarms | false alarms |
| SVMs | 11 | 27 | 9 | 7 |
| No SVMs | 15 | 25 | 9 | 5 |

## 5. CONCLUSION

In this paper, we have proposed a novel framework for multi-task supervisory target tracking based on TLD. With human in the loop of target tracking, training data can be obtained automatically for TLD learning as well as for confidence level learning. An eye-tracker has been used to approximately locate the target of interest, and a clustering algorithm has been used to decide the bounding box for the target as accurate as possible. In addition, we have used a supervised learning algorithm to learn the meanings of the confidence values so that coloured bars can alarm the human operator to correct tracking failures as soon as possible. Experiment results have shown that the TLD tracker responds faster by using eye-tracker than using mouse, and the users prefer to use the eye-tracker and the coloured bars of confidence levels.

In the future, we will further improve the supervisory system by optimizing the tracking algorithms as well as designing a more user-friendly interface. In addition, we will evaluate the system with more participants. We assume that the coloured bars can provide an option that relieves the human from watching all the videos which requires higher cognitive workload. We would like to know how much workload can be reduced using the proposed system, as well as whether the users prefer to monitor the videos with bounding boxes or the coloured bars representing the confidence levels. We will also compare the situation awareness of the multi-video tracking tasks between using the eye-tracker and using the mouse.

## 6. REFERENCES

Wang, W., Tian, B., Liu, Y., Liu, L., Li, J. (2017) Study on the electrical devices detection in UAV images based on region based convolutional neural networks. Journal of Geo-information Science, 19(2):256–263.

Wei, Y., Quan, J., Hou, Y. (2017) Aerial Image Location of Unmanned Aerial Vehicle Based on YOLO v2. Laser&Optoelectronics Progress, (11):95–104.

Chen, L., Jiang, Z., Yang, J., Ma, Y. (2013) A coarse-to-fine approach for vehicles detection from aerial images. International Conference on Computer Vision in Remote Sensing, Vol.21, pp.221-225.

Cheng, H., Weng, C., Chen, Y. (2012) Vehicle Detection in Aerial Surveillance Using Dynamic Bayesian Networks. IEEE Transactions on Image Processing, 21(4): 2152–2159.

Collins, R. T., Lipton, A. J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., et al. (2000) A system for video surveillance and monitoring, 59 (5),: 329–337.

Unzueta, L., Nieto, M., Cortes, A., Barandiaran, J., Otaegui, O., & Sanchez, P. (2012) Adaptive multicue background subtraction for robust vehicle counting and classification. IEEE Transactions on Intelligent Transportation Systems, 13(2), 527-540.

Khatoonabadi, S. H., Bajic, I. V. (2013) Video Object Tracking in the Compressed Domain Using Spatio-Temporal Markov Random Fields. IEEE Press.

Zhu, Y., Wang, C., Niu, Y., Wu, L. (2018) hTLD： A Human-in-the-loop Target Detection and Tracking Method for UAV. IEEE/CSAA Guidance, Navigation and Control Conference. To appear.

Jian, L., Yin, D., Shen, L., Niu, Y., Zhu, L. (2017) Human machine collaborative support scheduling system of intelligence information from multiple unmanned aerial vehicles based on eye tracker. Journal of Shanghai Jiaotong University, 22.3: 322–328.

Lucas, B. D., Kanade, T. (1981) An iterative image registration technique with an application to stereo vision. International Joint Conference on Artificial Intelligence, Vol.2, pp.674-679

Kalal, Z., Mikolajczyk, K., Matas, J. (2012) Tracking-learning-detection. IEEE transactions on pattern analysis and machine intelligence, 34(7): 1409–1422.

Kalal, Z., Matas, J. Mikolajczyk, K. (2010) PN learning: Bootstrapping binary classifiers by structural constraints. Computer Vision and Pattern Recognition, 49-56. Vol.238, pp.49–56.

Hartigan, J. (1979) A K-Means Clustering Algorithm. Appl Stat, 28(1):100–108.