Video Analytics for Airport Security: Determining Counter-flow in an Airport Security Exit

A Thesis Presented
by

Tom Hebble

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements
for the degree of

Master of Science

in

Electrical and Computer Engineering

Northeastern University
Boston, Massachusetts

Aug 2014

Abstract of the Thesis

Video Analytics for Airport Security: Determining Counter-flow in an Airport Security Exit

by
Tom Hebble
Master of Science in Electrical and Computer Engineering
Northeastern University, Aug 2014
Dr. Octavia Camps, Adviser

Airport terminals are equipped with hundreds to thousands of surveillance cameras, yet most surveillance systems merely facilitate the displaying of the video feeds and their recording. Advancements in video analytics and computer vision research over the past decade have made it possible to leverage these extensive video surveillance systems for autonomous, video-based, persistent intelligent surveillance as well as reconnaissance and assessment.

The real-time detection of individuals circumventing checkpoints is identified by the U.S. Transportation Security Administration amenable to video analytics solution using existing cameras. When an individual(s) enters a secure terminal using an exit corridor, the airport may need to be evacuated and areas searched before airport activity can resume. This process called a ”dump” can cost the airport, the carriers, and the passengers millions of dollars.

This paper presents a solution to the counter-flow detection challenge identified by the U.S. Transportation Security Administration. To achieve this goal, the algorithms must detect motion against the normal flow, tag the individual(s) performing the counter-flow, generate an alarm and log the event. Furthermore, all of these tasks must be performed in real time, with zero missed detections and low false alarms. To validate the algorithms accuracy, an ”In-The-Exit” video dataset containing mock security breaches has been created using video captured at the Cleveland Hopkins International Airport. A prototype In-The-Exit application using the solution presented in this paper is currently in testing at the Cleveland Hopkins International Airport to aid the TSA in monitoring corridors.

Contents

List of Figures

List of Tables

List of Acronyms

ALERT Awareness and Localization of Explosives Related Threats, a multi-university Center of Excellence sponsored by the U.S. Department of Homeland Security.

CFV Counter-flow Visualization.

CLE Cleveland Hopkins International Airport.

DHS U.S. Department of Homeland Security.

FSD Federal Security Director.

GPU Graphics Processing Unit.

HOG Histogram of Oriented Gradients.

ITE In-The-Exit; a target traversing a designated exit lane in the wrong direction.

KLT Lucas Kanade Tracker.

MOG Mixture of Gaussian probability model.

NVR Network Video Recorder.

OUP Office of University Programs.

SCR Siemens Corporate Research, an industry partner of ALERT.

TSA U.S. Transportation Security Administration.

TSO Transportation Security Officer.

VAST Video Analytics Surveillance Transition Project.

Acknowledgments

This material is based upon work supported by the U.S. Department of Homeland Security, Science and Technology Directorate, Office of University Programs, under Grant Award 2013-ST-061-ED0001 and Grant Award 2011-ST-104-000037. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

Here I wish to thank those who have supported me during the process of this thesis work. Without their help this thesis could not have been completed.

My advisor, Professor Octavia Camps, whose instruction and guidance in advanced computer vision and video analytics methods were paramount in the research needed to complete this thesis.

Richard Moore, John Romano, and Alyssa White for their administrative support and customer-focused guidance critical for the successful development and deployment of the prototype system.

Oliver Lehmann for his help in the debugging and optimization of the final counter-flow detection algorithm as well as his development of a graphical user interface allowing its use by airport security personnel.

Edward Hertelendy and all of the TSA personnel at the Cleveland Hopkins International Airport for their coordination of and assistance during our trips to Cleveland.

The Transportation Security Officers for the diverse and challenging execution of hundreds of mock security breaches for the video dataset used to evaluate our counter-flow detection methods.

The researchers at Rensselaer Polytechnic Institute and Boston University who worked in parallel developing their own unique counter-flow detection solutions. The friendly competition helped push all of the researchers to develop one-of-kind innovative solutions for this project.

Staff at the ALERT Research Center who, prior to my involvement in this project, worked with the TSA to identify counter-flow detection as a challenge in airport security that could be greatly alleviated using state-of-the-art video analytics methods.

Kristin Hicks, for her help in coordinating and supporting my participation in the HS-STEM CDG graduate fellowship program which providing funding for the research in this thesis.

Chapter 1

Introduction

Airport terminals are equipped with hundreds to thousands of surveillance cameras, yet most surveillance systems merely facilitate the displaying of the video feeds and their recording. Advancements in video analytics and computer vision research over the past decade have made it possible to leverage these extensive video surveillance systems for autonomous, video-based, persistent intelligent surveillance as well as reconnaissance and threat assessment.

The real-time detection of individuals circumventing checkpoints is identified by the U.S. Transportation Security Administration (TSA) amenable to video analytics solution using existing cameras. While passengers and their baggage go through extensive screening prior to entering a secure airport terminal, passengers leave most airports through a simple one-way corridor monitored by a Transportation Security Officer (TSO). When an individual(s) enters a secure terminal using an exit corridor, the airport may need to be evacuated and areas searched before airport activity can resume. This process called a ”dump” can cost the airport, the carriers, and the passengers millions of dollars.

This paper presents a solution to the In-The-Exit (ITE) challenge identified by the TSA. To achieve this goal, the algorithms must detect motion against the normal flow, tag the individual(s) performing the counter-flow, generate an alarm and log the event. Furthermore, all of these tasks must be performed in real time, with zero missed detections and low false alarms.

A partnership with the CLE has allowed the creation of an ITE event video dataset as well as providing a real-time platform for prototype evaluation. Offline test results using the video dataset and TSA feedback from the real-time testing in CLE have been used to evaluate the performance of the algorithms developed. A prototype ITE application using the solution presented in this paper is currently in testing at the CLE to aid the TSA in monitoring corridors.

1.1 Motivation

ITE breaches have consequences in any commercial airport. A breach is said to have occurred when an individual or object has entered a secure region of the airport without being subjected to proper screening procedures. The exit corridors intended for arriving passengers are a site of such breaches. Exit corridors typically face periods of significantly increased pedestrian traffic corresponding to the arrival of larger aircraft. In airports across the USA exit corridors are usually supervised by a TSO with the responsibility of alerting additional security personnel when needed.

Although the vast majority of ITE events are benign, the potential consequences are serious. In common benign cases, there is a substantial economic cost of ITE events. The movement of people in and out of the secure terminal must be halted. At larger airports, this halting of airport operations can cost millions of dollars.

1.2 Thesis Outline & Contributions

The remainder of this thesis is organized as follows: Chapter 2 provides background on the organizations and prior efforts leading to the contributions of this thesis. Chapter 3.1 details our approach to the research and development efforts of this thesis and presents the five complete solutions evaluated. Although only one of these solutions was ultimately found to satisfy all of the requirements of our specific ITE application, we feel that the other approaches may be helpful for the development of similar systems in different environments. In any case, the lessons learned from these failed attempts are relevant for the development of video analytics solutions for any high-risk surveillance and security application. The results and conclusions of the presented solutions are detailed in Chapter 4.

Chapter 2

Background

This section will provide an overview of the efforts leading up to the research and development work presented in this thesis. The participating organizations as well as their preliminary work to identify applications for video analytics research and secure funding for the creation of this project are detailed.

2.1 ALERT DHS Center of Excellence

The U.S. Department of Homeland Security (DHS) and the TSA were created in response to the growing national concern over airport security following the September 11, 2001 terrorist attacks. Since its creation, the DHS has understood the importance of the successful application of the very latest security innovations from university-based research communities. The DHS Science & Technology Directorate Office of University Programs (OUP) works to maximize the return on investment from university-based research funded by the DHS through the sponsorship of Centers of Excellence [1]. Headquartered at Northeastern University, the Awareness and Localization of Explosives Related Threats (ALERT) DHS Center of Excellence ”seeks to conduct transformational research, develop technology, and provide educational development to improve effective characterization, detection, mitigation and response to explosives-related threats facing the country and the world” [2].

2.2 Video Analytics Research

Recent years have seen an exponential increase in the quality and quantity of computer vision and video analytics research. With a greater focus on real-time performance in the computer vision research community as well as advances in processors and Graphics Processing Unit (GPU) technology, the results of much of this research are ready to provide computer vision solutions to real-world problems. Despite this, the state-of-the-art methods from this research are still largely un-applied.

Subject matter experts at ALERT have concluded that this research is mature enough for use in testbed applications. Working with the TSA Federal Security Director (FSD) of Northern Ohio, ALERT identified ITE security breaches as a security challenge faced by the TSA in airports nation-wide that could be benefit from the application of state-of-the-art video analytics research. In a collaborative partnership with CLE, the FSD was able to provide sample videos to ALERT for proof-of-concept and feasibility studies.

2.3 ALERT VAST Project

The Video Analytics Surveillance Transition (VAST) Project is a joint venture between researchers from Northeastern University, Rensselaer Polytechnic Institute, Boston University, Siemens Corporate Research (SCR), and the TSA. The VAST Project was created with the intent to: (1) facilitate the transition of real-world issues into algorithms and into commercial products; (2) provide the university researchers with new insight into the real-world domain of airport security, (3) provide industry with access to counter-flow data for testing, and (4) provided the TSA with improved traveler understanding essential to national security. The project was also intended to showcase the benefits of and provide an example for future DHS projects using a collaboration of resources from universities, industry, and government.

The first VAST program, ITE counter-flow detection, is outlined in the following section (see 3.1). Researchers at all three universities worked independently to develop their own unique solutions to the ITE security challenges. The solutions for this problem developed at Northeastern University are explained and evaluated in the remainder of this thesis.

Chapter 3

Approach & Design

3.1 Problem Definition

In this section, we will detail the ITE security challenges faced in airports worldwide as well as specify, in detail, the final requirements presented by the TSA at CLE and the verification and validation procedures used to determine success. Due to the flexible nature of academic research projects with regards to deliverables, the requirements were amended at times throughout the duration of the project.

Prior to the start of research, personnel from the TSA and ALERT surveyed the facilities and observed passenger traffic patterns at CLE resulting in the selection of the South Exit Lane, equipped with four existing cameras to be used for the real-time testing of our ITE application and the creation of a video dataset for offline evaluation (see Figure 3.1).

3.1.1 Requirements

- **Zero missed detections:** Any undetected ITE security breach is unacceptable for the TSA.
- **Low rate of false alarms:** Surveillance application such as this are only effective when the outputs are monitored and promptly assessed by security personnel. Excessive false alarm outputs from the ITE application are distracting.
- **Real-time Operation:** The output delay of the ITE application must be short in order to facilitate a rapid response from TSA personnel. The application will run on computer hardware provided by ALERT; an Alienware desktop PC containing an Intel i7-3820 Quad-core processor with 16GB RAM and a NVIDIA GTX580 GPU with 1.5GB RAM.

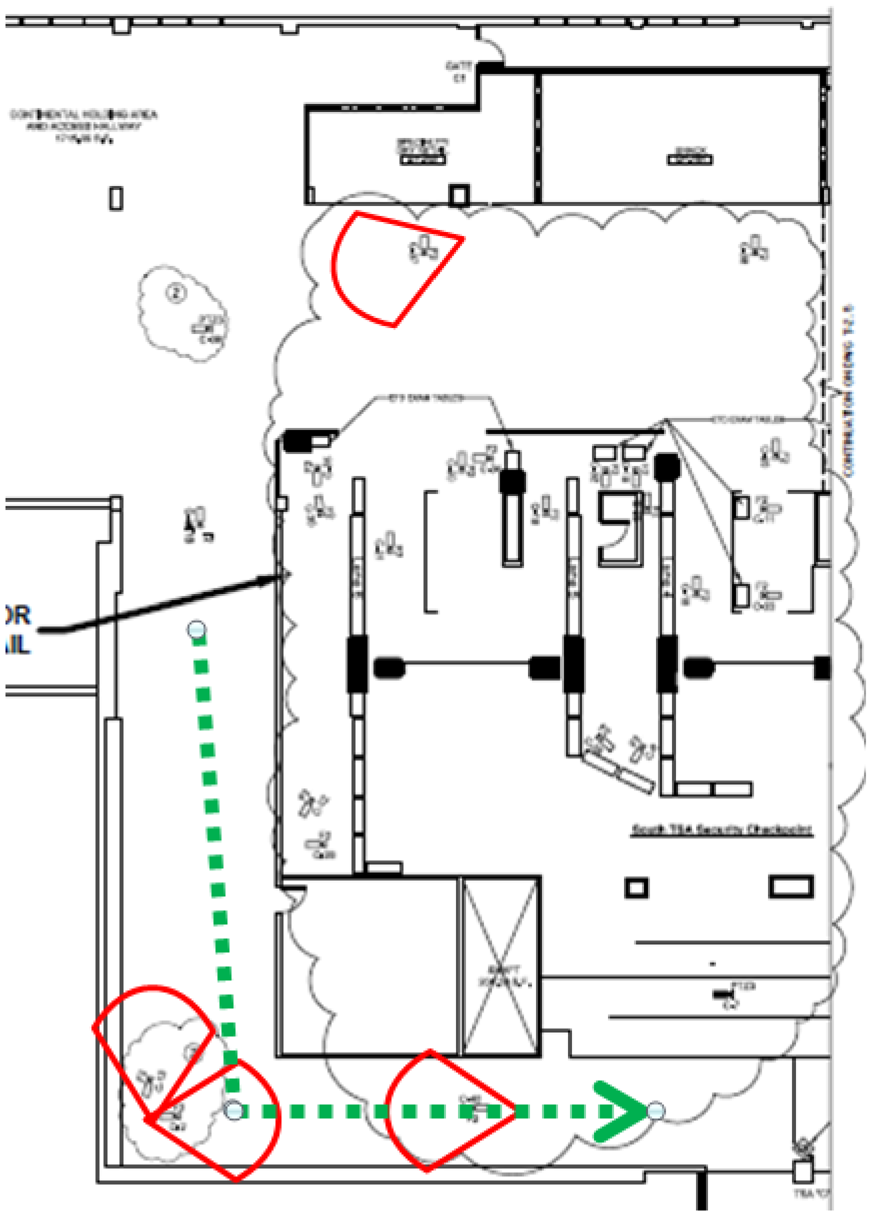


Figure 3.1: Layout of South Exit Lane at CLE showing camera positions (red) and direction of normal pedestrian flow (green)

- **Tagging of counter-flow:** When an ITE event occurs, the counter-flow detection algorithm must facilitate the tagging of counter-flow objects(s) by providing their location(s) in time and space.
- **Utilize existing camera systems:** Due to the costs of installing new camera systems in airports nationwide and the widespread deployment of existing cameras, the ITE application must exploit existing airport camera systems.
- **Portable to additional exit lanes & airports:** Future deployment of the ITE application to additional exit lanes and airports shall require only a minimal effort to tune a limited set of parameters. It will not require the recording of large training datasets at each new exit lane.

3.1.2 Test Bench

ALERTs partnerships with the TSA at CLE and SCR allowed the collection of camera video data and the online, real-time testing of our algorithm using a live video stream from the cameras. To ensure that our research did not interfere with the existing airport security effort, we were provided with a dedicated Ethernet subnet for streaming video data across system components. Bosch VIP X1 XF Video Encoders connected to the analog cameras provided live H.264 video streams to the Network Video Recorder (NVR) provided by SCR (Data Processing PC) and the desktop PC running our ITE application (Video Analytics PC). SCR also provided researchers with a Video Access Library that was used to capture live video from the Bosch encoders as well as previously recorded video from the NVR.

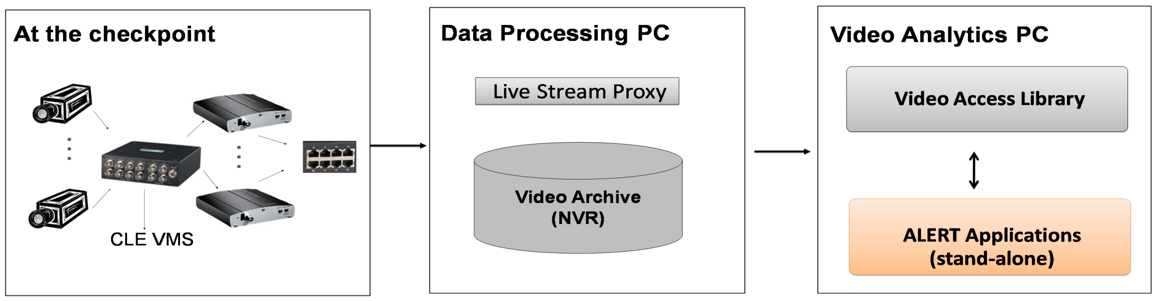


Figure 3.2: Block diagram showing video hardware installed at CLE

For a period of twelve weeks, TSOs at CLE staged up to ten mock ITE events per day. The videos of these events were used to create a video dataset of ITE events that were used to evaluate the algorithms prior to deployment in CLE. The complete ITE dataset includes 21.4 hours of video data containing 628 ITE events as well as three hours of video containing no ITE events (used for estimating the probability of false alarm). Due to time synchronization issues, some of the event videos were not included in this multi-camera dataset. Recorded events that did not contain the security breaching individual in the first three exit lane cameras were not included (the TSOs staging events often proceeded to the TSA staff room before reaching the forth camera). For the algorithms utilizing only a single camera, we were able to use an extended version of dataset containing 666 ITE events.

3.1.3 Software Verification & Validation

The verification and validation of our ITE solution was a joint effect requiring the participation of researchers, ALERT staff, and TSA personnel. The following conditions were used to verify that each requirement was met by our proposed solution.

- **Zero missed detections:** It is impossible to test every possible ITE event that may occur. To satisfy this requirement, our counter-flow detection algorithm must be successful in detecting every ITE event contained in the South Exit Lane video dataset created for this project.
- **Low rate of false alarms:** The detection results from running the application live at CLE for at least one week are used to determine the probability of false alarm. This requirement is satisfied if the results indicate, on average, fewer than ten false alarms per day.
- **Real-time operation:** Satisfied after observation of the complete application running successfully in the lab for a period of at least one week.
- **Tagging of perpetrator:** Internally, the mechanisms used for counter-flow detection must generate information that can be used to draw a bounding box (spline) to highlight the perpetrator(s) causing the event.
- **Utilize existing camera systems:** This requirement will be met by satisfying all other requirements in both the South and Central Exit Lanes at CLE without the need to install additional cameras.
- **Portable to additional exit lanes & airports:** Upon verifying that the other requirements have been met for the South Exit Lane at CLE, the counter-flow detection application operate successfully in the Central Exit Lane at CLE without the creation of an additional ITE event

video dataset. Since a test dataset does not exist for the Central Exit Lane, verification will be determined using feedback from TSA personnel.

The ITE solution presented in this thesis is validated by the deployment of a prototype system in CLE. In addition to the counter-flow detection software detailed in this paper, the Counter-flow Visualization (CFV) application was developed to provide a graphical user interface for use by TSA personnel. The CFV application displays a live feed of the exit lane cameras and provides visual and audible cues when a counter-flow event is detected. During such events, a bounding box (spline) around the perpetrator(s) is drawn in the displayed video and event information recorded. Using the recorded event information, the CFV software includes functionality allowing security personnel to review the event and provide input on the validity of the counter-flow detection. Feedback from the CFV software as well as qualitative feedback from security personnel using the application is used for the validation of the final counter-flow detection software presented in this thesis as well as our complete ITE solution.

3.2 Approach

The counter-flow detection algorithms developed and tested for this project can be deconstructed into three, largely independent, functional components each with unique responsibilities and challenges. In general, the three components are responsible for the following tasks:

1. Potential ”objects of interest” (usually pedestrians) must be identified.
2. The movement of these objects then must be reliably tracked over time.
3. The tracking results must be used to determine if an ITE breach has occurred.

The remainder of this section will outline the methods used to reliably and accurately perform the aforementioned tasks. Greater detail regarding the implementation and integration of these methods into a complete ITE application will be presented in section 3.3.

3.2.1 Target Initialization

Since calculating reliable and accurate optical flow for the image is impractical for this application due to the real-time requirements, our algorithm must be more intelligent in its selection of objects to track. Maintaining the identity of each target as they are tracked will be critical in determining if an ITE breach has occurred. This means that target track-ability is an important factor in evaluating the performance of target initialization methods. Above all else, the target initialization method must not fail to tag any objects that may cause an ITE breach.

3.2.1.1 Corner Features

The results of any visual-based tracking method will suffer (the *aperture problem*) when tracking an image patch with insufficient texture variation [10]. The ”corner-ness” of an image location can be estimated using eigenvalues of the weighted sum of the structure tensor matrix A in equation 3.1, where I_x and I_y are the partial derivatives of the image I [8] [10].

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{3.1}$$

Using corner feature detection for target initialization can help to maximize the accuracy of target tracking. The image contrast from the cameras is great enough that the ”corner-ness” threshold for target initialization can be set such that individuals cannot traverse the camera’s field of view without being detected.

3.2.1.2 Person Detection

In recent years pedestrian detection has become a popular field of computer vision research. Largely motivated by the desire to develop new safety features in automobiles, the application of reliable real-time pedestrian detection methods has been made possible by advances in computing hardware as well as the availability of new annotated video datasets for training and testing (such as the Caltech Pedestrian Dataset [6]). The evaluation of state-of-the-art methods in [7] shows the use of Histogram of Oriented Gradients (HOG) features to be common among the most effective detectors. Figure 3.3 shows an individual detected using this method. Our counter-flow detection applications utilized a HOG feature-based pedestrian detection method [5] implemented in the OpenCV C++ library [4].

3.2.1.3 Foreground Blobs

Due to a lack of potentially view-obstructing objects in the exit lane cameras’ field of view, there exists regions of the image that all pedestrians must pass though at least partially unobstructed. This allows us to limit the search space of our detection algorithms, reducing its computational

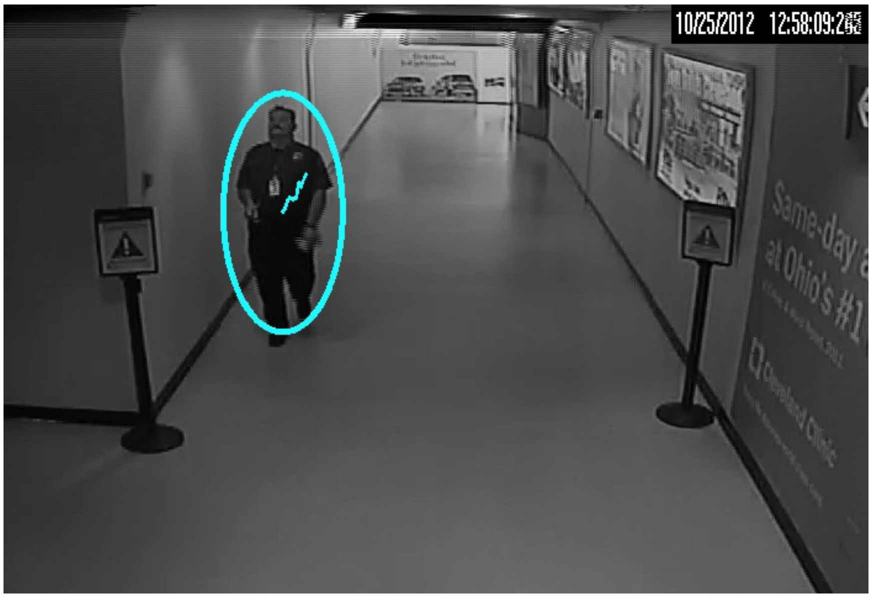


Figure 3.3: Detected pedestrian using HOG features

expense, without sacrificing reliability.[1] At CLE the direction of counter-flow is aligned with the vertical axis of each camera's field-of-view, allowing us to select a vertical range to search for objects of interest (see Figure 3.4).

Since the image region selected for detection is not large enough for accurate pedestrian detection, background subtraction was used to detect objects of interest. Our counter-flow detection applications utilized a MOG model background subtraction algorithm [13] [14] implemented in the OpenCV C++ library [4]. The target initialization positions and sizes were generated using a small sliding window (24x12 pixels) search of the detection region for image patches containing a percentage of foreground pixels greater than a predetermined threshold. At these locations, the window size was increased until the foreground percentage dropped below another predetermined threshold or a maximum size parameter was reached. A new target would not be initialized if the final window size remained smaller than a minimum size parameter.

[1] We acknowledge that a perpetrator and multiple accomplices may be able to circumvent this detection method but only with extremely detailed a priori knowledge of application and camera parameters. Additionally, we believe such a coordinated action could not be executed quickly or discretely enough to avoid attention from TSOs or suspicion from other passengers.

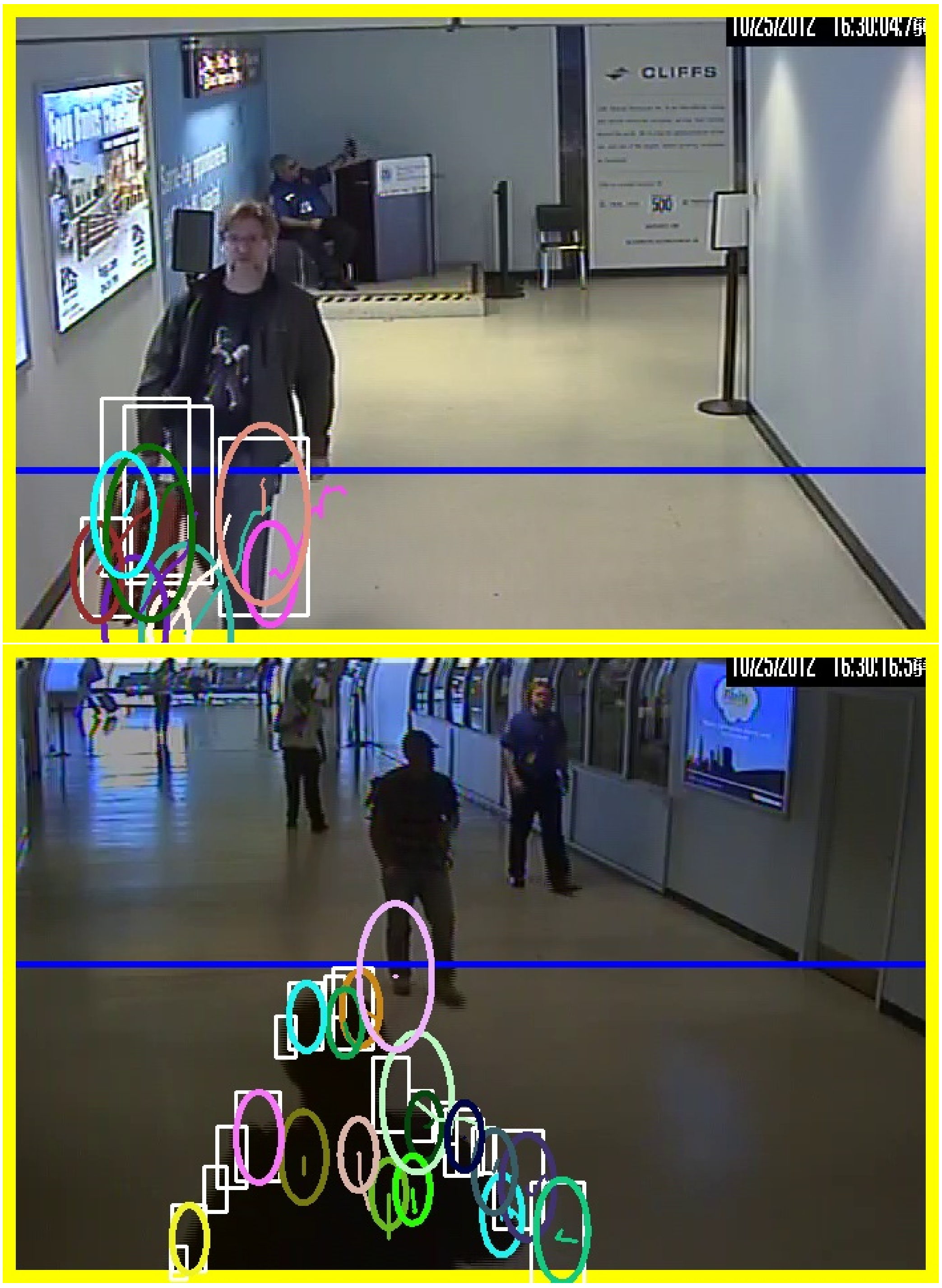


Figure 3.4: Targets are initialization when foreground is detected below the blue line. Colored ellipses indicates an active target. The white rectangles indicate newly detected foreground regions.

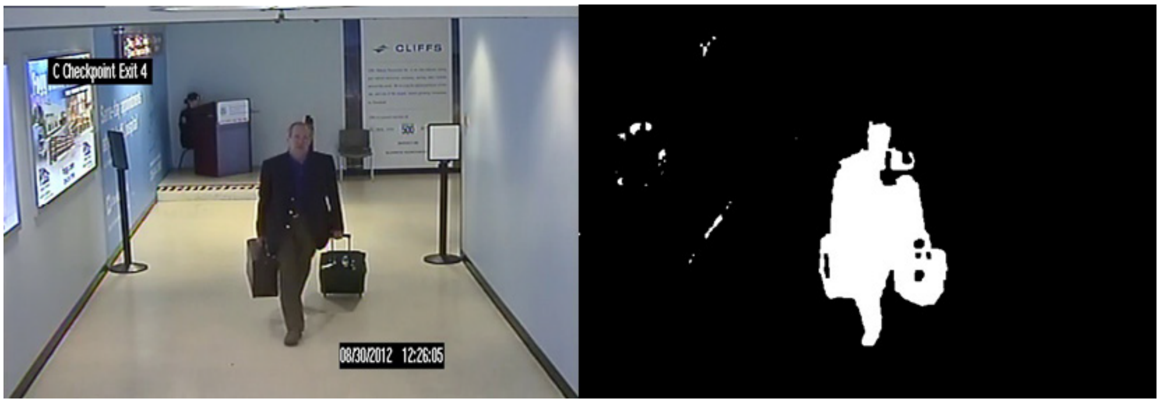


Figure 3.5: Camera video frame (left) with MOG Background subtraction results (right). White indicates foreground region.

3.2.1.4 Naive Over-Tracking

Relying entirely on background subtraction for target initialization proved to be unreliable in some cases. Background subtraction algorithms can require a considerable number of frames to learn the background model and provide a valid output. The ITE video dataset used for evaluation contains a number of videos where a counter-flow event occurs before a valid background model can be learned effecting test results in unpredictable ways.

At this point in the project we had also developed an implementation of our visual tracker (see 3.2.2.2) that utilized the computer's GPU. The use of GPU acceleration allowed our application to initialize a much larger number of tracking targets without sacrificing real-time operation. Capitalizing on this, we developed a target initialization method that defined a number of location where trackers were initialized on every frame, regardless of foreground detection. In each frame, new targets are initialized at the locations shown in Figure 3.6 if there are not any existing tracks already in that location (as determined by an overlap percentage greater than a threshold parameter).

3.2.2 Tracking and Target Termination

Once objects of interest have been identified, the objects' locations in the video frame must be tracked over time. In addition to accurate estimation of target position, the tracking algorithm must be able to determine when its output is no longer valid (e.g. target obstructed and not visible). The tracker's ability to self-evaluate is critical for minimizing the number false alarms from the complete system. The Lucas Kanade [11] and Circulant [9] tracking methods, which could be integrated in a complete system satisfying all of the requirements, were evaluated for use

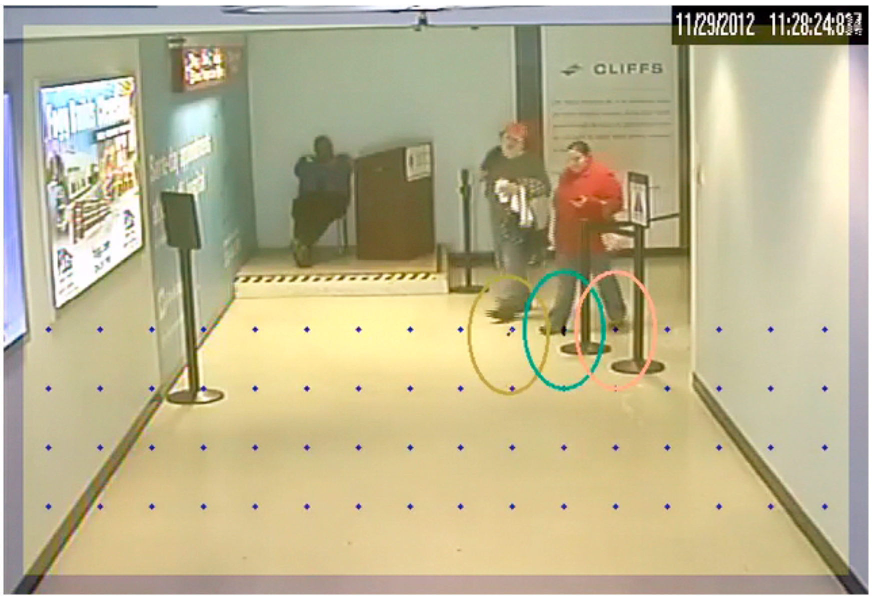


Figure 3.6: Over-Tracking Target Initialization. Camera video frame showing locations where trackers will be initialized.

in our counter-flow detection algorithm. We also examined augmenting these primary trackers with additional features, such as background subtraction, to improve accuracy.

3.2.2.1 Lucas Kanade Tracker

The Lucas Kanade Tracker (KLT) [11] is widely used in computer vision for calculating frame-to-frame optical flow. This method can quickly calculate small (compared to size of target) displacement using the temporal gradient of two consecutive frames. It attempts to match local patches around features across frames using an affine distortion model to take into account appearance changes from frame to frame, i.e. find an affine transformation between frames that minimizes the sum of the square errors between a patch in frame at time t and an affine transformed patched in frame at time $t + 1$:

$$E(u, v) = \sum_{x,y} [I((W(x, y); P)) - T(x, y)]^2 \tag{3.2}$$

where $T(x, y)$ is the image gray level of the template at pixel location (x, y) at time t, $W(x, y)$ is an affine warping transformation defined in terms of a set of parameters P, I is the frame at time $t+1$, $E(u, v)$ is the sum of square errors for a match corresponding to a displacement of the template equal to (u, v). The KLT preforms extraordinarily well on patches with significant texture variations, but struggles tracking points in more homogeneous regions [10]. As a result, the KLT was only effective when initialized using the corner detection method (see 3.2.1.1).

Our counter-flow detection applications utilized a sparse iterative multi-scale version of the KLT [12] implemented in the OpenCV C++ library [4]. This tracking algorithm using a pyramid representation containing the grayscale image at four scales. The KLT algorithm is applied at each scale, starting with the deepest (lowest resolution) scale. For the remaining scales, the tracking estimation from the previous scale is used to pre-translate the second image, resulting in a smaller residual displacement vector computed in a standard KLT step, even when the target's motion is large. This four scale pyramid implementation can handle maximum target displacement fifteen times that of a single-scale KLT implementation. As noted in [12], this tracking method is unable to reliably self-evaluate. For this tracker, we calculated the foreground-to-background ratio of an image patch surrounding the target using the same background subtraction method used for blob detection (see 3.2.1.3). The track was terminated if the ratio fell below a threshold parameter.

3.2.2.2 Circulant Tracker

The Circulant Tracker [9] follows its target by performing template matching between the current frame and the previous frame. However, instead of performing cross-correlations between the target and multiple sliding windows on the current frame, the circulant tracker performs a single cross-correlation operation involving a ”circulant” template image and an image patch of the same size from the current frame.

When a tracking target is selected, a Gaussian kernel is applied to the auto-correlation of the image patch containing the target. An optimal Weiner filter is calculated to obtain a Gaussian response with a specified variance parameter. The cross-correlation is computed in the frequency domain by first computing the Fourier transform of the circulant template of the target, $T(u, v)$, and of an image patch in the next frame, $I(u, v)$, centered at the previous location of the target, where u and v are the spatial frequencies. The cross-correlation input is made nearly ”circulant” by applying a Gaussian kernel to the cross-correlation between the pixel intensity values of the template and current frame. The circulant structure in the time domain causes symmetry in the frequency domain that can be exploited when calculating cross-correlation to improve computational efficiency and achieve real time performance. Then, the most likely location of the target in the current frame is found by searching for the maximum value when the kernel cross-correlation is passed though the filter calculated at tracker initialization. Figure 3.7 shows an example for the circulant tracking method. The circulant tracker has the ability to self-evaluate. Invalid tracking results can be identified by a low peak-to-sidelobe ratio of the output response [3].

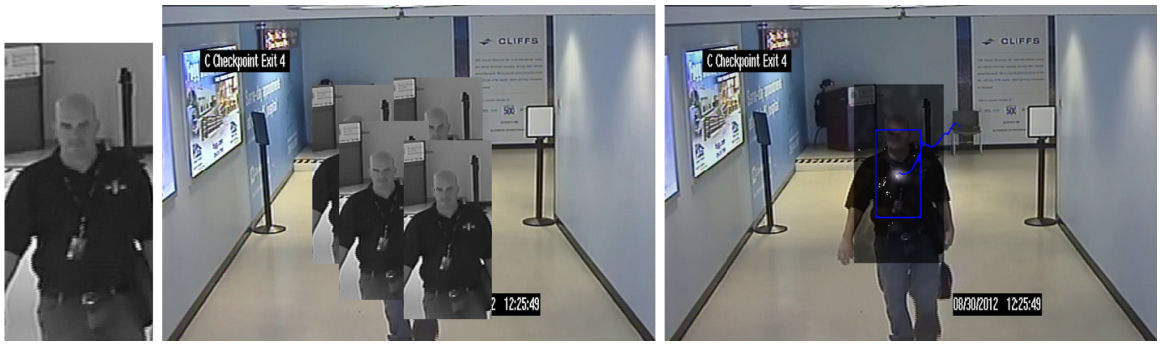


Figure 3.7: Pedestrian tracked using circulant tracker. Cross-correlation with a target template (left) is used to search possible locations (center). The correct location (right) is indicated by maximum of cross-correlation result.

3.2.3 Counter-Flow Classification

The final processing component is responsible for determining if an ITE event has occurred using the object tracking information from the previous stage. Traditional multi-dimensional machine learning methods, such as boosting and SVM, have difficulty correctly classifying 100% of the data available. The requirements of this application designate even a single missed detection as unacceptable. Our approaches to the task of counter-flow classification rely low-dimensional inputs (one or two dimensions) with threshold parameters able to be set such that 100% of the ITE events contained in the video dataset are correctly detected.

To detect counter-flow in a single camera, we found two values to be important for correct classification. The first is the number of tracks found to be moving in the counter-flow direction. The second, more reliable, value is a magnitude of movement in the direction of counter-flow. In our complete applications, we used thresholds for one or both of these values to classify an ITE event. For multi-camera implementations, the detection results from each camera are fused using Boolean logic that accounts for the probability of detection in each camera.

3.3 Design and Implementation

This section provides greater detail of the configuration and implementation of the five complete counter-flow detection applications developed and evaluated for this project. Except for the Corner Tracklets approach (see 3.3.1), which was abandoned early on in the project, each implementation was evaluated using the ITE event video dataset (see 3.1.2).

3.3.1 Corner Tracklets

In this approach, with the least computational cost, the system detects local features on each frame and tracks them using a Lucas Kanade tracker for at least five frames as shown in Figure 3.8. Events are detected when there are at least three trajectories in the wrong direction. The benefit of this approach is that it can run extremely fast (faster than frame rate) providing timely alerts and achieving zero missed detections.

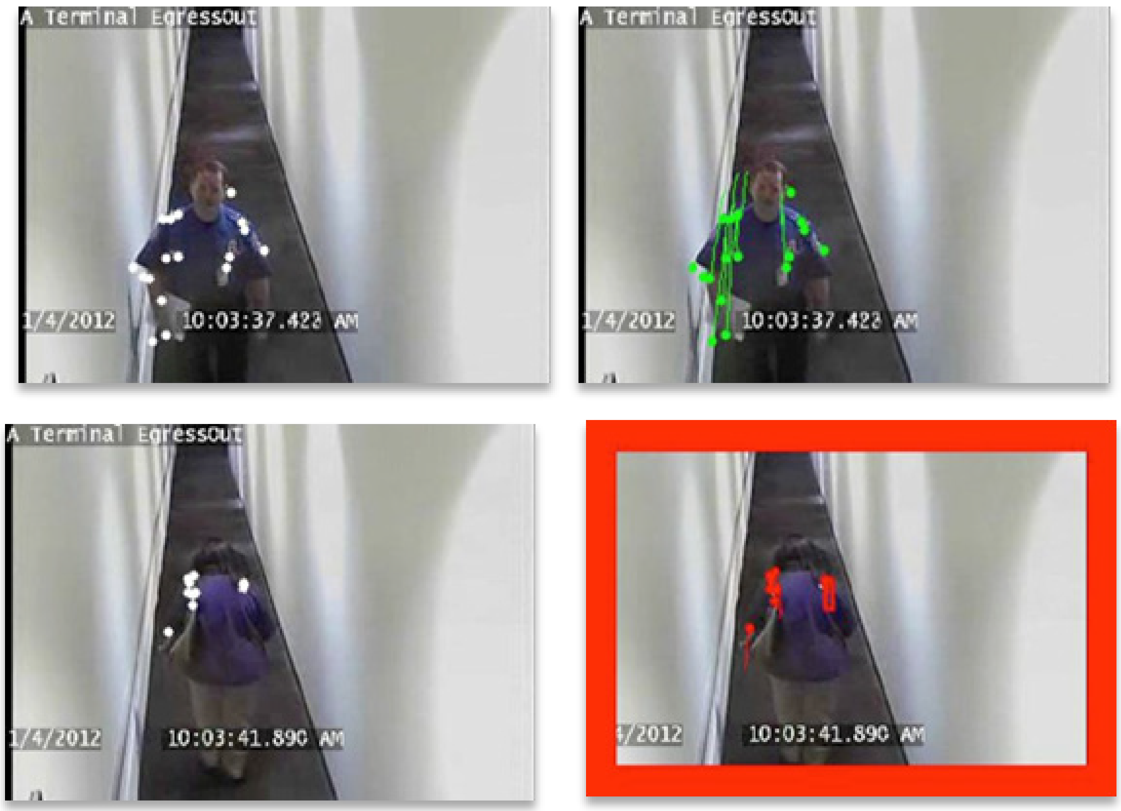


Figure 3.8: Corner Tracklets. Detected corners are tracked (left) and then used to identify counter-flow (right).

In our tests of the single camera implementation, we were able to achieve rates of 39 frames/sec using a single thread and up to 100 frames/sec using three threads while processing recorded video on a 2.2. GHz Intel Core i7 machine. However, it typically produces a large number of false alarms. In most cases, the false alarms are due to brisk movements of the arms such as lifting a bag or answering a cell phone, events that are too common for this approach to be useful. Multiple camera implementations of this method, which used a simple voting scheme to determine the final output, were also evaluated in hopes that this could be used to minimize the number of false alarms. Unfortunately, false alarms were so frequent that they often occur concurrently on more than one camera. Furthermore, because it relies on local features (i.e. each individual feature is tracked independently of the others), this approach does not provide a good ”tagging” mechanism to track the individual in other views. Because of these reasons, this method was not pursued any further.

3.3.2 Multiple Camera Pedestrian Tracking

The two alternative methods described in this section introduce two important modifications to the previous implementation. First, tracking is performed at a higher level instead of at the local features level. By higher level, we mean that the tracker follows a larger region in the image. The region may be defined as the output of a pedestrian detector or as a set of contiguous pixels that exhibit similar motion. The reasons for this change are twofold. Foremost, we believe that using a global tracker would reduce false alarms generated by localized counter-flow movements due for example to a swinging arm while walking. In addition, having a global tracker in place would provide a better interface for the next stage when the tagged individual must be tracked across the airport terminal.

The second modification is to use information from all the available cameras to further reduce false alarms. However, due to the drain in computational resources caused by processing each video stream, not all cameras are treated equal. We explored two possibilities where two or one of the cameras are used as ”primary” cameras and the remaining camera(s) is(are) used on a secondary or ”verification” role. Using this allocation of computational resources, the data from the primary cameras is processed using more reliable but computationally expensive algorithms, while the data from the verify cameras are processed using less demanding software to filter out possible false alarms created by the other cameras.

3.3.2.1 Two Primary Cameras with One Verify Camera

In this approach, the two cameras facing the security line (labeled #1 and #2 in Figure 3.13) are used as primary cameras and the remaining camera (labeled #3 in Figure 3.13) is used as a verification camera.

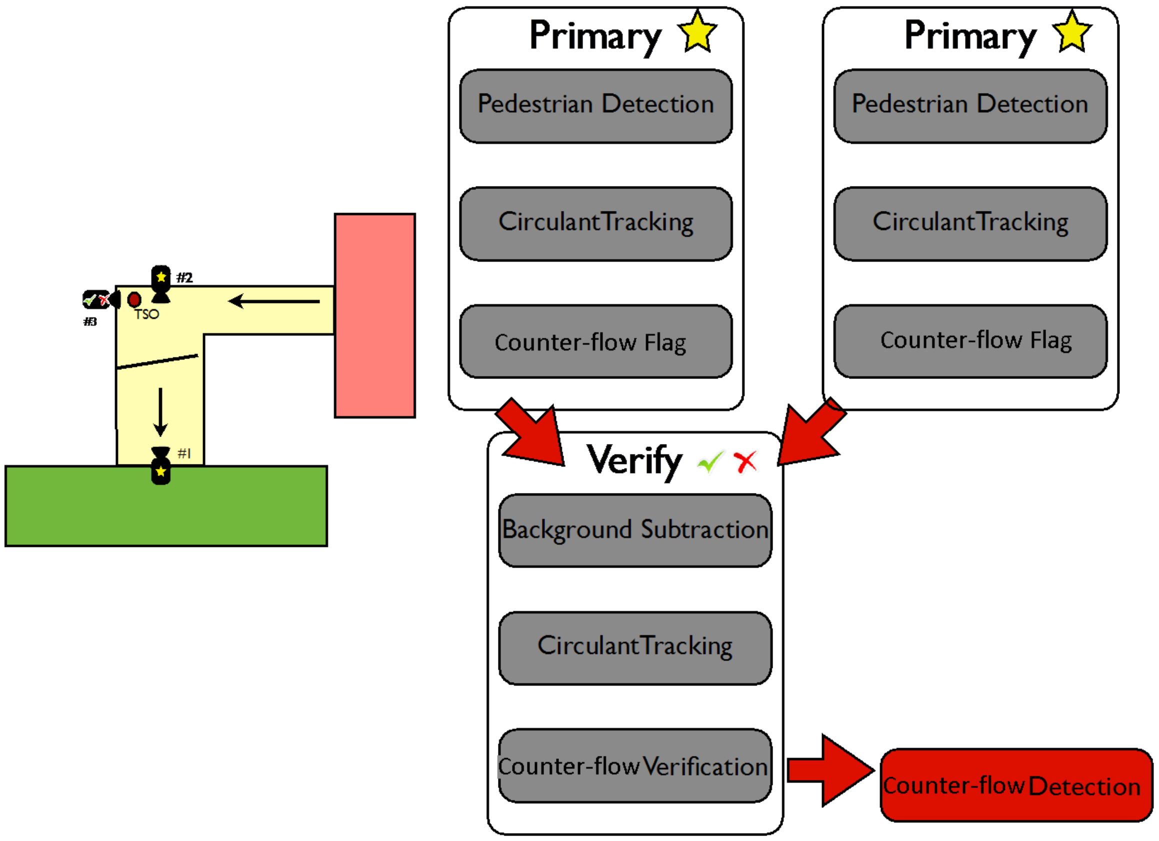


Figure 3.9: Camera locations (left) and flow-chart (right) of Two Primary Cameras with One Verify Camera counter-flow detection algorithm.

In the primary cameras, we use high-quality pedestrian detectors based on HOG features. The detected pedestrians (as a whole) are tracked in the input streams from the primary cameras using a circulant tracker. Finally, the third camera (#3) (called here the ”verify” camera) facing into the terminal is used to confirm true events after the transgressor crossed the security line and turned into the sterile area. However, due to the fact that the processes required by cameras #1 and #2 are quite expensive, the algorithm used with this camera is less sophisticated and cheaper to compute. For this view, we fall back into the cheaper, albeit less reliable, background subtraction method to described in Section 3.2.1.3. Since this camera is only used as a validation camera, it was considered that the performance of this algorithm was sufficient for this task.

Once these blobs are detected, they are tracked using the same tracker as with the other two cameras. In order to avoid false alarms from TSA personnel using a hallway door that can be seen from this view, the detector only considers the region in the image below the door. This approach is computationally less demanding than the approach used in cameras #1 and #2 and is only active for a set length of time after an alarm has occurred in one of the two primary cameras.

In this approach, in all three cameras, an event is signaled when a pedestrian's vertical progress (distance of the motion towards the security line) exceeds a threshold value. In the two primary cameras, pedestrian tracks are checked for events once they have ended (i.e. when they leave the screen or are lost by the tracker). If an event is detected in a primary camera, the third camera is activated. This camera checks every track in every frame, including those still active. Finally, the program logs a true event when an event has occurred in any two of the three cameras. Due to the computational cost of the HOG pedestrian detector used in the two primary cameras, this algorithm was run at 15 frames-per-second.

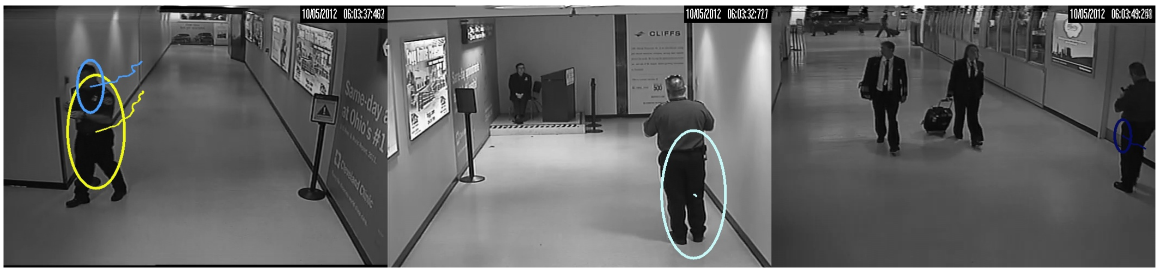


Figure 3.10: Counter-flow Detection using the two primary cameras (left & center) with one verify camera (right) approach.

3.3.2.2 One Primary Camera with Two Verify Cameras

Alternatively, we tested a system where camera #2 is the primary camera and the other two cameras are used as verification cameras as shown in Figure 3.11. Camera #3 is considered the ”first verify” camera and camera #1 is considered as the second verify camera in this approach.

During normal operation, a set length of ”past” video is saved from the second verify camera. When an event is detected in the primary camera, both of the verify cameras become active. The second verify camera begins searching the past video for the perpetrator using the same motion blob tracking methods previously described. To avoid false alarms, only the area below the

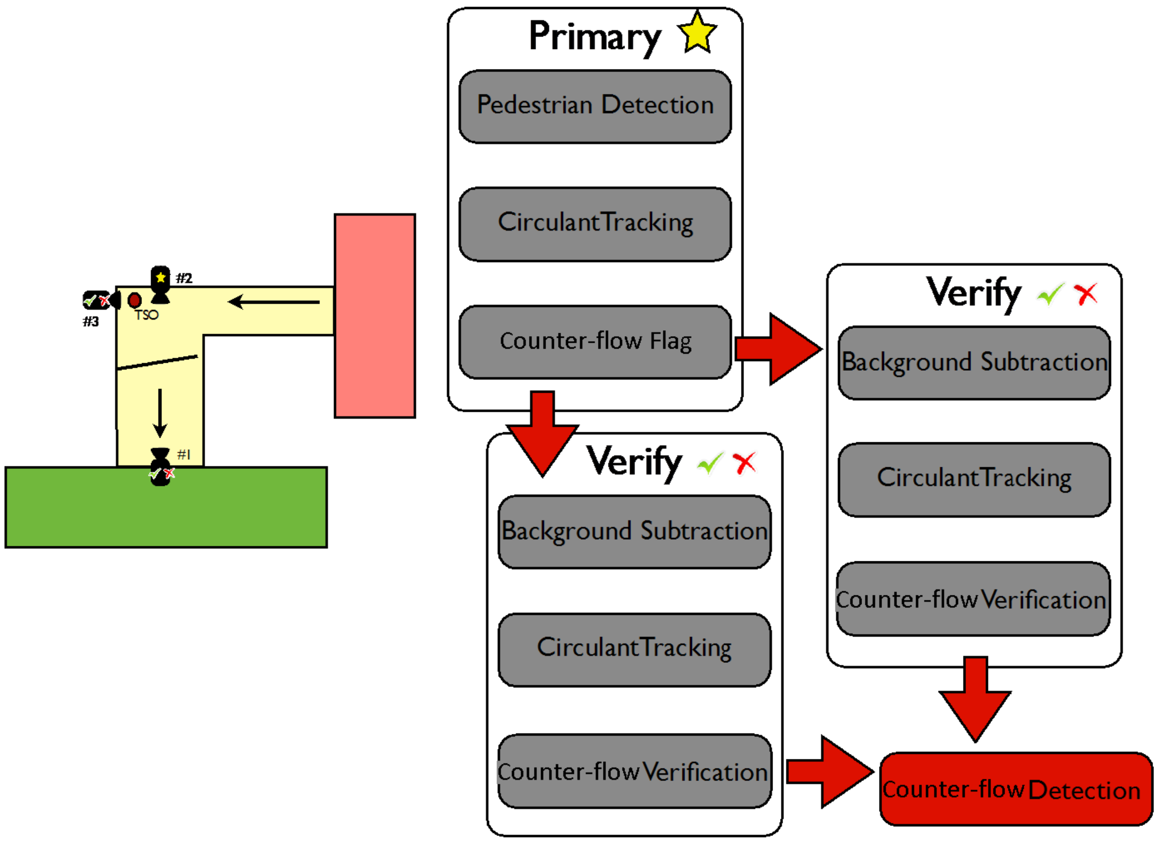


Figure 3.11: Camera locations (left) and flow-chart (right) of One Primary Camera with Two Verify Cameras counter-flow detection algorithm.

exit line is considered for blob detection. The primary camera and first verify camera operate the same as in the previous algorithm.

In this approach, a true event is logged after the primary camera has signalled an event, followed by either one of the verify cameras signalling an event. In all three cameras, an event is signalled when a pedestrian's vertical progress in the image exceeds a threshold value. Again, due to the computational cost of the HOG pedestrian detector, this algorithm was run at 15 frames-per-second.

3.3.3 Single Camera Blob Tracking

After testing the previous algorithms a number of key observations were made. In order to make sure that all events were caught on the primary camera, the vertical progress threshold for an event needed to be set extremely low (10 pixels). This resulted in many false alarms, though the

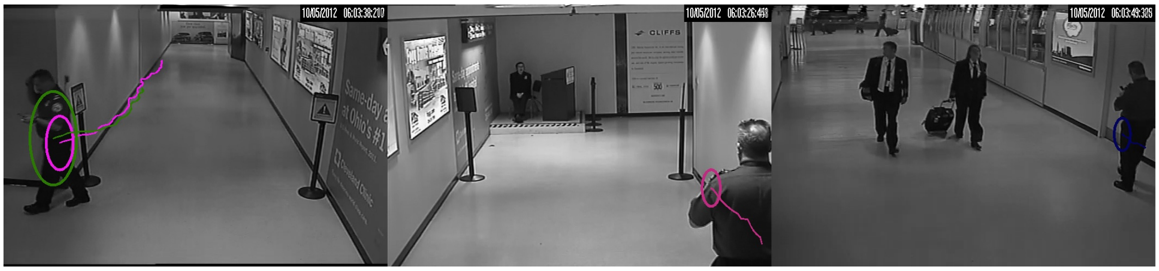


Figure 3.12: Counter-flow Detection using the one primary camera (left) with two verify cameras (center & right) approach.

verify cameras were able to have a much higher threshold and prevent these from being recorded as true events. However, the primary camera still missed a number of events and adjusting parameters such as the frequency of the HOG pedestrian detection had non-deterministic effects on the performance. It was observed that on all the three cameras, the background subtraction is cleanest in lower portions of the frames due to H.264 compression artifacts in the upper portions. Indeed, the motion blob detection using background subtraction is extremely robust on the verify cameras where the detection is only performed on the bottom of the frame.

Thus, in this approach, only camera #2 is used. This camera has a clear view of the exit line and has the cleanest background subtraction results. It also has a well-defined entrance and exit paths for all pedestrians. The algorithm used on the this camera is essentially the same as the one previously used on the second verify camera (see Figure 3.13), except that the detection region was moved up to be centered around the exit line with a height of 100 pixels. Doing this enables the algorithm to catch the individuals entering from the terminal, stepping across the alert line, and returning to the terminal.

3.3.3.1 Smart Foreground Blobs

Using the code from the second verify camera from the previous implementation (see 3.3.2.2) as a starting point, we added functionality to augment the circulant tracker results using output from the background subtraction used for target initialization. If the position from the circulant tracker resulted in the foreground percentage falling below a threshold parameter, the size is increased and the position of the tracked object is moved towards the center-of-mass of the foreground region. This feature improves reliability of the entire ITE application by helping to maintain target identity even when the tracking fails. This results in fewer false alarms by allowing a higher vertical distance threshold to be used for classifying counter-flow events. The brown track in Figure 3.14

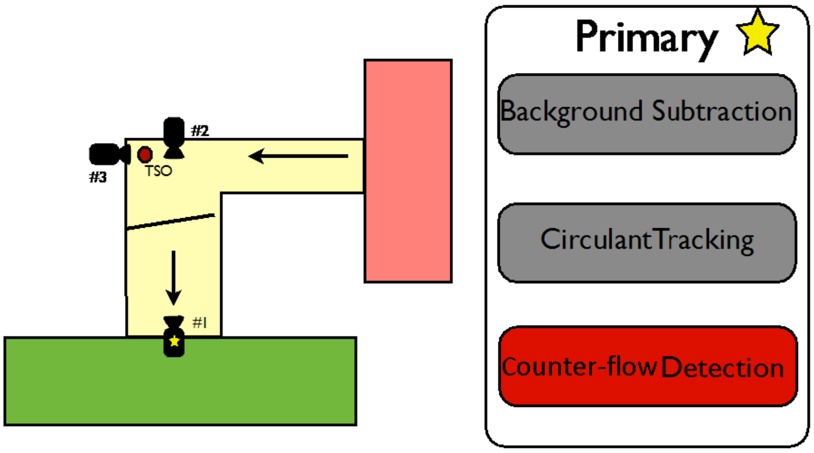


Figure 3.13: Camera locations (left) and flow-chart (right) of Single Camera Blob Tracking counter-flow detection algorithm.

shows an example how foreground measurements can be used to maintain target identity even when the tracker begins to drift.

3.3.3.2 Naive Over-Tracking via GPU Acceleration

Taking advantage of the GPU accelerated circulant tracker implementation (see 3.2.2.2), our final counter-flow detection algorithm utilized the over-tracking target initialization approach (see 3.2.1.4). The GPU tracker implementation was limited to a single target size, but this was made up for by the ability to track a much larger number of targets. This Monte-Carlo style approach ensured that the perpetrator would be tracked by at least one of the many partially redundant initialized targets. In this approach, an ITE event was indicated if the vertical progress of any track surpasses a threshold parameter.

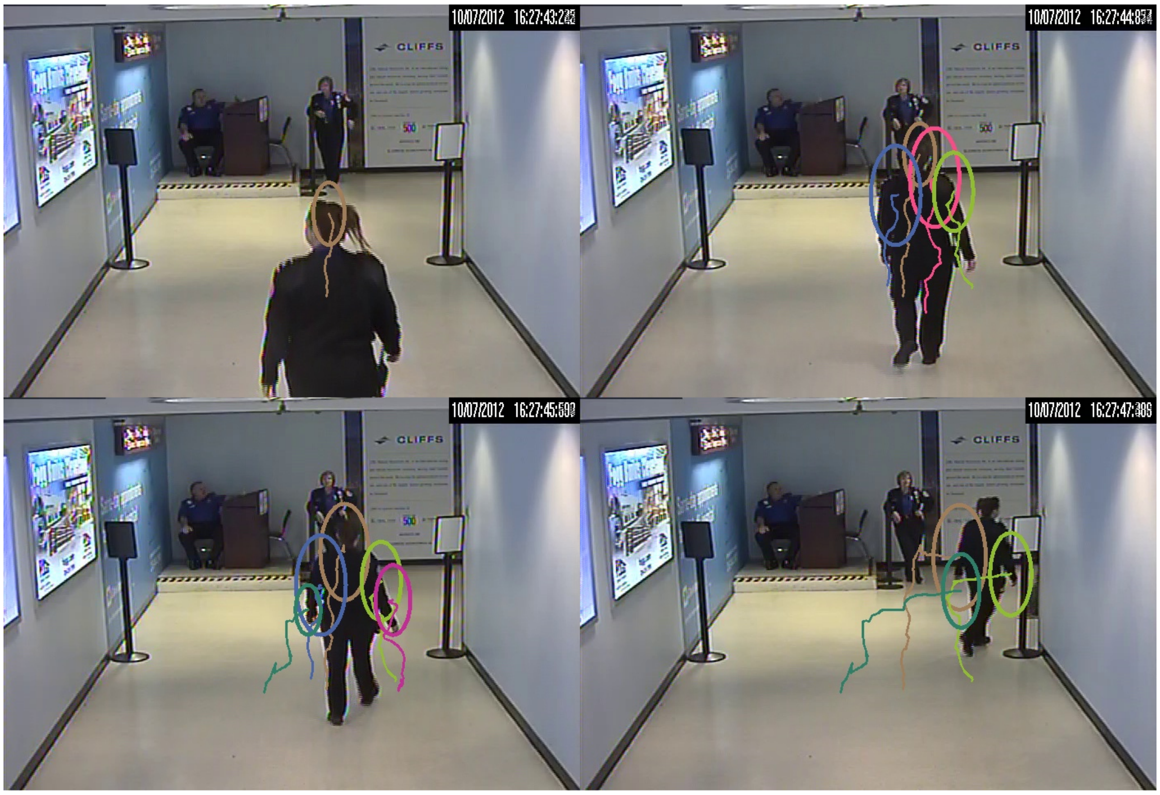


Figure 3.14: Example of smart adapting of target size and position over time using background subtraction output.

Chapter 4

Results & Conclusions

In this section, we will present the results of the ITE solutions previously described. The Corner Tracklets method (see 3.3.1) was abandoned early in the project and is not included in this section. The two multiple camera methods (see 3.3.2) and the single camera smart foreground blob tracking method (see 3.3.3.1) were evaluated and verified using the ITE event video dataset of the south exit lane at CLE. The final method was verified and validated through the deployment of a complete prototype system in CLE monitoring both the south and central exit lanes.

4.1 Cleveland Benchmark Results

	Algorithm:		2P+1V cameras			1P+2V cameras			Single Camera		
Week	Date	Duration (Hrs)	TE	MD	FA	TE	MD	FA	TE	MD	FA
No event test	10/10/12	3.0	0	0	2	0	0	2	0	0	1
4	10/11/12	2.34	77	3	0	77	2	0	84	0	0
6	10/25/12	4.17	85	1	3	85	2	3	92	0	2
7	11/01/12	2.5	84	2	0	84	1	0	87	0	0
8	11/09/12	2.5	82	4	1	82	2	0	85	0	2
9	11/14/12	2.38	75	0	4	75	2	2	78	0	3
10 & 11	11/28/12	4.67	144	4	3	144	5	1	152	0	3
12	12/6/12	2.84	81	0	2	81	0	2	88	0	1
TOTAL		24.4	628	14	15	628	14	10	666	0	12

Table 4.1: Counter-flow detection results from ITE video dataset. TE = Total Events, MD = Missed Detections, FA = False Alarms.

As shown in Table 4.1, the multiple camera pedestrian tracking methods were unable to meet the requirement for zero missed detections. We found that even with highly sensitive parameters and a high frequency of detection, the pedestrian detector was still unable to identify all counter-flow objects (pedestrians). In addition, we were informed by the TSA that in order for our final application to be integrated into their security procedures, ITE events must be detected prior to reaching camera #3. These new requirements along with the results from the second verify camera (camera #1 in Figure 3.11) lead us to abandon pedestrian detection and focus our efforts on single camera foreground blob detection methods.

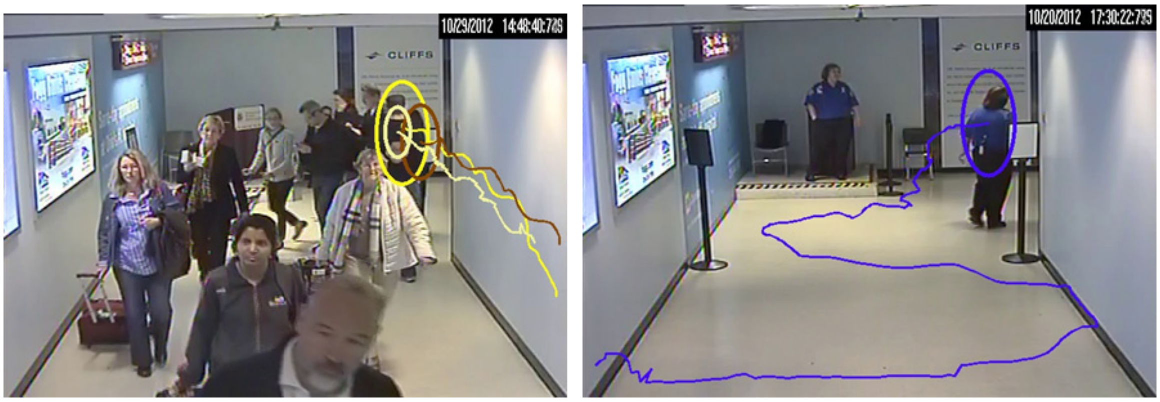


Figure 4.1: Counter-flow Detections using the single camera smart foreground blob tracking approach (see 3.3.3.1)

The single camera smart foreground blob tracking method (see 3.3.3.1) showed extremely promising results from the ITE event video dataset. Figure 4.1 provides examples of this implementation's ability to handle detection in high pedestrian traffic and maintain target identity over long periods of time. The video dataset results showed two main causes of false alarms; miss-tracking from incorrect foreground estimation (see Figure 4.3) and passenger moving luggage or other belongings (see Figure 4.2). When installed in CLE for online testing though, the output logs reported a high number of false alarms. The videos causing these false alarms were brought back to the lab for evaluation. In the lab, we were unable to recreate the vast majority of the false alarms report. This discrepancy between the online and offline results was most likely due our inability to accurately re-create the state of the background model using relatively short videos. These conclusion lead us to reduce our reliance on the foreground calculation and to use online results from CLE, rather than the video dataset, for the final verification of our solutions.

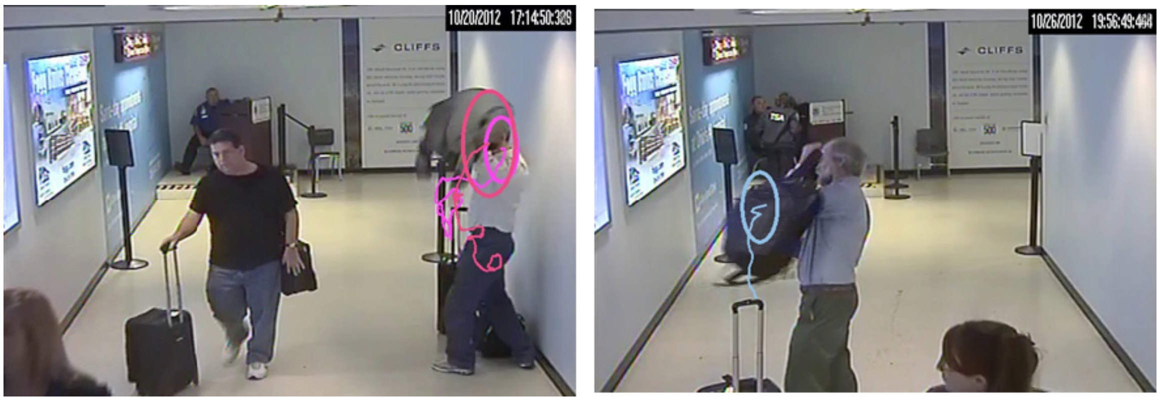


Figure 4.2: False alarm caused by passenger lifting luggage using the single camera smart foreground blob tracking approach (see 3.3.3.1)

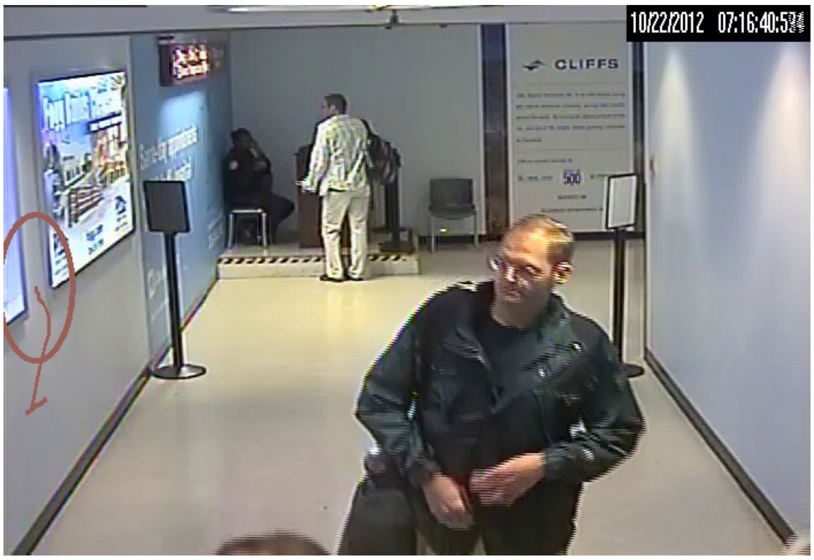
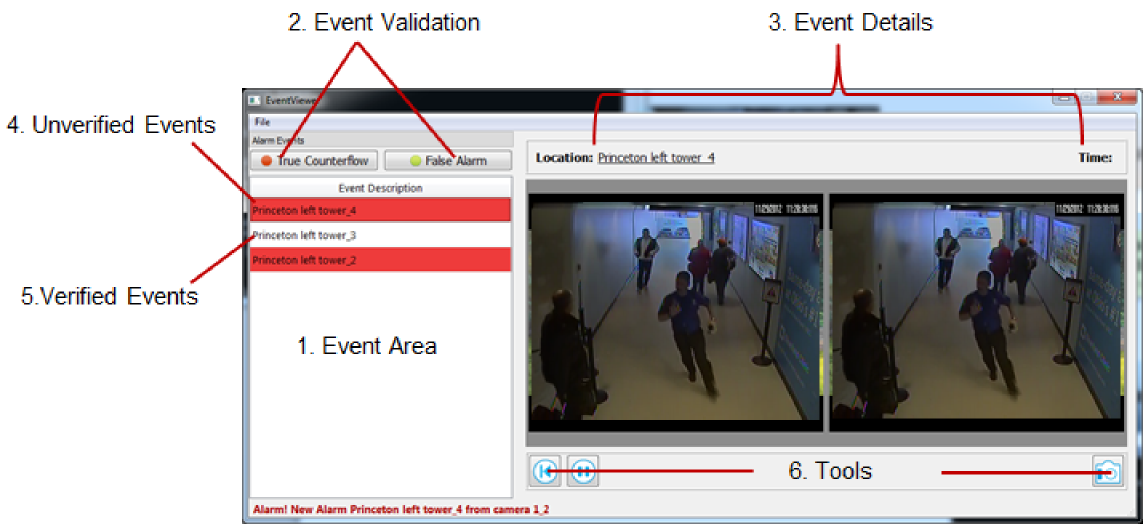


Figure 4.3: False alarm caused by incorrect detection and tracking using the single camera smart foreground blob tracking approach (see 3.3.3.1)

4.2 Prototype Results

To verify the naive over-tracking method (see 3.3.3.2), we installed two versions of the counter-flow detection application in CLE. The first version used extremely sensitive parameters to ensure zero missed detections. Although this version produced a high number of false alarms, the event logs and recorded videos could compared with the output of the second (real) version in order to confirm that no counter-flow events had been missed by our final implementation (the second version). From these results it was determined that of the three counter-flow detection applications developed at each university, only our (Northeastern University) application was able to achieve zero missed detections.



Princeton left tower_4
Princeton left tower_3
Princeton left tower_2

Alarm! New Alarm Princeton left tower_4 from camera 1_2

Figure 4.4: Prototype counter-flow detection application deployed in CLE

The final stage of this project was the creation of a prototype counter-flow detection system that could be installed for observation in CLE. In addition to providing visual and audible cues when counter-flow is detected, the prototype also provides a feedback mechanism for the TSA report on the accuracy of the detection algorithm. Figure 4.4 shows a screenshot of the application's user interface, developed in collaboration with SCR. At the time of publishing, the prototype systems had been operating continuously for four months. It has been prototyped within the TSA's security procedures. Feedback from the TSA has shown zero missed and an acceptably low number of false alarms.

4.3 Future Efforts

The next research challenge facing the ALERT VAST project team is the tracking of tagged individuals as they move throughout the airport. When our ITE application detects the counter-flow, the individual(s) may be tagged for tracking. We hope that when complete this technology will expedite the TSA's response to ITE events, minimizing the fiscal collateral in benign cases and helping security personnel to assess and neutralize any real threats.

4.4 Conclusion

In this thesis, we have presented a number of video analytics solutions to the ITE security challenge. We have detailed the problem and our high level approach to developing and evaluating solutions. This approach breaks down the problem into three functional components that each make use of state of the art computer vision techniques to reliably and efficiently complete their tasks. Although only one of the presented solutions was ultimately utilized in the prototype system deployed in CLE, we feel that the other approaches may be helpful for the development of similar systems in different environments. The lab test results as well as TSA feedback from the real-world performance of our ITE application has not only provided a valuable tool for airport security, but also shown that computer vision and video analytics research is ready and able for surveillance applications. These solutions are not only relevant to academic research communities but could be ready for commercialization in a matter years, not decades. As a whole, the success of the ALERT VAST project has provided an example of how the collaboration of academia, industry, and government can be leveraged to develop novel solutions to real-world problems to a high level of technical readiness.

Bibliography

[1] Science and Technology Directorate Office of University Programs. `http://www.dhs.gov/st-oup`, 2013. Accessed: 2014-08-03.

[2] ALERT DHS Center of Excellence: About. `http://www.northeastern.edu/alert/about/`, 2014. Accessed: 2014-08-03.

[3] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, pages 2544–2550. IEEE, 2010.

[4] G. Bradski. *Dr. Dobb's Journal of Software Tools*.

[5] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[6] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *CVPR*, June 2009.

[7] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *PAMI*, 34, 2012.

[8] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[9] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *proceedings of the European Conference on Computer Vision*, 2012.

[10] Jianbo Shi and Carlo Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

[11] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

[12] Jean yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.

[13] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02*, ICPR '04, pages 28–31, Washington, DC, USA, 2004. IEEE Computer Society.

[14] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recogn. Lett.*, 27(7):773–780, May 2006.