# Network Based Cooperative Surveillance System

Shunrong Shen
Department of Electrical Engineering
Stanford University
Email: eetsrshen@stanford.edu

Tongda Zhang
Department of Electrical Engineering
Stanford University
Email: tdzhang@stanford.edu

Xiaoxiong Lu
Department of Electrical Engineering
Stanford University
Email: lxx@stanford.edu

*Abstract*—**In this project, we worked on a network based cooperative surveillance system. It can be used to track either humans or monitor still objects for anti-theft purpose. We designed multiple ways for user to select the tracking target. The tracking algorithm is based on a Tracking-Learning-Detection pipeline. Experiment results show fast and accurate tracking performance of the system. The potential of cooperative surveillance over the internet is also demonstrated.**

***Keywords:*** *learning, human detection, tracking, histogram of oriented gradient (HoG) Introduction*

## I. Introduction

Nowadays, surveillance cameras are used everywhere in daily life. Video processing techniques for tracking and detection can be applied to monitor the trace of concerned targets. The targets could be either an object or a human like a baby. In the latter case, the surveillance system can help parents keep an eye on their children, and desirably issue a warning when anything abnormal happens, e.g. the children leave the playground and walk into the middle of a road. Other examples may involve anti-theft of vehicles and bicycles. From what we have learnt about features detection, we can use surveillance cameras to replace human eyes and constantly monitoring interested targets.

The core of the system is a long term tracking system. It consists of three major components: detection, learning and tracking. The detection components is designed to detect the presence of the selected target using a classifier updated by the learning components frame by frame. The tracking component tries to estimate the target in the surroundings of its previously known location and it is also updated by the learning component. The learning algorithm is called P-N learning as proposed in [1]. It generates false negative and false positive examples from the input images and previous detection and tracking results. These two independent training sets are used to train a classifier using supervised bootstrap algorithm. Fig. i1 shows the operating flowchart of our system.

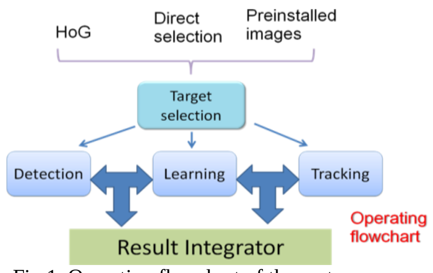


Fig 1. Operating flowchart of the system

To start the whole process, the system requires a sample image to initialize the classifier. As shown in the figure, the user can decide how to initialize the target object, either using HoG to detect human targets automatically, or using preinstalled target images. If these two approaches fails in some circumstances, users can also directly select the target by drawing a rectangular box on the first frame.

In addition to the essential tracking function, our system is enabled to expand over the internet. As shown in Fig. 2, our system can be configured to work in a Client-Server mode. Each client and server is connected to an individual camera, and can implement the tracking algorithm locally. The tracking information on the client side will be gathered by the Server. For different purpose of tracking, the server will decide when to send alarm based on the received tracking status from the clients.

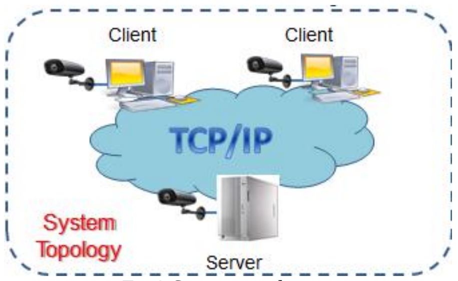


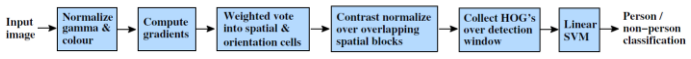
Fig 2 System topology

In the remaining sections, each of the aforementioned components will be discussed in more details. Following that, experiment results are presented with discussions. The whole report will be ended by a brief conclusion and proposal of future work.

## II. System Initialization

### A. HoG

Our program integrates an automatic human detection function using Histogram of Oriented Gradients (HoG) algorithm to facilitate the selection of human targets. The basic idea of HoG is that the features of an object can be characterized by the distribution of local intensity gradients or edge directions [1]. As the intensity gradient distribution is locally calculated for each small spatial tile of the image window, the extracted feature is independent of the background and actual location in the image. Fig 3 shows a typical pipeline of HoG implementation. The input image is first divided into many tiles and each tile is subdivided into smaller image cells. A sliding window goes through every tile with overlapping in cells. The normalization processes in the pipeline are intended to enhance the robustness of HoG against different Gamma , color and illumination distortions. The gradients can be computed by applying a differentiator mask, i.e. [-1, 0, 1]. The calculated gradients is then binned into an orientation histogram which decides the gradient orientation in the cell. After a sequence of aforementioned processes, the collected information is fed into support vector machine (SVM) for training or classification if it is used for prediction. To fully exploit the capability of HoG, parameters such as the size of sliding window in each step of the pipeline may need to be specifically tuned for particular applications. It is also reported in [1] that HoG performs best if the gradients are derived from a image with finest possible details. Hence in our application, we do not apply any Gaussian kernel filtering to blur the image before HoG detection.


Input image
Normalize gamma & colour
Compute gradients
Weighted vote into spatial & orientation cells
Contrast normalize over overlapping spatial blocks
Collect HOG's over detection window
Linear SVM
Person / non–person classification
Fig.3. Pipeline of typical HoG implementation [1].

### B. Pre-installation of images

This is another pre-tracking part of the algorithm. Images of expected targets are installed in the system beforehand, and hence can provide the option for the user to choose target directly from the database. User can choose several common situations as data categories. In order to make the data in this part compatible with the tracking algorithm, the target is saved as a whole picture with a bounding box of the target object in that picture. For a more efficient realization of this part, the image of the target will be processed and directly put into the tracking dataset, where the content within the bounding box will be resized to a standardized pattern and used as a positive template for the target. The parts around the bounding box, in other words, the background will be regard as the negative templates.

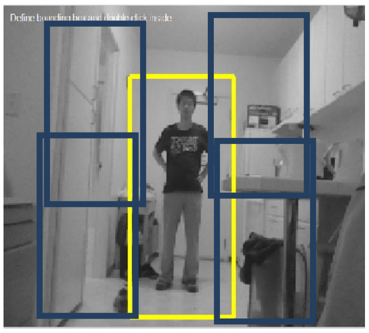


Fig. 4

For example, in Fig. 4, the yellow bounding box of the preinstalled images are considered as expected object templates. The blue bounding boxes around the yellow one are the templates that is regarded as negative image patches.

### C. Object tracking and detection

This part is mainly based on TLD[2]. TLD(TRACKING-LEARNING-DETECTION) mainly consists of three components: Tracking, Detection and Learning. Tracking is used to estimate the position of the object position based on frame-to-frame motion of the objective, assuming the motion is continuous and the object is visible all the time. When the object is missing in a frame, the tracking would fail. By using the additional detection information, the tracking of an object can be recovered once the missing object returns to the camera view. As for the learning, it is used to make the system adapted to a new background and a new target feature set so that the whole system can work stable for a long-term tracking task.

The output (a bounding box or object missing notation) is a combination of the results from both the tracking part and the detection part. When the tracking part is working well, the bounding box is generated and the detection part will ensure the output result. When the tracking part is missing target, the missing notation will be generated as a output, at the same time, the detection part will keep working to detect if the missing object is back to the sight.

## III. Operating Principle of the System

### A. Tracking

The tracking part is mainly based on Median-Flow tracker[6], adding negative patches estimation parts. The target is described by the bounding box within the image mentioned earlier. The tracking parts estimates the motion between consecutive frames and determines the target motion by searching for its possible position around its last known positions. The pyramidal Lucas-Kanade tracker[7] is used to realize that. As for the negative patches estimation, we know that for a tracker, there are two possible situations to miss targets. One is due to the sudden disappearance of targets (be sheltered off, or move out of the image). Another cause could be ascribed to the fast motion of the targets. In both cases, the trajectory of the continuous motion breaks, and the wrong estimation of target position is considered as negative template patches that the classifier can use for learning. In short, after a tracking failure, negative learning image patches will be generated, and the system will give a target-missing notation. Units

### B. *Detection*

To ensure the robustness of the detection, the detector using a template matching method should cover samples on a wide variety of scaled, shifted and rotated versions of the original frames. However, this approach consumes too much time to be practical in real-time application.

To save time of detection, several approaches have been taken. First of all, instead of a continuously scanning window, the sliding window will jump to the next patch which is several pixels away from the former position but will be still close enough. This will save tremendous time for template matching, as we have less candidate patches to be compared with our templates. The next bottleneck is the matching part, which is also time-consuming. The solution we adopted is to pre-filters out a lot of the candidates based on variance comparison. This method is simple and efficient. We compare the gray-value variance $E(p-E(p))^2$ between the candidates and the matching patches. If the difference is big enough, the candidate is not going to be compared in the next round. After the variance filtering, a second stage filtering using pixel comparison cuts more candidates out. The idea is to generate a certain length pixel position sequence of the patches and then do the pixel-by-pixel comparison of the gray value of the sequence. Using several position sequence comparison, we will have enough result to conduct the second filtering. If the comparison result is not good, we will cut it out of the candidates set. By means of pre-filtering, we significantly reduce the number of candidate patches to be compared with the templates. In the next step, we can use nearest neighbor method to do the final decision. This is a time-consuming part. But fortunately, the major part of the candidate are already been filter out, only a small number of patches are involved in the nearest neighbor calculation, which is achievable in real-time situation.

In short, the detection component uses variance difference comparison, sample pixel comparison and nearest neighbor method to search for the most possible target position.

### C. *Learning*

The learning component is intended to initialize the whole classifier at the beginning of the tracking with the first frame and bounding box (or the HoG detection result and the preinstalled the target images). During runtime, the learning component is used to update the classifier to make it more robust against the change of the tracking conditions, including the changes in background, illumination and so on.

During the initialization stage, only one positive image patch needs to be generated. The negative images are easy to generate, using the content from the background which is outside the bounding box will give us enough initial negative training data. As illustrated in Fig. 4. However, single positive patch is far from enough to train a classifier. More positive examples should be derived from this only frame. The method is to use geometric transformation of the original bounding box. The combination of position shifting, scale changing and angle rotation generates hundreds of new positive patches to be used for training.

After the initialization, the whole surveillance system starts running and the role of learning component switches to updating the system classifier. The way how learning part achieves this has already been briefly discussed in both the tracking part and detection sections. The dataset will keep two categories of errors that current classifier is making. One is false positive that the system is tracking at a object that is not a target, another is false negative that the system claim the target is missing while the object is actually in sight. To compensate both errors, the learning algorithm combines the result from both the tracking part and the detection part. For example, it is assumed that there is only one object being tracked. Hence any candidates that exceed the one with highest score generated by the detection part will be considered as a false positive. Also, for the conflict results from tracking part and detection part, we are also assured there exists a false positive. As for false negative, when the tracking part misses the target and the detection part find a really high score patch, we can assume there may be a false negative. With the new dataset of false negative and false positive, the system classifier can be retrained for better performance.

In short, the learning component will generate many distorted image for the training unit to make a good starting tracking. During the runtime, the learning part will constantly collect possible false positive errors and false negative errors, adding them to the training dataset to improve the tracking ability of the system.

## IV. Operating principle of the system

### A. *Tracking*

The tracking part is mainly based on Median-Flow tracker[6], adding negative patches estimation parts. The target is described by the bounding box within the image mentioned earlier. The tracking parts estimates the motion between consecutive frames and determines the target motion by searching for its possible position around its last known positions. The pyramidal Lucas-Kanade tracker[7] is used to realize that. As for the negative patches estimation, we know that for a tracker, there are two possible situations to miss targets. One is due to the sudden disappearance of targets (be sheltered off, or move out of the image). Another cause could be ascribed to the fast motion of the targets. In both cases, the trajectory of the continuous motion breaks, and the wrong estimation of target position is considered as negative template patches that the classifier can use for learning. In short, after a tracking failure, negative learning image patches will be generated, and the system will give a target-missing notation.

### *B. Detection*

To ensure the robustness of the detection, the detector using a template matching method should cover samples on a wide variety of scaled, shifted and rotated versions of the original frames. However, this approach consumes too much time to be practical in real-time application.

To save time of detection, several approaches have been taken. First of all, instead of a continuously scanning window, the sliding window will jump to the next patch which is several pixels away from the former position but will be still close enough. This will save tremendous time for template matching, as we have less candidate patches to be compared with our templates. The next bottleneck is the matching part, which is also time-consuming. The solution we adopted is to pre-filters out a lot of the candidates based on variance comparison. This method is simple and efficient. We compare the gray-value variance $E(p-E(p))^2$ between the candidates and the matching patches. If the difference is big enough, the candidate is not going to be compared in the next round. After the variance filtering, a second stage filtering using pixel comparison cuts more candidates out. The idea is to generate a certain length pixel position sequence of the patches and then do the pixel-by-pixel comparison of the gray value of the sequence. Using several position sequence comparison, we will have enough result to conduct the second filtering. If the comparison result is not good, we will cut it out of the candidates set. By means of pre-filtering, we significantly reduce the number of candidate patches to be compared with the templates. In the next step, we can use nearest neighbor method to do the final decision. This is a time-consuming part. But fortunately, the major part of the candidate are already been filter out, only a small number of patches are involved in the nearest neighbor calculation, which is achievable in real-time situation.

In short, the detection component uses variance difference comparison, sample pixel comparison and nearest neighbor method to search for the most possible target position.

### *C. Learning*

The learning component is intended to initialize the whole classifier at the beginning of the tracking with the first frame and bounding box (or the HoG detection result and the preinstalled the target images). During runtime, the learning component is used to update the classifier to make it more robust against the change of the tracking conditions, including the changes in background, illumination and so on.

During the initialization stage, only one positive image patch needs to be generated. The negative images are easy to generate, using the content from the background which is outside the bounding box will give us enough initial negative training data. As illustrated in Fig. 4. However, single positive patch is far from enough to train a classifier. More positive examples should be derived from this only frame. The method is to use geometric transformation of the original bounding box. The combination of position shifting, scale changing and angle rotation generates hundreds of new positive patches to be used for training.

After the initialization, the whole surveillance system starts running and the role of learning component switches to updating the system classifier. The way how learning part achieves this has already been briefly discussed in both the tracking part and detection sections. The dataset will keep two categories of errors that current classifier is making. One is false positive that the system is tracking at a object that is not a target, another is false negative that the system claim the target is missing while the object is actually in sight. To compensate both errors, the learning algorithm combines the result from both the tracking part and the detection part. For example, it is assumed that there is only one object being tracked. Hence any candidates that exceed the one with highest score generated by the detection part will be considered as a false positive. Also, for the conflict results from tracking part and detection part, we are also assured there exists a false positive. As for false negative, when the tracking part misses the target and the detection part find a really high score patch, we can assume there may be a false negative. With the new dataset of false negative and false positive, the system classifier can be retrained for better performance.

In short, the learning component will generate many distorted image for the training unit to make a good starting tracking. During the runtime, the learning part will constantly collect possible false positive errors and false negative errors, adding them to the training dataset to improve the tracking ability of the system.

## V. COMMUNICATION

Could based network services is an emerging technology nowadays and it can be easily imaged that our surveillance system can provide much more flexible services, if it can handle information from multiple cameras over the network. In our project, we demonstrated the potential of cooperative surveillance using Matlab based TCP/IP communication. More than one client can be connected to the server over the internet and each of them is connected to a camera. Each node in the system is able to track the target locally and communicates the tracking information with the server. The system can operate in a cooperative mode so that every node focus on a single target, or the cameras can monitor individual target and report status back to the server. Since the computation is distributed to the local nodes, only few information need to be transmitted across the network. In our project, we configure two computers in cooperative mode. Once the client loses contact on the target, it will send an warning to the server and display a warning message on its own screen. The server will not sound an alarm upon the receipt of warning message until it also loses contact with the target. Similarly, if the server loses contact, it will not immediately report a loss until the status of the clients are checked out. More details of the demonstration is included in the following section.

We can also setup Android phone as the wireless surveillance camera. The wireless camera can send the real time video to the server for tracking and detection. You can put this wireless camera anywhere and also can disguise it secretly. However, we do not have time to finish this part. Our Android phone can only send back real time video to our server.

## VI. Experiment Result and Discussion

The HoG function used in our project is based on openCV library. One disadvantage of HoG is that some of its parameters such as the window size and the hit threshold need to be carefully adjusted for best performance in particular situations. It also requires the presence of whole human body. If the camera captures only a partial body of a person, HoG detection will most likely fail. This could possibly be explained by the fact that the descriptor used in openCV library is trained on standard pedestrian photos which always show complete body. It is also found that the accuracy of HoG detection heavily depends on the contrast of the clothing and the background. If the intensities of the two are too close, HoG detection may also fail.

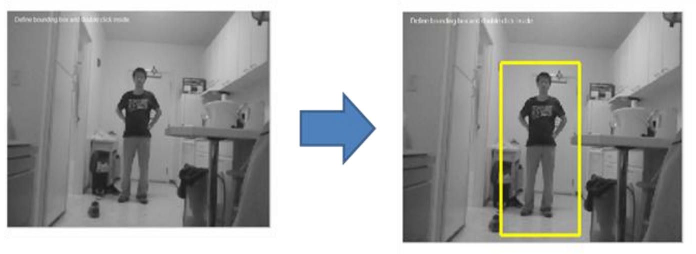


Fig 5. Demonstration of HoG detection

To demonstrate our network based cooperative surveillance system, we used two laptops, each connected to a camera, to monitor a person walking on a playground. One of the laptop serves as a client and the other one is set as server. The two laptops communicate with each other using TCP/IP protocol. The core function of TLD implementation is based on openTLD [8] and is modified for better performance in our case. As shown in Fig 6(a), once the system is successfully initialized, it can easily track a walking person. If the client camera loses contact on the target, it will display a red warning on the screen and send a warning message to the server, as illustrated in Fig. 6(b). Since the camera on the server side still had contact with the person, it suppressed the global alarm until the target also walked out of the view of server camera in Fig. 6(c).

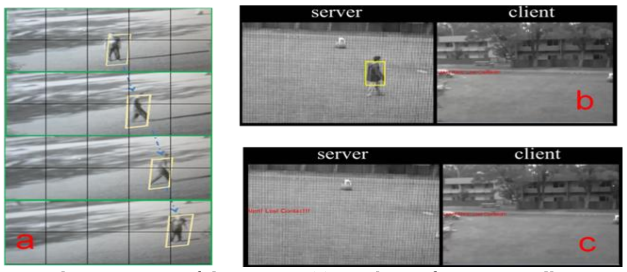


Fig 6 Tracking outcome of the system. (a) Tracking of a person walking on the playground. (b) Client camera loses contact on the target. (c) Both camera loses contact with the target and an alarm is set on the server screen.

During the experiment, we observed some potential problems of TLD algorithms. As the image is processed in grayscale, the accuracy of detection and tracking depends on the contrast between the background and target image. For instance, if the person in the camera wears a cloth with shallow color, the TLD algorithm can easily mix it with the lawn in the background. Moreover, the presence of other object that resembles the curve of the human body edge such as small trees can also cause tracking errors.

## VII. Conclusion and Future Work

In this report, we introduce the functions of our network based cooperative surveillance system. The underlying operating principles are also discussed. Experiment results of both positive and negative outcomes are presented with discussion. We also demonstrate the usage of internet to enhance the capability and flexibility of the system in a people tracking context.

The system can be further improved in multiple ways. The HoG function could possibly be enhanced by using color images for human detection. The classifier of HoG could also be trained with a broader range of cases to cover more possible presences of people. In the case of tracking part, color information may also be added to lower the probability of false detection due to the low contrast in grayscale. The whole system may also be modified to be more efficient so that image with higher resolution can be processed real-time. This could significantly improve the system performance by providing more fine details of the target.

## Breakdown of Work

Shunrong Shen: HoG detection. TLD tracking and communication of client and server.
Tongda Zhang: GUI, usage of pre-installed images. TLD tracking and communication of client and server.
Xiaoxiong Lu: Communication between server and Android phone. TLD analysis.

## Reference

[1] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern recognition, 2005

[2] Kalal, Zdenek, “Tracking-Learning-Detection” Pattern Analysis and Machine Intelligence, IEEE Transactions, Vol. 34, Issue 7, Page(s) 1409-1422, 2012

[3] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection" Coference on Computer Vision and Pattern Recognition (CVPR), 2005.

[4] Y. Alper, J. Omar, and S. Mubarak. “Object Tracking: A Survey” ACM Computing Surveys, vol. 38, no. 4, Article 13, December 2006.

[5] Musa, Z. B. and Watada, J (2006): “A Grid-Computing Based Multi-Camera Tracking System for Vehicle Plate Recognition,” Kybernetika Journal, Czech Republic, vol. 42, No. 4, pp. 495-514

[6] Z. Kalal, K. Mikolajczyk, and J. Matas, “Forward-Backward Error: Automatic Detection of Tracking Failures,” International Conference on Pattern Recognition, pp. 23–26, 2010.

[7] J. Y. Bouguet, “Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm,” Technical Report, Intel Microprocessor Research Labs, 1999.

[8] https://github.com/zk00006/OpenTLD