Using Scene Features to Improve Wide-Area Video Surveillance

Ziyan Wu
Rensselaer Polytechnic Institute
Troy, NY
wuz5@rpi.edu

Richard J. Radke
Rensselaer Polytechnic Institute
Troy, NY
rjradke@ecse.rpi.edu

Abstract

We introduce two novel methods to improve the performance of wide area video surveillance applications by using scene features. First, we evaluate the drift in intrinsic and extrinsic parameters for typical pan-tilt-zoom (PTZ) cameras, which stems from accumulated mechanical and random errors after many hours of operation. When the PTZ camera is out of calibration, we show how the pose and internal parameters can be dynamically corrected by matching the scene features in the current image with a precomputed feature library. Experimental results show that the proposed method can keep a PTZ camera calibrated, even over long surveillance sequences. Second, we introduce a classifier to identify scene feature points, which can be used to improve robustness in tracking foreground objects and detect jitter in surveillance videos sequences. We show that the classifier produces improved performance on the problem of detecting counterflow in real surveillance video.

1. Introduction

Scene features — that is, features on stationary background surfaces — are widely used in landmark recognition, image mosaicking, and image understanding. However, in surveillance and object tracking applications, scene features are usually considered to be a distraction from the main goal of detecting changing foreground objects in video sequences. In this paper, we show that scene features can in fact provide useful information in important surveillance scenarios, and focus on two main problems: pan-tilt-zoom (PTZ) camera calibration and counterflow detection.[1]

[1]This material is based upon work supported by the U.S. Department of Homeland Security under Award Number 2008-ST-061-ED0001. 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. Thanks to Edward Hertelendy and Michael Young for supplying the data in Section 4.

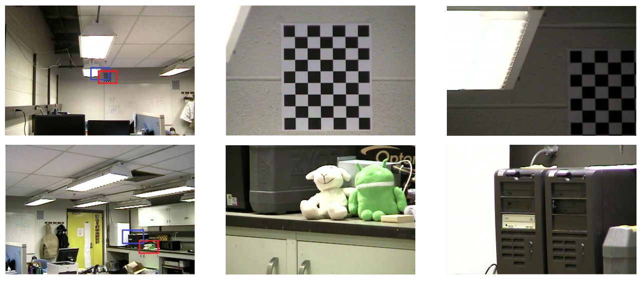


Figure 1. A PTZ camera acquires two images at the same absolute (pan, tilt, zoom) coordinates both before and after 36 hours of continuous random operation. (a) The red rectangles indicate the initial images and the blue rectangles indicate the final images. (b) Close-ups of the acquired initial images. (c) Close-ups of the acquired final images.

Most modern wide-area camera surveillance networks make extensive use of PTZ cameras. However, since such cameras are in constant motion, accumulated errors from imprecise mechanisms, random noise, and power cycling render any calibration in absolute world coordinates useless after many hours of continuous operation. For example, Figure 1 illustrates an example in which a PTZ camera is directed to the same absolute (pan, tilt, zoom) coordinates both before and after 36 hours of continuous operation. We can see that the images are quite different, which means that these absolute coordinates are virtually meaningless in a real-world scenario. However, in practice it is unrealistic to re-calibrate a camera in a busy environment after its mounting. In the first part of this paper, we propose a method based on the automatic detection and matching of scene features to maintain the calibration of a PTZ camera after its initial calibration, so that when a user directs the camera to given (pan, tilt, zoom) coordinates, the same field of view is always attained. Consequently, the absolute PTZ coordinates for a given camera can be trusted to be accurate, leading to improved performance on important tasks like the 3D triangulation of a tracked target.

The second problem we discuss is counterflow detection, a critical problem in security-related surveillance. For example, a person moving the wrong way through the exit corridor of an airport can prompt an entire terminal to be "dumped", resulting in hundreds of delayed flights and inconvenienced passengers. By tracking low-level feature points, the typical flow direction can be easily determined. However, most of the cameras deployed in security surveillance networks have poor resolution and quality compared to a consumer digital camera, which can negatively affect tracking algorithms. In the second part of this paper, we introduce a classifier to identify scene features from the image, which are then used to mitigate cases in which foreground and background features are mixed in the same point trajectory, as well as to identify jitter frames that should not play a role in tracking. We demonstrate that our counterflow detection algorithm is significantly improved by using the scene-feature-based classifier.

2. Related Work

PTZ cameras are traditionally calibrated in controlled environments, using precisely manufactured calibration patterns or active lighting targets. Since PTZ cameras are usually used for video surveillance, self-calibration technologies are often adopted, such as the method with pure rotation proposed by de Agapito et al. [4]. However, a critical issue of PTZ-camera-based video surveillance systems is that even if the camera is initially well-calibrated, after many hours of continuous motion and zooming, accumulated mechanical errors will corrupt both the internal and external parameter estimates (Figure 1). Song and Tai [11] proposed a dynamic calibration method for a PTZ camera based on estimating a vanishing point from a set of parallel lanes with known width. Schoepflin and Dailey [9] proposed a dynamic calibration method with a simplified camera model based on extracting the vanishing point of a roadway. However, these methods are limited to environments featuring reliable straight lines that can be extracted with high precision. In contrast, the method we propose here makes no assumptions about the content of the scene.

The problem of detecting dominant motions in crowded video and classifying outlying motions has been widely studied. For example, Cheriyadat and Radke [3] proposed an automatic dominant motion detection method by clustering trajectories based on longest common subsequences. Since individual people are difficult to segment, the inputs to the algorithm are tracked low-level features obtained using optical flow. Our algorithm takes a similar approach. However, these types of algorithms might not yield good results in situations involving low-resolution cameras and poor image quality. Marcenaro and Vernazza [7] proposed an image stabilization algorithm based on feature tracking in which scene features are used as references to compensate the motion of the camera. In this paper, we propose a classifier to identify scene features in the context of detecting counterflow motion. We show that using information from the scene features, the performance and accuracy of foreground object point tracking can be improved under low-quality, complex-background conditions.

3. Dynamic Correction for PTZ Cameras

In this section, we propose a dynamic correction method to maintain the calibration of a PTZ camera, based on detecting and matching scene features.

3.1. Sources of Error

First, we have to characterize and measure the sources of error in real PTZ cameras. These sources include mechanical offsets in the cameras' stepper motors, random errors in the reported (pan, tilt, zoom) coordinates, accumulated errors in these coordinates that increase with extended continuous operation, and unpredictable jumps in error that occur when the power to the camera is cycled. These types of error combine to make open-loop calibration of PTZ cameras inherently inaccurate, especially at high zoom levels.

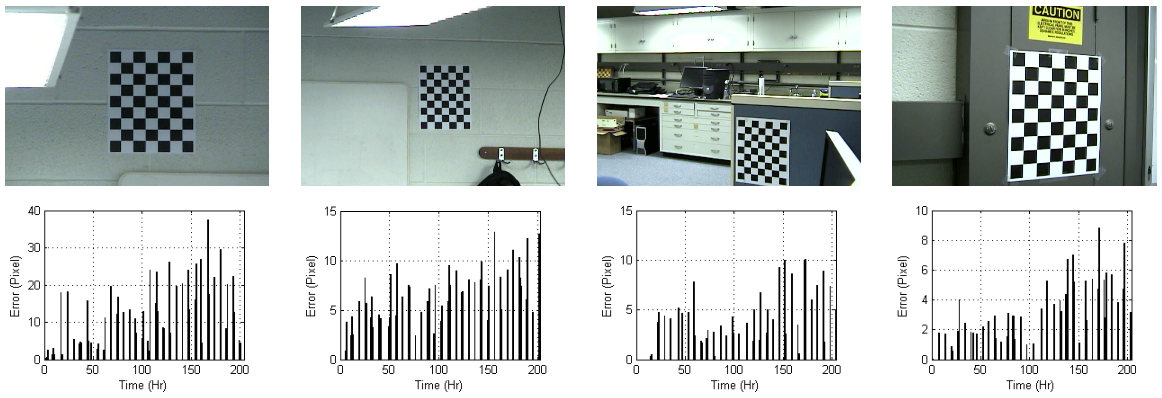
We conducted an experiment to observe the calibration error in a PTZ camera over a long time span (200 hours), shown in Figure 2(a). Printed chessboard targets were placed in four monitored positions in the scene. A PTZ camera (Axis 213) was programmed to randomly move to a new position and zoom level every 30 seconds. After every hour, the camera points to one of the monitored positions and acquires an image, from which the locations of the "x" corners on the target are automatically extracted and compared to the reference image. From the results we can see that the higher the zoom level, the larger the error. A maximum of 38 pixels of error was recorded at position A, the maximum zoom level.

From the results of Figure 2(a), we can see that in addition to random errors, there is a raising trend of accumulated error over time. We were surprised to find out that serious error is induced every time we restart the PTZ camera, as illustrated in Figure 2(b). In positions A and B, the targets have nearly disappeared from the field of view after restarting the camera and pointing it to the same (pan, tilt, zoom) coordinates. This could be a major issue when the PTZ camera is used to monitor an important location like the keypad on a secure door.

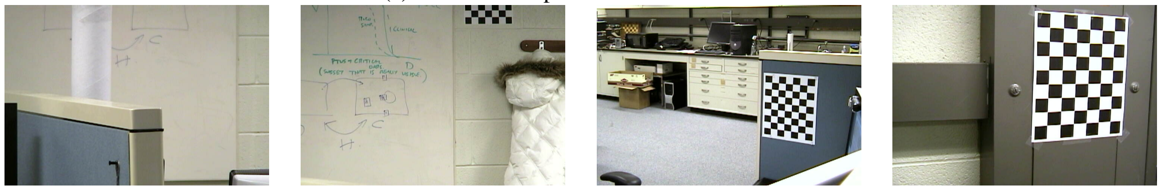
3.2. Camera Model

We assume the standard model for the matrix $\mathbf{K}$ collecting the internal parameters of a camera at a fixed zoom scale, namely

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$



(a) Continuous operation error evaluation


(a) Pos.A (1,0,8000) (b) Pos.B (16,-2,7000) (c) Pos.C (60,-10,1000) (d) Pos.D (-70,-7,0)

(b) Images from the same positions after restarting the camera.

Figure 2. Repeatability experiment.

where f_x, f_y represent the focal length in units of x and y pixels, and (c_x, c_y) is the principal point. We assume the pixel skew is 0 in this paper. The relationship between a 3D point $\mathbf{X}$ and its image projection $\mathbf{x}$ is given by

$$\mathbf{x} \sim \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X} \tag{2}$$

in which $\mathbf{R}$ and $\mathbf{t}$ are the rotation matrix and translation vector respectively.

In order to obtain an accurate image formation model for the wide-angle lenses typically used for PTZ cameras, we must also consider lens distortion. We model the radial distortion using a single-parameter division model (DM) [5], that is,

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} u' \\ v' \\ 1 + \kappa(\widetilde{u'}^2 + \widetilde{v'}^2) \end{bmatrix} = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} + \kappa \begin{bmatrix} 0 \\ 0 \\ \widetilde{u'}^2 + \widetilde{v'}^2 \end{bmatrix} \tag{3}$$

$$\mathbf{x_u} = \frac{\mathbf{x_d}}{1 + \kappa z_d} \tag{4}$$

in which $\mathbf{x_u} = (u, v)$ is the undistorted image coordinate, $\mathbf{x_d} = (u', v')$ is the corresponding distorted image coordinate, $\widetilde{u'} = u' - c_x, \widetilde{v'} = v' - c_y$ and $z_d = \widetilde{u'}^2 + \widetilde{v'}^2$. κ is the distortion coefficient.

3.3. Feature Library for the Scene

The PTZ camera is initially calibrated off-line. Before the PTZ camera begins its regular operation, we construct a feature library in spherical coordinates, as illustrated in Figure 3. To build the library, we have to fix the zoom level at which the camera is calibrated. We then control the camera to sweep through different angles of pan p_o and tilt t_o. For each position, we acquire an image and extract all the SURF features [1] except at the image borders. We store the SURF descriptors and position of each feature in absolute spherical coordinates (p_x, t_x, f), in which f is the calibrated focal length, and p_x, t_x can be obtained by,

$$\begin{aligned} p_x &= p_o + \arctan \frac{u - c_x}{f_x} \\ t_x &= t_o + \arctan \frac{v - c_y}{f_y} \end{aligned} \tag{5}$$

in which (u, v) are the image coordinates of the feature. We merge features that highly overlap in both descriptor and position with those already in the feature library.

Sinha and Pollefeys [10] hypothesized that one could use a calibrated panorama for closed-loop control of a PTZ camera, in which an on-line image taken by the camera is matched to the panorama. This approach may be difficult, since to obtain reasonable accuracy, a large number of high-resolution images would need to be stored and compensated for varying internal parameters. Since our library stores all features in a compact form immediately suitable for descriptor comparison and PTZ parameter estimation, we avoid this problem.

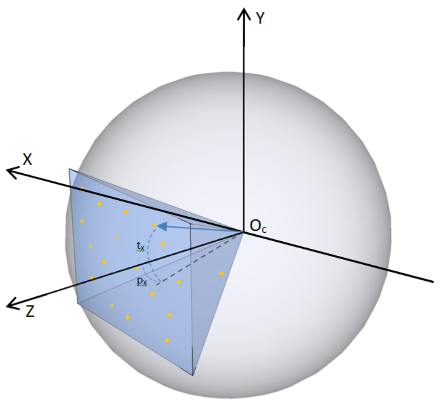


Figure 3. The spherical feature library for a PTZ camera.

3.4. Online Correction

We can safely assume that the principal point of the PTZ camera is fixed over all (pan, tilt, zoom) settings. The parameters we need to correct are the intrinsic parameters, including f_x, f_y, and κ, and the extrinsic parameters, including the pan angle p and tilt angle t. Since the aspect ratio $\alpha = f_x/f_y$ is constant, we only have to correct

$$f = f_x = \alpha f_y \tag{6}$$

When the camera is restarted, or dynamic correction is needed, we assume that the camera is in an unknown setting (p_o, t_o, f_o) that must be accurately estimated. We take a hierarchical approach to estimating these parameters, based on matching the set of SURF features extracted from the current image with the recorded feature library. We match using the usual k-nearest-neighbor approach, and remove outlier matches with 2-parameter RANSAC based on the pan and tilt angles estimated as described below. The basis for estimation is therefore a set of image feature locations $\{(u_i, v_i), i = 1, \ldots, N\}$ and corresponding pose parameter matches from the feature library $\{(p_i^l, t_i^l), i = 1, \ldots, N\}$.

We first obtain a rough initial estimate of (p_o, t_o) from any two matched features i and j. Based on (5) and (6), for match i we have

$$\frac{u_i - c_x}{\tan(p_i^l - p_o)} = \alpha \left(\frac{v_i - c_y}{\tan(t_i^l - t_o)} \right) \tag{7}$$

Similarly, for matches i and j, we have

$$\begin{aligned} \frac{u_i - c_x}{\tan(p_i^l - p_o)} &= \frac{u_j - c_y}{\tan(p_j^l - p_o)} \\ \frac{v_i - c_y}{\tan(t_i^l - t_o)} &= \frac{v_j - c_y}{\tan(t_j^l - t_o)} \end{aligned} \tag{8}$$

We can expand (8) to obtain independent quadratic equations (9) in $\tan p_o$ and $\tan t_o$, which produce our initial estimate. We denote $\widetilde{u_i} = u_i - c_x$ and $\widetilde{v_i} = v_i - c_y$, and (7) can be used to eliminated the false roots. However, this estimate doesn't use all of the feature matches, and more importantly doesn't take into account lens distortion, which is non-negligible for typical PTZ cameras.

$$\begin{aligned} &(\widetilde{u_j} \tan p_j^l - \widetilde{u_i} \tan p_i^l) \tan^2 p_o \\ &\quad + \left[(\widetilde{u_i} - \widetilde{u_j})(\tan p_i^l \tan p_j^l - 1)\right] \tan p_o \\ &\quad + \tan p_j^l \widetilde{u_i} - \tan p_i^l \widetilde{u_j} = 0 \\ &(\widetilde{v_j} \tan t_j^l - \widetilde{v_i} \tan t_i^l) \tan^2 t_o \\ &\quad + \left[(\widetilde{v_i} - \widetilde{v_j})(\tan t_i^l \tan t_j^l - 1)\right] \tan t_o \\ &\quad + \tan t_j^l \widetilde{v_i} - \tan t_i^l \widetilde{v_j} = 0 \end{aligned} \tag{9}$$

To take lens distortion into account, we combine (4) and (7). We denote $z_i = \widetilde{u}_i^2 + \widetilde{v}_i^2$, and define

$$s_i = \frac{\widetilde{u}_i(1 + \kappa z_i)}{\alpha \widetilde{v}_i(1 + \kappa z_i)} = \frac{\widetilde{u}_i}{\alpha \widetilde{v}_i} \tag{10}$$

This suggests the objective function in (11), which operates correctly on the distorted image coordinates but eliminates the unknown coefficient κ. Note that (11) is a nonlinear function in (p_o, t_o), which we minimize using Levenberg-Marquardt optimization.

Now that the pan and tilt angles are determined, we estimate the focal length f (or equivalently the zoom scale) via the distortion coefficient κ. Substituting (4) into (8), we see that every pair of features i and j produces two estimates of κ via

$$\begin{aligned} \frac{\widetilde{u}_i(1 + \kappa z_j)}{\tan(p_i^l - p_o)} &= \frac{\widetilde{u}_j(1 + \kappa z_i)}{\tan(p_j^l - p_o)} \\ \frac{\widetilde{v}_i(1 + \kappa z_j)}{\tan(t_i^l - t_o)} &= \frac{\widetilde{v}_j(1 + \kappa z_i)}{\tan(t_j^l - t_o)} \end{aligned} \tag{12}$$

We can therefore obtain an estimate of κ by averaging the estimates produced by every pair of features. If there are $M = \frac{1}{2}N(N-1)$ such pairs, this corresponds to

$$\begin{aligned} \kappa = \frac{1}{2M} \sum_{i,j} \Bigg[&\frac{\tan(p_i^l - p_o)\widetilde{u}_j - \tan(p_j^l - p_o)\widetilde{u}_i}{\tan(p_j^l - p_o)\widetilde{u}_i z_j - \tan(p_i^l - p_o)\widetilde{u}_j z_i} \\ &+ \frac{\tan(t_i^l - t_o)\widetilde{v}_j - \tan(t_j^l - t_o)\widetilde{v}_i}{\tan(t_j^l - t_o)\widetilde{v}_i z_j - \tan(t_i^l - t_o)\widetilde{v}_j z_i} \Bigg] \end{aligned} \tag{13}$$

The focal length f_o can then be recovered by

$$f_o = \frac{1}{2N} \sum_{i=1}^{N} \left[\frac{\widetilde{u}_i \tan(t_i^l - t_o) + \alpha \widetilde{v}_i \tan(p_i^l - p_o)}{(1 + \kappa z_i) \tan(p_i^l - p_o) \tan(t_i^l - t_o)} \right] \tag{14}$$

Finally, in order to reduce the influence of noise, we perform a nonlinear optimization over all the parameters at

$$F_1(p_o, t_o) = \left\| \begin{bmatrix} s_1 \tan p_1^l \tan t_1^l + 1 & s_1 - \tan p_1^l \tan t_1^l & \tan t_1^l - s_1 \tan p_1^l \\ s_2 \tan p_2^l \tan t_2^l + 1 & s_2 - \tan p_2^l \tan t_2^l & \tan t_2^l - s_2 \tan p_2^l \\ \cdots & \cdots & \cdots \\ s_n \tan p_n^l \tan t_n^l + 1 & s_n - \tan p_n^l \tan t_n^l & \tan t_n^l - s_n \tan p_n^l \end{bmatrix} \begin{bmatrix} \tan p_o \\ \tan t_o \\ \tan p_o \tan t_o \end{bmatrix} - \begin{bmatrix} \tan p_1^l - s_1 \tan t_1^l \\ \tan p_2^l - s_2 \tan t_2^l \\ \cdots \\ \tan p_n^l - s_n \tan t_n^l \end{bmatrix} \right\| \quad (11)$$

once. That is, starting from the estimates of (p_o, t_o, κ, f_o) obtained above, we minimize the sum of squared differences between the observed feature locations and their positions predicted by the model, namely the objective function

$$F_2(p_o, t_o, f_o, \kappa) = \sum_{i=1}^{N} \left\| \begin{matrix} f_o \tan(p_i^l - p_o) - \widetilde{u}_i \frac{1}{1+\kappa z_i} \\ f_o \tan(t_i^l - t_o) \frac{1}{\alpha} - \widetilde{v}_i \frac{1}{1+\kappa z_i} \end{matrix} \right\|^2 \quad (15)$$

Thus, we can automatically compensate the inaccurate (pan, tilt, zoom) parameters reported by the camera to the parameters (p_o, t_o, f_o) obtained by minimizing (15), which are consistent with the original calibration of the camera.

3.5. Experiments

Figure 4 shows an example of the feature library and feature matching process in a real PTZ camera installation. From Figure 4(b) we can see that there are a number of initial mismatches (e.g., similar corners on checkerboard targets), which are correctly removed by RANSAC in Figure 4(c). Figure 4(d) shows the algorithm works well in correcting the substantial error induced after restarting the camera and directing it to the "same" position.

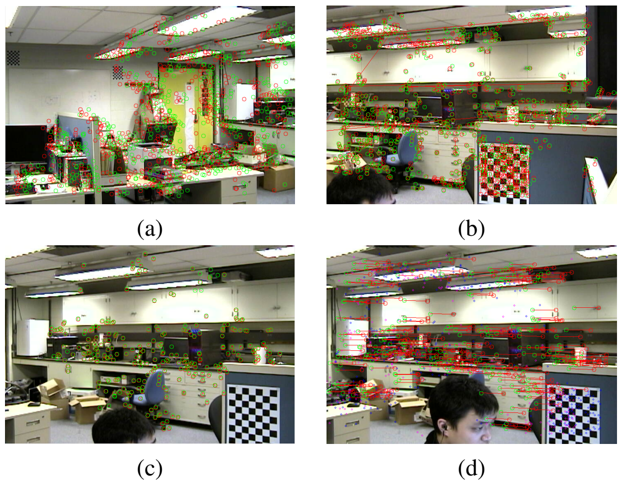

(a) (b) (c) (d)

Figure 4. Building the feature library and feature matching. (a) Features found in two images acquired at roughly the same position. (b) Matching with KNN. (c) Matching result after RANSAC. (d) Matching result after restarting the camera.

We conducted the same experiment as in Figure 1 after incorporating the dynamic correction method, as illustrated in Figure 5. We can see that the targets of interest are successfully located, which suggests that the corrected pose model can substantially improve the usefulness of PTZ cameras in video surveillance.

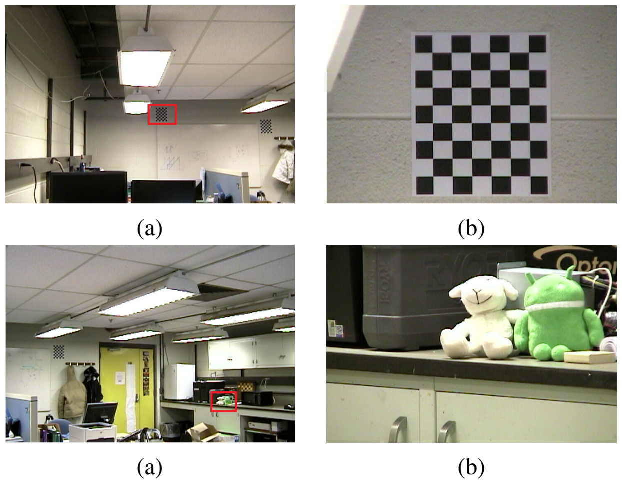
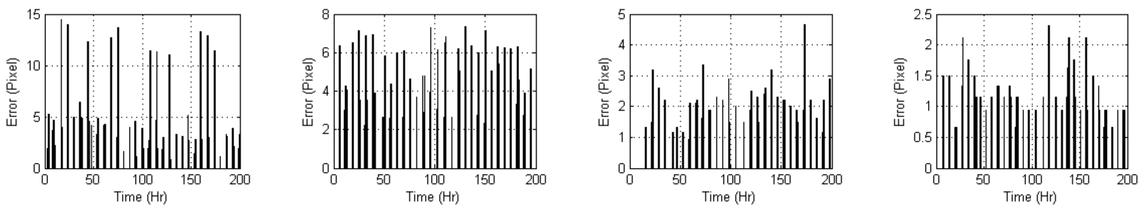

(a) (b) (a) (b)

Figure 5. The red rectangle indicates the desired image border of the zoomed camera while the blue rectangle indicates the actual image border of the zoomed camera. (a) Image at original position. (b) Zoomed image.

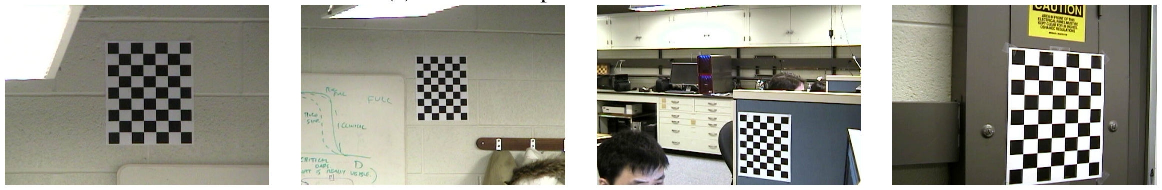
In order to evaluate the effects of dynamic correction on accumulated error, we conducted the same experiment as in Figure 2 after incorporating the dynamic correction method, as illustrated in Figure 6. We can see that the error is reduced for all monitored positions. Furthermore, there is no increasing trend of error over time. Even after restarting the camera, all four monitored positions can be well-recovered using dynamic correction.

4. Scene Features in Counterflow Detection

Counterflow detection is an important problem in security surveillance applications, such as one-way passages from secure areas to non-secure areas in airports. Figure 7 illustrates several such passages acquired by the camera network in Terminals A and B of the Cleveland airport. In this part of the paper, we describe a solution to detecting counterflow in a fixed camera that leverages the detection of scene features to improve performance.



(a)Continuous operation error evaluation



(a) Pos.A (1,0,8000) (b) Pos.B (16,-2,7000) (c) Pos.C (60,-10,1000) (d) Pos.D (-70,-7,0)

(b)Images from the same positions after restarting the camera.

Figure 6. Repeatability experiment using dynamic correction.

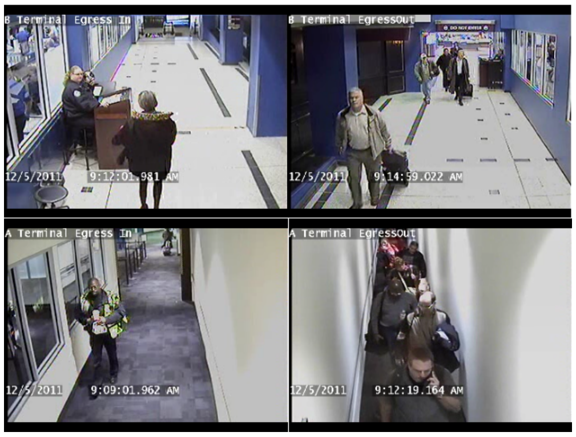


Figure 7. Sample images from the camera network at CLE.

4.1. Feature Tracking

Videos from surveillance camera networks are frequently low-resolution (e.g., 352×240). Since we want the system to process video streams from tens of cameras in real time, and the dominant (or allowable) direction of motion is all we need to know, we use low-level features to track the flow. We first identify low-level features in the initial frame using the FAST corner detector [8]. The features are then tracked over time using the Kanade-Lucas-Tomasi (KLT) optical flow algorithm [6], adapting the pyramid representation in [2], which can track large pixel motions while keeping the size of the integration window relatively small.

The results of feature tracking are shown in Figure 8, in which red circles indicate all the features detected in the current frame (up to a maximum number, e.g., 300) and blue circles indicate reliably-trackable features. New features are added to the tracker in every 5-10 frames, discarding those too close to current tracks. These feature tracks form a large trajectory set.

4.2. Improving Robustness with Scene Features

This low-level feature point tracking is often inaccurate, due to both the low resolution and quality of the input videos and periodic jitter. Consequently, it is common for features on foreground objects (corresponding to the allowable/counter flow) to mix or merge with stationary scene features, as illustrated in Figure 9.

Our solution to this problem is to build a three-way classifier to classify normal flow, counterflow, and scene features. The point tracks are classified at a specified interval (e.g., every 300 frames). The recognized scene features can also be used to compensate for location drift caused by jitter.

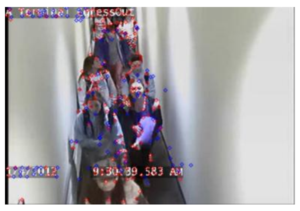
Let $\mathbf{L}^j = \{(x_j(1), y_j(1)), \ldots, (x_j(n_j), y_j(n_j))\}$ be the data of the j^{th} point track. We define two features (d_1^j, d_2^j) for each trajectory $\mathbf{L}^j$ as

$$d_1^j = \frac{1}{n_j^2} \sum_{i=1}^{\lfloor \frac{n_j}{3} \rfloor} y_j(n_j - i) - y_j(i)$$

$$d_2^j = \frac{1}{n_j} \sqrt{\sum_{i=2}^{n_j} (x_j(i) - x_j(0))^2 + (y_j(i) - y_j(0))^2}$$

That is, d_1^j represents the difference in sum on y between the first third and last third of the trajectory, and d_2^j represents the variance of the points on the trajectory from their initial position.

As Figure 10 illustrates, the three-way classifier separates trajectories corresponding to normal flow, counterflow, and scene features based on the rule


(a)

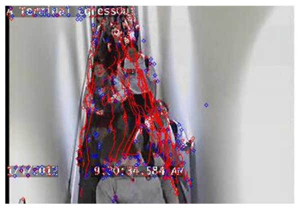

(b)

Figure 8. Results of feature tracking. (a) Features detected in the image. (b) Point tracks extracted from a video sequence.

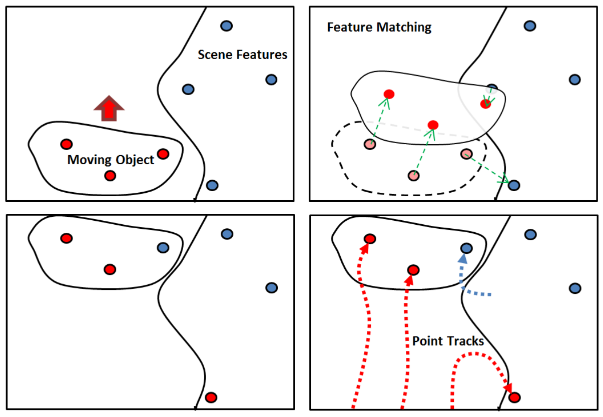
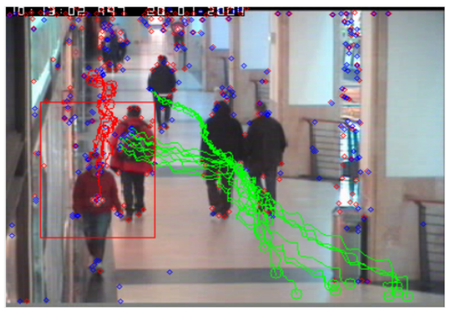


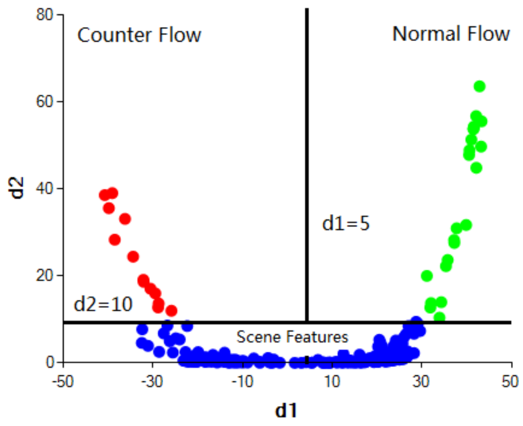
Figure 9. Foreground points mixing with scene points.

$$\mathbf{L}^j = \begin{cases} \text{normal flow} & d_1^j > a, d_2^j > b \\ \text{counterflow} & d_1^j \le a, d_2^j > b \\ \text{scene feature} & d_2^j \le b \end{cases} \tag{16}$$

The value of b in the classifier to separate scene points is relatively easy to set (we use 10 in our experiments). The value of a is trained on a short sequence based on user editing of missed detections and false alarms. That is, a is set to an initial value (e.g., 5) and is adjusted based on user edits to the smallest number such that the classifier has no missed detections (which are operationally extremely costly).

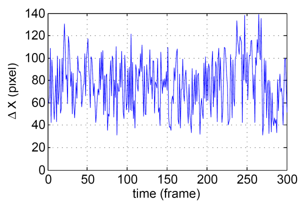
After the scene features are classified, they can be used to deal with two issues. First, point tracks that were classified as scene points in the previous decision are matched and tracked only after all of the other (flow) features are matched and tracked for each frame. This step significantly reduces the probability that scene features mix with moving features and confuse the tracker/classifier, as we show in the next section. Second, the statistics of scene features provide an easy way to detect frames with jitter, as shown in Figure 11. We can easily learn a threshold on the change in x values along the trajectories for scene points that detects jitter frames. These frames are then ignored for the purposes of tracking and classification, which substantially improves robustness.


(a)

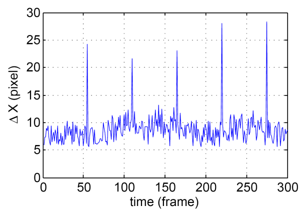


(b)

Figure 10. Result of the three-way classifier. (a) Feature tracks. (b) Classifier result corresponding to (a).



(a)

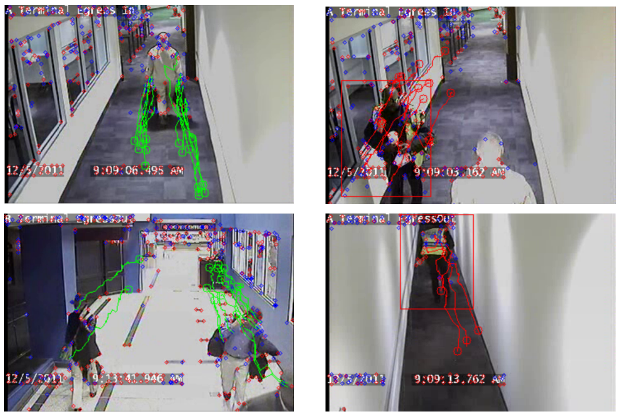


(b)

Figure 11. Statistics of (a) moving points and (b) scene feature points. Jitter frames are clearly visible as spikes in (b).

4.3. Experiments

We tested the counterflow detection algorithm on six video sequences from the camera network at the Cleveland Airport. Several examples are shown in Figure 12. Normal flows are displayed in green while counterflows are displayed in red. A bounding box corresponding to each suspicious target is also created. The classifier is first trained with a small portion of the video and tested with the rest. As a baseline, we compare the results against a classifier that only discriminates between normal flow and counterflow using the d_1 feature (i.e., not taking into account scene



(a) (b)

Figure 12. Sample flow classification results. (a) Normal flow. (b) Counterflow is detected and the target is located.

features). The results for the six sequences are collected in Table 1.

Video	Length (min)	TT (min)	GT	2-Class TP	2-Class FA	3-Class TP	3-Class FA
Ter.A Eg.In	40	3	1	1	2	1	0
Ter.A Eg.Out	32	3	2	2	6	2	1
Ter.B Eg.In	40	3	10	8	4	10	1
Ter.B Eg.Out	32	3	10	6	12	10	6
Ter.C Eg.In	5	1	0	1	3	0	0
Ter.C Eg.Out	5	1	2	0	5	2	3
Total	154	14	25	18	32	25	11

Table 1. Results of the counterflow experiment. **GT** denotes the number of ground truth counterflows, **TT** the training time, **TP** the number of true positives and **FA** the number of false alarms.

As desired, the algorithms successfully detected all the counterflow occurrences in all the sequences without error. Most of the false alarms are due to noise, jitter or mixing point tracks. Since false alarms are easy to correct and clear, this number seems acceptable in practical applications given the lengths of the videos involved. The last video is particularly challenging due to mixing feature tracks, a complex background, and passengers coming from an exit far from the camera. Both the false alarm rate and true positive rate are improved by the proposed scene-feature-based algorithms.

5. Conclusion

We proposed two algorithms relevant to wide-area surveillance with camera networks. First, we presented a dynamic correction method for keeping a PTZ camera calibrated. The lightweight, unsupervised method compares scene features in the current image against a feature library created at the time of mounting to ensure that the user always gets the same field of view when directing the camera to given (pan, tilt, zoom) coordinates. We also introduced a classifier that simultaneously identifies scene feature trajectories and detects counterflow motion. Explicitly including scene features in the flow classifier improves robustness, reduces the mixing of foreground and background in point tracks, and allows the detection of jitter frames.

Future work includes a complete self-calibration method for PTZ cameras based on scene features; dynamic, optimal selection of good features to track in both the background scene and foreground objects; and more robust algorithms to improve tracking and classification results using scene features.

References

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[2] J.-Y. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm. Technical report, Microprocessor Research Labs, Intel Corporation, 2002.

[3] A. Cheriyadat and R. Radke. Detecting Dominant Motions in Dense Crowds. *IEEE Journal of Selected Topics in Signal Processing*, 2(4):568–581, Aug. 2008.

[4] L. de Agapito, E. Hayman, and I. Reid. Self-Calibration of Rotating and Zooming Cameras. *International Journal of Computer Vision*, 45(2):107–127, Nov. 2001.

[5] A. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *CVPR*, 2001.

[6] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[7] L. Marcenaro and G. Vernazza. Image stabilization algorithms for video-surveillance applications. In *ICIP*, 2001.

[8] E. Rosten, R. Porter, and T. Drummond. Faster and better: a machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–19, Jan. 2010.

[9] T. Schoepflin and D. Dailey. Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, 4(2):90–98, June 2003.

[10] S. N. Sinha and M. Pollefeys. Pan-tilt-zoom camera calibration and high-resolution mosaic generation. *Computer Vision and Image Understanding*, 103(3):170–183, Sept. 2006.

[11] K.-T. Song and J.-C. Tai. Dynamic calibration of Pan-Tilt-Zoom cameras for traffic monitoring. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, 36(5):1091–103, Oct. 2006.