VIDEO-BASED MOTION DETECTION FOR STATIONARY AND MOVING CAMERAS

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

RUI WANG

Dr. Kannappan Palaniappan, Thesis Supervisor

JULY 2014

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled:

VIDEO-BASED MOTION DETECTION FOR
STATIONARY AND MOVING CAMERAS

presented by Rui Wang,
a candidate for the degree of Master of Science and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Kannappan Palaniappan

Dr. Filiz Bunyak

Dr. Tony Xu Han

ACKNOWLEDGMENTS

I would like to thank Professor Kannappan Palaniappan my committee chair, who has the attitude and the substance of a genius. Without his guidance and persistent help this thesis would not have been possible.

I would like to express my deepest appreciation to Professor Filiz Bunyak, my committee member, for her valuable and constructive suggestions during the planning and development of this thesis. Her patient guidance and enthusiastic encouragement helps me continually making progress.

Algorithms and discussion in Section 4.2 is developed with the assistance of Dr. Hadi Aliakbarpour and Dr. Adel Hafiane. Dr Hadi implemented the RANSAC homography estimation method for moving camera video stabilization that extends the UAV video frame-to-frame registration algorithm previously developed by Adel Hafiane for frame-to-frame matching. The flux tensor implementation (in Matlab and C++) used in this thesis was developed by Filiz Bunyak and Ilker Ersoy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In real world monitoring applications, moving object detection remains to be a challenging task due to factors such as background clutter and motion, illumination variations, weather conditions, noise, and occlusions. As a fundamental first step in many computer vision applications such as object tracking, behavior understanding, object or event recognition, and automated video surveillance, various motion detection algorithms have been developed ranging from simple approaches to more sophisticated ones. In this thesis, we present two moving object detection frameworks. The first framework is designed for robust detection of moving and static objects in videos acquired from stationary cameras. This method exploits the benefits of fusing a motion computation method based on spatio-temporal tensor formulation, a novel foreground and background modeling scheme, and a multi-cue appearance comparison. This hybrid system can handle challenges such as shadows, illumination changes, dynamic background, stopped and removed objects. Extensive testing performed on the CVPR 2014 Change Detection benchmark dataset shows that FTSG outperforms most state-of-the-art methods among 16 algorithms. The second framework adapts moving object detection to full motion videos acquired from moving airborne platforms. The video is stabilized with respect to a set of base-frames in the sequence. The stabilization is done by estimating four-point homographies using prominent feature (PF) block matching, motion filtering and RANSAC for robust matching. Once the frame to base frame homographies are available the flux tensor motion detection module using local second derivative information is applied to detect moving salient features. Spurious responses from the frame boundaries are removed and other post- processing operations are applied to reduce the false alarms and produce accurate moving blob regions that will

be useful for tracking.

Chapter 1

Introduction

In real world monitoring applications, moving object detection remains to be a challenging task due to factors such as background complexity, illumination variations, noise and occlusions. As a fundamental first step in object tracking, behavior understanding, object or event recognition, automated video surveillance, etc. various motion detection algorithms have been developed ranging from simple approaches to more sophisticated ones.

Moving object detection approaches can be categorized into three broad classes as background subtraction, temporal differencing and optical flow methods. *Optical flow* methods,[cite] first compute optical flow vectors then group the flow vectors that belong to the same motion to identify moving objects. They can thus be used even with non-stationary cameras. However reliable motion field computation under real-world conditions is challenging and computationally expensive and these methods can not deal with stopped objects. *Temporal differencing* based methods rely on pixel-wise differences between consecutive frames. They do not maintain a background model, are simple and fast, and can quickly adapt to different changes thus are suitable for dynamic backgrounds, illumination changes, uncovered background by removed objects etc. However, without an explicit background model, temporal differencing cannot detect slow moving or stopped objects and often result in foreground aperture problem and fail to detect parts of objects (particularly large objects with homogeneous interiors result in holes). *Background subtraction* based methods maintain and use an explicit background model thus can handle slow moving or stopped objects and do not suffer from foreground aperture problem. However they are sensitive to dynamic scene changes due to illumination changes, uncovered background by removed objects etc. Even with adaptive multi-modal background modeling mechanisms [3], adaptation capability of background subtraction based methods is still worse than temporal difference based methods.

The main contributions of this paper are: (i) A motion computation method based on flux-tensor our spatio-temporal tensor formulation. (ii) A novel Split-Gaussian method to separately model foreground and background. (iii) A robust multi-cue appearance comparison module to remove false detections due to illumination changes, shadows etc. and to differentiate stopped objects from revealed background by removed objects.

Regarding the performance, our method can handle shadow, illumination changes, ghosts, stopped or removed objects, some dynamic background and camera jitter while still maintaining a fast boot-strapping. Our method outperforms most well known techniques on moving object detection. As of submission date of this paper, our results outrank submissions to CVPR 2012 change detection [2] at the six category-wide overall rankings.

1.1 Related Work

Moving object detection approaches can be categorized into three broad classes as optical flow methods, temporal differencing and background subtraction. *Optical flow* methods [] first compute optical flow vectors, then group the flow vectors that belong to the same motion to identify moving objects. They can thus be used even with non-stationary cameras. However reliable motion field computation under real-world conditions is a challenging and computationally expensive task and these methods can not deal with stopped objects. *Temporal differencing* based methods rely on pixel-wise differences between consecutive frames. Since they do not maintain a background model, they are simple and fast. Temporal differencing methods can quickly adapt to dynamic environments such as illumination changes, revealed background by removed objects etc. However, without an explicit background model, temporal differencing cannot detect slow moving or stopped objects and often results in foreground aperture problem (particularly large objects with homogeneous interiors result in holes). *Background subtraction* based methods maintain and use an explicit background model thus can handle slow moving or stopped objects and do not suffer from foreground aperture problem. However they are sensitive to dynamic scene changes due to illumination changes, revealed background by removed objects etc. Even with adaptive multi-modal background modeling mechanisms (e.g. [3]), adaptation capability of background subtraction based methods tends to be not as good as temporal difference based methods. The focus of this paper are the two latter categories. A comprehensive survey of these approaches can be found in [4]. A more contemporary evaluation of particularly background subtraction methods can be found in [5, 6].

Simple temporal differencing methods as surveyed in [4] rely on pixel-wise differencing and global thresholds. These methods are sensitive to noise and illumination changes, and often produce incomplete object masks. Various extensions have been proposed to improve performance of temporal differencing methods. Hsu et al [7] incorporate spatial information by fitting a polynomial function to each pixel block. Change detection is then performed by thresholding the variance of the residuals from the polynomial fit to corresponding blocks in two images. Mahadevan et al. [8] motivated by biological vision propose use of discriminant center-surround spatiotemporal saliency. For each pixel, center, surround, and temporal windows are created. A saliency map is generated using center-surround classification based on previously obtained distributions. Locations with large saliency are classified as foreground. Despite the high computational cost, this method is robust to dynamic scenes.

While moving object detection using temporal differencing is efficient and adaptive, real world foreground detection tasks usually involves not only moving objects but also slow moving and stopped objects (sleeping person in [9]), that are hard or impossible to detect by temporal differencing. Background subtraction methods handle these cases by explicitly maintaining a model of the background. Many background subtraction methods operate at pixel level and model each pixel independently. Gaussian models are widely adopted by these methods. In the simplest case, pixel history is modeled using a single Gaussian [10]. Dominant values in the pixel history are assumed to belong to background pixels. Pixel values that are different from the mean by some multiple of the standard deviation are classified as foreground. Adaptive version of this approach updates model parameters making this simple Gaussian method adaptive to gradual illumination changes. Because foreground pixels are identified not through motion or change in consecutive frames, but rather by difference from a background model, these models can handle slow moving and stopped objects. By extending single Gaussian model to mixture of Gaussians (MoG), more complex scenes such as dynamic backgrounds can be handled [3, 11]. In [3] Stauffer and Grimsom present the popular adaptive background mixture models method where the history of a pixel is modeled by a mixture of K Gaussians. An online K-means clustering is used to update the parameters of the K Gaussians for each pixel. Number of Gaussian distributions K is an important parameter and if not selected properly can lead to under- or over-fitting problems. In [1] Haines & Xiang propose a new background subtraction approach with Dirichlet processes, and use a non-parametric Bayesian method to estimate the number of Gaussian distributions required to model the background pixels. Besides widely used intensity and color features, use of gradient and texture information helps in noise and illumination handling. [12–14] incorporate spatial consistency to background subtraction by using LBP (local binary pattern) to produce a feature vector for each pixel. LBP in background subtraction is first used in [12]. In [13], a spatiotemporal local binary pattern feature is used to model dynamic backgrounds. [14] further extends [12] to a shadow invariant local binary pattern descriptor called scale invariant local ternary pattern. All these methods are using a simple adaptive filter to constantly update their LBP based background models.

Beside parametric background modeling approaches, there are also many nonparametric background modeling methods. Elgammal et al. [15] propose a nonparametric background modeling approach where kernel density estimation is used to model N samples in recent pixel history. This can be seen as a generalized Gaussian mixture model where each sample set is considered as a Gaussian model. Selection of window size is crucial since it can lead the model to adapt too quickly or too slowly. The paper uses double models to handle quick background changes while still preserving a relative static background model. A simple intersection operation is used on the detection results to combine the models. In [16] Nonaka et al.incorporate spatial consistency to nonparametric background model. The approach combines nonparametric pixel level model which is a Parzen density model, together with a region level model, which is built by using Radial Reach Correlation. In [17] Li et al. incorporate spectral, spatial, and temporal features using principle feature representation and uses Bayes classification to identify background and foreground. An online K-mean clustering similar to [3] is used to update the prior and posterior probabilities of the principle features. In [18] Tsai et al. propose an Independent Component Analysis based method where a demixing matrix obtained from a training set is used for foreground vs. background classification. This method does not have an update mechanism, thus it is more suitable for indoor applications where continuous background change is limited. But lack of update mechanism help detection of stopped foreground objects since they do not get integrated into the background model. Instead of computing a probability density function or model ViBe [19] directly stores a collection of N background samples as its background model. Two main differences between Vibe and conventional background subtraction methods are, first, that an incoming pixel only needs to be similar to some of the samples in the background model rather than the majority of all values, and second, it is not necessary to discard an old background value. Based on these two ideas, a pixel is classified as background if it is similar to some of the background samples and the background model is updated by replacing a randomly selected sample from the background samples with the current pixel value. The paper also points out that the traditional Mixture of Gaussians (MoG) methods cannot handle stopped objects because they use a blind update strategy where all pixels are blindly updated into the background model. So they use a conservative update strategy where the background model is only updated when a pixel is classified as background. Conservative update suffers from deadlock problem where temporary detection errors become permanent ghosts. Based on the assumption that neighboring pixels share the same temporal distributions, at the model update stage, Vibe not only updates a randomly selected background sample, but also updates a randomly selected neighbor pixel. So ghosts resulting from removed objects dissolve in time starting from their boundaries. Random neighbor update reduces stopped and removed object problems, but temporary ghosts for removed object still occur and stopped objects still blend into the background due to random noise. The selection of number of background samples, learning rate for background updating and threshold for foreground classification is critical. In [20] Hofmann et al. propose up an adaptive thresholding method to adaptively select learning rate and decision threshold based on how dynamic the background is.

By using multiple models and adaptive filters, some motion detection problems such as dynamic backgrounds and illumination changes can be solved. However, to handle more complex problems such as stopped objects or revealed background by removed objects, simple statistic models are not sufficient. Therefore some background subtraction methods use higher-level analysis to classify these cases. [21] combines temporal differencing and background subtraction to first identify stationary and transient pixels, then a region level analysis classify regions into moving or stopped. However, this system cannot differentiate stopped objects from revealed background from removed objects. [22] also uses a combination of temporal differencing and background subtraction at pixel level. At background model update stage region based analysis is performed. Pixels with same intensity values are grouped together and update is performed group by group. In this way, revealed background will be recovered quickly. In [23] Evangelio et al. use finite state machine and edge-based analysis to identify stopped objects and revealed background. Two Mixture of Gaussians models are used one with high learning rate called short-term background model as a moving object detector and one with low learning rate called long-term background model to reconstruct the static background. A finite state machine is used on the results of these two models to further classify foreground pixels as stationary foreground or moving foreground. Then a group of stationary foreground pixels is classified as a new static object or as revealed background by using edge-based analysis.

Chapter 2

Motion Detection Using a Flux Tensor

2.1 Flux Tensor Framework

Motion blob detection is performed using our novel *flux tensor* method which is an extension to 3D grayscale structure tensor [24]. Both the grayscale structure tensor and the proposed flux tensor use spatio-temporal consistency more efficiently, thus produce less noisy and more spatially coherent motion segmentation results compared to classical optical flow methods [25]. The flux tensor is more efficient in comparison to the 3D grayscale structure tensor since motion information is more directly incorporated in the flux calculation which is less expensive than computing eigenvalue decompositions as with the 3D grayscale structure tensor.

2.1.1 3D Structure Tensors

Structure tensors are a matrix representation of partial derivative information. As they allow both orientation estimation and image structure analysis they have many applications in image processing and computer vision. 2D structure tensors have been widely used in edge/corner detection and texture analysis, 3D structure tensors have been used in low-level motion estimation and segmentation [25, 26].

Under the constant illumination model, the optic-flow (OF) equation of a spatiotemporal image volume $\mathbf{I}(\mathbf{x})$ centered at location $\mathbf{x} = [x, y, t]$ is given by Eq. 2.1 [27] where, $\mathbf{v}(\mathbf{x}) = [v_x, v_y, v_t]$ is the optic-flow vector at $\mathbf{x}$, I doing like this.

$$\begin{aligned} \frac{d\mathbf{I}(\mathbf{x})}{dt} &= \frac{\partial \mathbf{I}(\mathbf{x})}{\partial x}\, v_x + \frac{\partial \mathbf{I}(\mathbf{x})}{\partial y}\, v_y + \frac{\partial \mathbf{I}(\mathbf{x})}{\partial t}\, v_t \\ &= \nabla \mathbf{I}^T(\mathbf{x})\ \mathbf{v}(\mathbf{x}) = 0 \end{aligned} \tag{2.1}$$

and $\mathbf{v}(\mathbf{x})$ is estimated by minimizing Eq. 2.1 over a local 3D image patch $\mathbf{\Omega}(\mathbf{x}, \mathbf{y})$, centered at $\mathbf{x}$. Note that v_t is not 1 since spatio-temporal orientation vectors will be computed. Using Lagrange multipliers, a corresponding error functional $e_{ls}(\mathbf{x})$ to minimize Eq. 2.1 using a least-squares error measure can be written as Eq. 2.2 where $W(\mathbf{x}, \mathbf{y})$ is a *spatially invariant* weighting function (e.g., Gaussian) that emphasizes the image gradients near the central pixel [26].

$$\begin{aligned} e_{ls}(\mathbf{x}) = \int_{\mathbf{\Omega}(\mathbf{x},\mathbf{y})} & \left(\nabla \mathbf{I}^T(\mathbf{y})\ \mathbf{v}(\mathbf{x})\right)^2\ W(\mathbf{x}, \mathbf{y})\ d\mathbf{y} \\ & + \lambda \left(1 - \mathbf{v}(\mathbf{x})^T \mathbf{v}(\mathbf{x})\right) \end{aligned} \tag{2.2}$$

Assuming a constant $\mathbf{v}(\mathbf{x})$ within the neighborhood $\mathbf{\Omega}(\mathbf{x}, \mathbf{y})$ and differentiating $e_{ls}(\mathbf{x})$ to find the minimum, leads to the standard eigenvalue problem (Eq. 2.3) for solving $\hat{\mathbf{v}}(\mathbf{x})$ the best estimate of $\mathbf{v}(\mathbf{x})$,

$$\mathbf{J}(\mathbf{x}, \mathbf{W})\ \hat{\mathbf{v}}(\mathbf{x}) = \lambda\ \hat{\mathbf{v}}(\mathbf{x}) \tag{2.3}$$

The 3D structure tensor matrix $\mathbf{J}(\mathbf{x}, \mathbf{W})$ for the spatiotemporal volume centered at $\mathbf{x}$ can be written in expanded matrix form, without the spatial filter $W(\mathbf{x}, \mathbf{y})$ and the positional terms shown for clarity, as Eq. 2.4.

$$\mathbf{J} = \begin{bmatrix} \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial x}\frac{\partial \mathbf{I}}{\partial x} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial x}\frac{\partial \mathbf{I}}{\partial y} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial x}\frac{\partial \mathbf{I}}{\partial t} d\mathbf{y} \\ \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial y}\frac{\partial \mathbf{I}}{\partial x} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial y}\frac{\partial \mathbf{I}}{\partial y} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial y}\frac{\partial \mathbf{I}}{\partial t} d\mathbf{y} \\ \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial t}\frac{\partial \mathbf{I}}{\partial x} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial t}\frac{\partial \mathbf{I}}{\partial y} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial t}\frac{\partial \mathbf{I}}{\partial t} d\mathbf{y} \end{bmatrix} \tag{2.4}$$

A typical approach in motion detection is to threshold $\mathbf{trace}(\mathbf{J})$ (Eq. 2.5); but this results in ambiguities in distinguishing responses arising from stationary versus moving features (e.g., edges and junctions with and without motion), since $\mathbf{trace}(\mathbf{J})$ incorporates total gradient change information but fails to capture the nature of these gradient changes (i.e. spatial only versus temporal).

$$\mathbf{trace}(\mathbf{J}) = \int_{\mathbf{\Omega}} ||\nabla I||^2 d\mathbf{y} \tag{2.5}$$

To resolve this ambiguity and to classify the video regions experiencing motion, the eigenvalues and the associated eigenvectors of $\mathbf{J}$ are usually analyzed [28, 29]. However eigenvalue decompositions at every pixel is computationally expensive especially if real time performance is required.

2.1.2 Flux Tensors

In order to reliably detect only the moving structures *without* performing expensive eigenvalue decompositions, the concept of the *flux tensor* is proposed. Flux tensor is the temporal variations of the optical flow field within the local 3D spatiotemporal volume. Computing the second derivative of Eq. 2.1 with respect to t, Eq. 2.6 is obtained where, $\mathbf{a}(\mathbf{x}) = [a_x, a_y, a_t]$ is the acceleration of the image brightness located at $\mathbf{x}$.

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{d\mathbf{I}(\mathbf{x})}{dt}\right) = {} & \frac{\partial^2 \mathbf{I}(\mathbf{x})}{\partial x \partial t} v_x + \frac{\partial^2 \mathbf{I}(\mathbf{x})}{\partial y \partial t} v_y + \frac{\partial^2 \mathbf{I}(\mathbf{x})}{\partial t^2} v_t \\ & + \frac{\partial \mathbf{I}(\mathbf{x})}{\partial x} a_x + \frac{\partial \mathbf{I}(\mathbf{x})}{\partial y} a_y + \frac{\partial \mathbf{I}(\mathbf{x})}{\partial t} a_t \end{aligned} \tag{2.6}$$

which can be written in vector notation as,

$$\frac{\partial}{\partial t}(\nabla \mathbf{I}^T(x)\mathbf{v}(\mathbf{x})) = \frac{\partial \nabla \mathbf{I}^T(\mathbf{x})}{\partial t}\, \mathbf{v}(\mathbf{x}) + \nabla \mathbf{I}^T(\mathbf{x})\, \mathbf{a}(\mathbf{x}) \tag{2.7}$$

Using the same approach for deriving the classic 3D structure, minimizing Eq. 2.6 assuming a constant velocity model and subject to the normalization constraint $||\mathbf{v}(\mathbf{x})|| = 1$ leads to Eq. 2.8,

$$\begin{aligned} e_{ls}^F(\mathbf{x}) = {} & \int_{\mathbf{\Omega}(\mathbf{x},\mathbf{y})} \left(\frac{\partial(\nabla \mathbf{I}^T(\mathbf{y})}{\partial t}\, \mathbf{v}(\mathbf{x})\right)^2 W(\mathbf{x},\mathbf{y})\, d\mathbf{y} \\ & + \lambda\left(1 - \mathbf{v}(\mathbf{x})^T \mathbf{v}(\mathbf{x})\right) \end{aligned} \tag{2.8}$$

Assuming a constant velocity model in the neighborhood $\mathbf{\Omega}(\mathbf{x},\mathbf{y})$, results in the acceleration experienced by the brightness pattern in the neighborhood $\mathbf{\Omega}(\mathbf{x},\mathbf{y})$ to be zero at every pixel. As with its 3D structure tensor counterpart $\mathbf{J}$ in Eq. 2.4, the 3D flux tensor $\mathbf{J_F}$ using

Eq. 2.8 can be written as

$$\mathbf{J_F}(\mathbf{x}, \mathbf{W}) = \int_{\mathbf{\Omega}} W(\mathbf{x}, \mathbf{y}) \frac{\partial}{\partial t} \nabla \mathbf{I}(\mathbf{x}) \cdot \frac{\partial}{\partial t} \nabla \mathbf{I^T}(\mathbf{x}) d\mathbf{y} \tag{2.9}$$

and in expanded matrix form as Eq. 2.10.

$$\mathbf{J_F} = \begin{bmatrix} \int_{\mathbf{\Omega}} \left\{\frac{\partial^2 \mathbf{I}}{\partial x \partial t}\right\}^2 d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial^2 \mathbf{I}}{\partial x \partial t} \frac{\partial^2 \mathbf{I}}{\partial y \partial t} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial^2 \mathbf{I}}{\partial x \partial t} \frac{\partial^2 \mathbf{I}}{\partial t^2} d\mathbf{y} \\ \int_{\mathbf{\Omega}} \frac{\partial^2 \mathbf{I}}{\partial y \partial t} \frac{\partial^2 \mathbf{I}}{\partial x \partial t} \, d\mathbf{y} & \int_{\mathbf{\Omega}} \left\{\frac{\partial^2 \mathbf{I}}{\partial y \partial t}\right\}^2 d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial^2 \mathbf{I}}{\partial y \partial t} \frac{\partial^2 \mathbf{I}}{\partial t^2} d\mathbf{y} \\ \int_{\mathbf{\Omega}} \frac{\partial^2 \mathbf{I}}{\partial t^2} \frac{\partial^2 \mathbf{I}}{\partial x \partial t} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial^2 \mathbf{I}}{\partial t^2} \frac{\partial^2 \mathbf{I}}{\partial y \partial t} d\mathbf{y} & \int_{\mathbf{\Omega}} \left\{\frac{\partial^2 \mathbf{I}}{\partial t^2}\right\}^2 d\mathbf{y} \end{bmatrix} \tag{2.10}$$

As seen from Eq. 2.10, the elements of the flux tensor incorporate information about temporal gradient changes which leads to efficient discrimination between stationary and moving image features. Thus the trace of the flux tensor matrix which can be compactly written and computed as,

$$\mathbf{trace}(\mathbf{J_F}) = \int_{\mathbf{\Omega}} || \frac{\partial}{\partial t} \nabla \mathbf{I} ||^2 d\mathbf{y} \tag{2.11}$$

and can be directly used to classify moving and non-moving regions without the need for expensive eigenvalue decompositions. If motion vectors are needed then Eq. 2.8 can be minimized to get $\hat{\mathbf{v}}(\mathbf{x})$ using

$$\mathbf{J_F}(\mathbf{x}, \mathbf{W}) \ \hat{\mathbf{v}}(\mathbf{x}) = \lambda \ \hat{\mathbf{v}}(\mathbf{x}) \tag{2.12}$$

In this approach the eigenvectors need to be calculated at just moving feature points.

2.1.3 Flux Tensor Implementation

To detect motion blobs, only the trace of flux tensor $\mathbf{trace}(\mathbf{J_F}) = \int_{\mathbf{\Omega}(\mathbf{y})} ||\frac{\partial}{\partial t}\nabla\mathbf{I}||^2 d\mathbf{y}$ needs to be computed. That requires computation of I_{xt}, I_{yt} and I_{tt} and the integration of squares of I_{xt}, I_{yt}, I_{tt} over the area $\mathbf{\Omega}(\mathbf{y})$. The following notation is adopted for simplicity:

$$I_{xt} = \frac{\partial^2 I}{\partial x \partial t}, \qquad I_{yt} = \frac{\partial^2 I}{\partial y \partial t}, \qquad I_{tt} = \frac{\partial^2 I}{\partial^2 t} \tag{2.13}$$

The calculation of the derivatives is implemented as convolutions with a filter kernel. By using separable filters, the convolutions are decomposed into a cascade of 1D convolutions. For numerical stability as well as noise reduction, a smoothing filter is applied to the dimensions that are not convolved with a derivative filter e.g. calculation of I_{xt} requires smoothing in y-direction, and calculation of I_{yt} requires smoothing in x-direction. I_{tt} is the second derivative in temporal direction; the smoothing is applied in both spatial directions. As smoothing and derivative filters, optimized filter sets presented by Scharr et. al. in [30, 31] are used.

The integration is also implemented as an averaging filter decomposed into three 1D filters. As a result, calculation of trace at each pixel location requires three 1D convolutions for derivatives and three 1D convolutions for averages in the corresponding spatio-temporal cubes.

A brute-force implementation where spatial and temporal filters are applied for each pixel separately within a spatio-temporal neighborhood would be computationally very expensive since it would have to recalculate the convolutions for neighboring pixels. For an efficient implementation, the spatial (x and y) convolutions are separated from the temporal convolutions, and the 1D convolutions are applied to the whole frames one at a time. This minimizes the redundancy of computations and allows reuse of intermediate results. Steady state operation (i.e. after FIFO is full) of this process is summarized in the following paragraph.

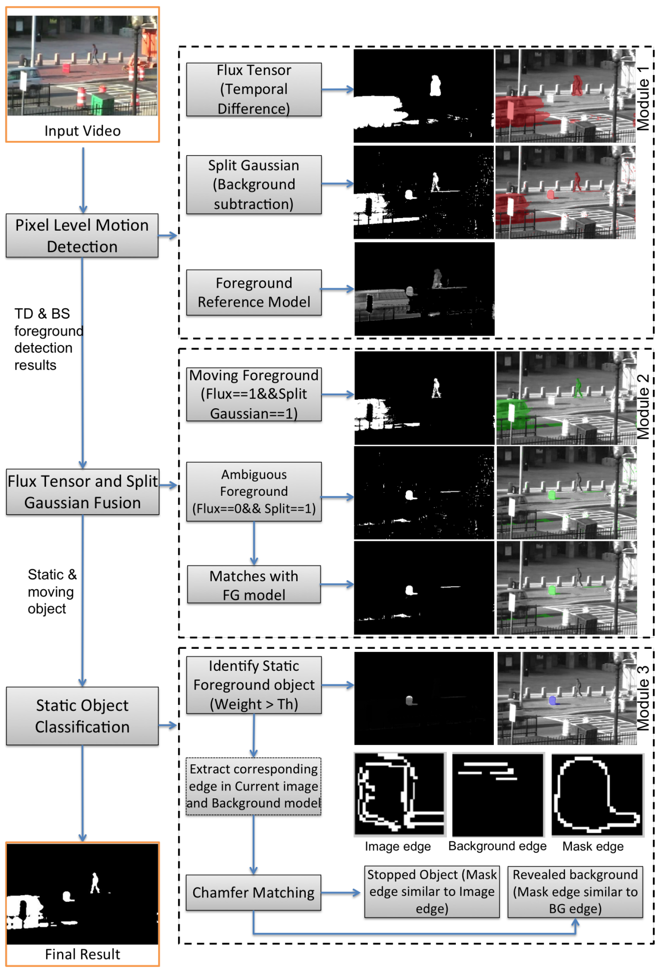
The spatial convolutions required to calculate I_{xt}, I_{yt} and I_{tt} are I_{xs} , I_{sy} and I_{ss} where s represents the smoothing filter. Each frame of the input sequence is first convolved with two 1D filters, either a derivative filter in one direction and a smoothing filter in the other direction, or a smoothing filter in both directions. These intermediate results are stored as frames to be used in temporal convolutions, and pointers to these frames are stored in a First In First Out (FIFO) buffer of size $n_{FIFO} = n_{Dt} + n_{At} - 1$ where n_{Dt} is the length of the temporal derivative filter and n_{At} is the length of the temporal averaging filter. For each input frame, three frames I_{xs} , I_{sy} and I_{ss} are calculated and stored. Once n_{Dt} frames are processed and stored, FIFO has enough frames for the calculation of the temporal derivatives I_{xt} , I_{yt} and I_{tt}. Since averaging is distributive over addition, $I_{xt}^2 + I_{yt}^2 + I_{tt}^2$ is computed first and spatial averaging is applied to this result and stored in the FIFO structure to be used in the temporal part of averaging. Once flux tensor trace of n_{At} frames are computed, temporal averaging is applied. Motion mask $\mathrm{FG_M}$ is obtained by thresholding and post-processing averaged flux tensor trace. Post-processing include morphological operations to join fragmented objects and to fill holes.

Chapter 3

Static and Moving Object Detection Using Flux Tensor with Split Gaussian Models

3.1 System overview

The task for motion detection is to detect and segment foreground regions from a sequence of images. Most change detection algorithms treat this as a binary classification problem based on either frame differencing or pre-established background model. However, for real monitoring tasks, we usually need to deal with more complex situations such as stopped or slow moving foreground objects, which cannot be classified based on temporal differencing or illumination changes and revealed background regions by removed objects, which may be misclassified as foreground by background subtraction . In this paper, we introduce a novel hybrid system named Flux Tensor with Split Gaussian Model (FTSG) that combines our motion computation method based on a spatio-temporal tensor formulation and a novel



Input Video
Pixel Level Motion Detection
TD & BS foreground detection results
Flux Tensor and Split Gaussian Fusion
Static & moving object
Static Object Classification
Final Result
Flux Tensor (Temporal Difference)
Split Gaussian (Background subtraction)
Foreground Reference Model
Module 1
Moving Foreground (Flux==1&&Split Gaussian==1)
Ambiguous Foreground (Flux==0&& Split==1)
Matches with FG model
Module 2
Identify Static Foreground object (Weight > Th)
Module 3
Extract corresponding edge in Current image and Background model
Image edge
Background edge
Mask edge
Chamfer Matching
Stopped Object (Mask edge similar to Image edge)
Revealed background (Mask edge similar to BG edge)

Figure 3.1: Flux with Split Gaussian models system flow. The system is categorized into three modules. Module 1 is pixel level motion detection that two complementary change detectors run separately on input image and ouput foreground detection results. Module 2 fuse the detection results from module 1, which gets rid of noise, illumination changes and halo effects. Module 3 is a high level classification that distinguish removed objects from stopped objects. Edges of the static objects in foreground detection mask are compared to the edges of the corresponding object in current image and background model using chamfer matcing.

foreground/background modeling scheme.

These two complementary methods are combined to form a robust moving object detector , FTSG, that can correctly classify static background, background with illumination changes, revealed background by removed objects, moving foreground objects and stopped foreground objects. Figure3.1 shows the system flow of our proposed FTSG method .

There are three main modules in FTSG which are pixel level detection, decision fusion, object level classification.

a) Two binary masks of foreground/background classification results are obtained using TD and BS methods separately in pixel level motion detection.

b) In Flux Tensor and Spit Gaussian modeling methods fusion module, pixels classified as foreground by both methods are kept as real moving foreground. Pixels classified as foreground by Split Gaussian modeling only are considered to be potential static foreground regions and are compared to foreground reference model. Matched pixels are kept as foreground, whereas unmatched pixels are illumination changes on background region or noise.

c) In stopped and removed objects classification, static objects are extract from previously fused mask. An edge based classification method is used to distinguish stopped objects from removed objects.

Detailed descriptions of each component are in the following sections.

3.1.1 Split Gaussian Models

Barnich *et al.* [19] pointed out that three fundamental considerations of a background subtraction algorithm are: 1) What is the model and how does it behave? 2) how to initialize the model? and 3) how to maintain the model? Ideally, a successful background subtraction method should be able to initialize in a few frames and without any prior knowledge of the clean background. Besides, its update mechanism should allow it to adept to environment changing and does not blend foreground information into the background model. More importantly, background and foreground should be separated perfectly by using the model. Most background subtraction methods such as [3, 15, 17, 32] needs a sequence of training images but there are some methods such as [19, 23] that have the one-frame initialization capability. And except only a few background subtraction methods such as ICA [18] that do not update overtime, most background subtraction methods adaptively update their background model. However, perfectly classify background and foreground regions using pre-established background model is almost impossible because no matter how good the background model is, changes in background such as sudden illumination changes, cloud moving and random noises that could not find a match in background model will be classified as foreground. On the other hand, if the background model is adjustd to decrease these false positives then background model will likely start to model foreground too resulting in increased false negatives (missing foreground). As we mentioned previously, this is because background subtraction method treat such classification problem as a binary classification problem, whereas the problem incorperates many sub-classes such as static vs. dynamic background, static vs moving foreground, illumination changes, noise etc. Some of these challenges are handled by our later modules, fusion and classification. Therefore, in order to obtain better classification result, our background subtraction model is designed to capture complementary information of Flux Tensor. In this section, we introduce a background subtraction method named Split Gaussian models that has fast boost-trapping, adaptive updating and complex background environment modeling capabilities.

Pixel model and Classification

Gaussian models have been widely used in background subtraction methods. Mixture of Gaussians can efficiently represent multimodal signals, which makes them suitable for background modeling and subtraction. We adopt mixture of Gaussians as our background model. However, unlike MoG in [3] where background and foreground are blended together into a single model with fixed number of Gaussians, we model foreground and background separately, and use adaptively changing number of Gaussians for the background model. This simplifies the background/foreground classification step, prevents background model from being corrupted by foreground pixels, and also provides better adaptation for different background types (static vs. dynamic backgrounds). This approach has fast bootstrapping, adaptive updating and complex background environment modeling capabilities.

Background model: We use a mixture of K Gaussians to model the background where K is a spatially and temporally adaptive variable. Every new pixel value, $I_t(x, y)$, is checked against the existing K Gaussian distributions. A match to a Gaussian is defined as pixel values within T_b standard deviations of the mean :

$$D_{min}(x, y) = \min_{i \in K} \max_{j \in C}((\mathbf{I}_t(x, y) - \mu_{i,j})^2 - T_b \cdot \sigma^2) \tag{3.1}$$

A pixel is labeled as foreground if it does not match any of the Gaussians in the background model:

$$\mathbf{F}_B(x, y) = \begin{cases} 1, & if\ \mathbf{D}_{min}(x, y) > 0 \\ 0, & otherwise \end{cases} \tag{3.2}$$

T_b is a fixed threshold and stands for number of standard deviations, and $\sigma = \sum_i^k \omega_i \sigma_i$. For each pixel, there will be $K \times C$ Gaussian models where C is the number of channels, e.g. 3 for RGB. For simplicity, all the channels share the same variance σ and weight ω.

Foreground appearance model: We use a single Gaussian to model the foreground. Foreground appearance model (shown in Figure 3.1, module 1) is used to distinguish static foreground (stopped object and revealed background) from spurious detections due to illumination changes and noise within ambiguous regions, $\mathbf{F}_{amb}(x, y)$ where $\mathbf{F}_F = 0$ and $\mathbf{F}_B = 1$ (detected as background by flux but as foreground by background subtraction shown as *ambiguous foreground* in Figure 3.1 module 2). Static foreground regions $\mathbf{F}_S$ are identified within ambiguous detections $\mathbf{F}_{amb}$ using foreground model:

$$\mathbf{F}_S(x, y) = \begin{cases} 1, \quad if \qquad \mathbf{F}_{amb}(x, y) = 1 \, and \\ \qquad\qquad \mathbf{I}_t(x, y) - \mu_f(x, y) < T_f \\ 0, \qquad\qquad otherwise \end{cases} \tag{3.3}$$

Model initialization:

Many background subtraction methods need a sequence of frames to initialize their models such as [3], [17]. However, using a sequence of images to train a background model is not appropriate if the image sequence itself is already very short. It is also very difficult to restart the program when there is a drastic background change. On the contrary, temporal difference based methods does not have such problems because they don't need to maintain a model of empty background scene. If we already have a well-defined temporal difference based method that does not have initialization problem, can we take advantage of it and make the background subtraction method can also be initialized in a few frames? The answer is yes. Our split Gaussian models have the capability to be initialized in a few frames and the length depends on the temporal window size of Flux Tensor.

Since, flux tensor provides motion information, and the fusion and classification modules greatly reduce false positives. Therefore, the background model can be directly initialized using the first few frames and the foreground appearance model can be initialized to be empty.

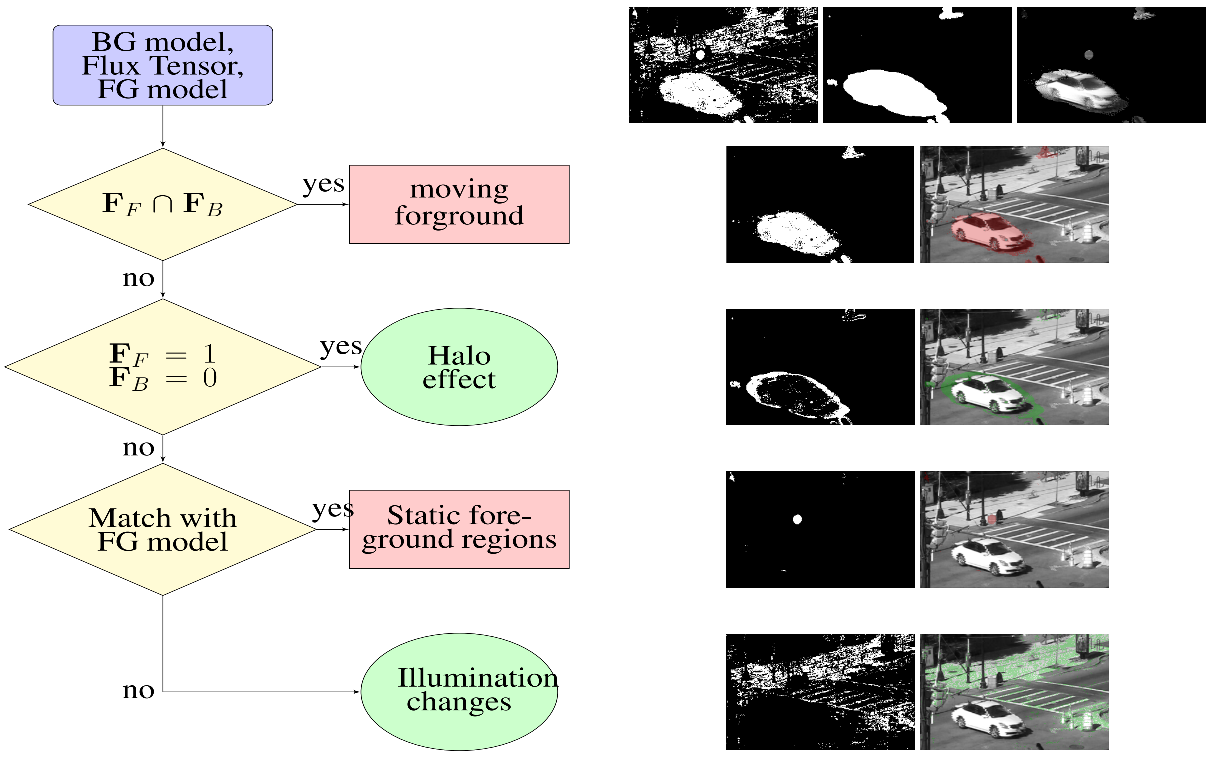
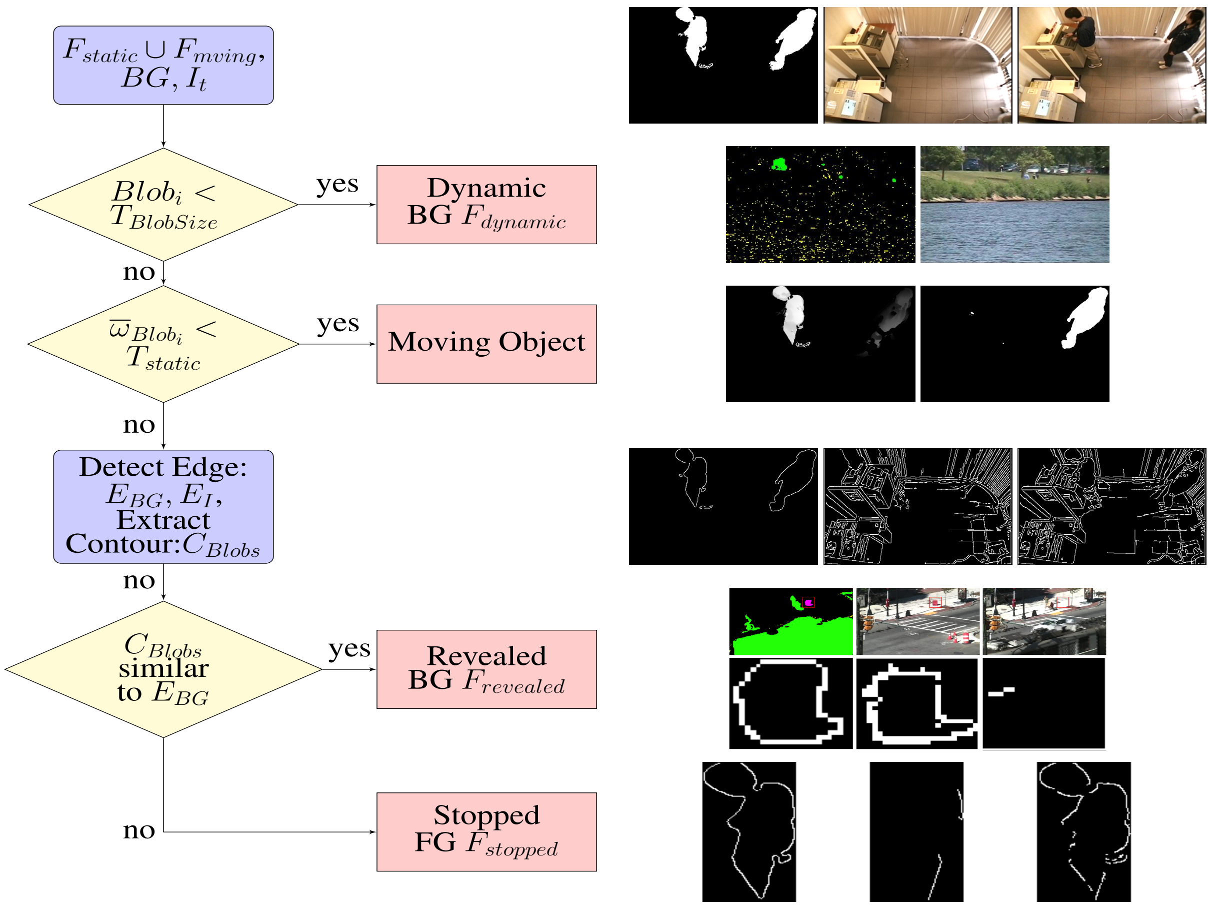

BG model, Flux Tensor, FG model
$\mathbf{F}_F \cap \mathbf{F}_B$
yes
moving forground
no
$\mathbf{F}_F = 1$ $\mathbf{F}_B = 0$
yes
Halo effect
no
Match with FG model
yes
Static foreground regions
no
Illumination changes

Figure 3.2: Fusion of flux tensor and split Gaussian models. Images on the right hand side are corresponding to those elements in the flowchart on the left hand side. $\mathbf{F}_F$, $\mathbf{F}_B$ stand for flux tensor motion segmentation mask and split Gaussian background subtraction mask respectively.

3.2 Fusion of Flux Tensor and Split Gaussian Models

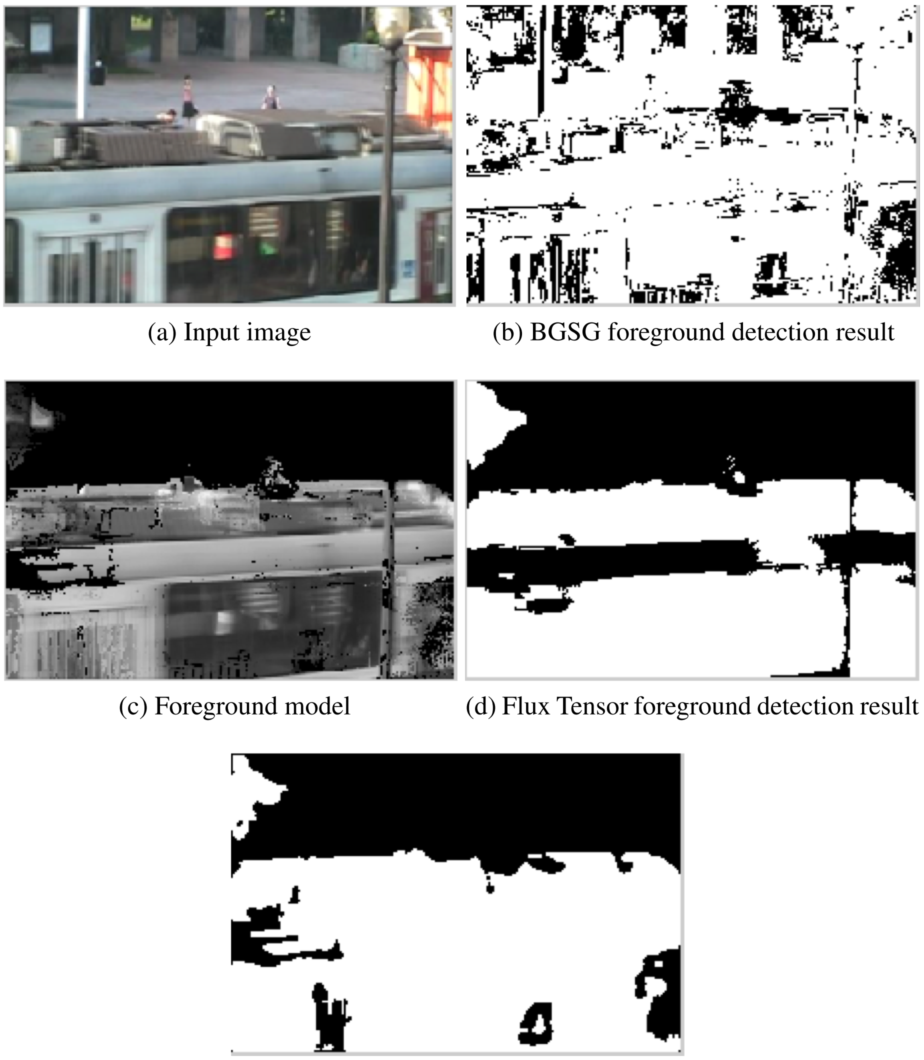
The goal of this decision fusion module is to exploit complementary information from two inherently different approaches to boost overall detection accuracy. Flux tensor based motion segmentation produces spatially coherent results due to spatio-temporal integration. These results are also robust to illumination changes and soft shadows due to use of gradient based information. But since the method relies on motion, it fails to detect stopped foreground objects and tends to produce masks larger than the objects. Background subtraction on the other hand can detect stopped objects, but is sensitive to noise, illumination changes and shadows. Here we extend flux tensor based motion segmentation with split



$F_{static} \cup F_{mving}$, BG, I_t
$Blob_i < T_{BlobSize}$
yes
Dynamic BG $F_{dynamic}$
no
$\overline{\omega}_{Blob_i} < T_{static}$
yes
Moving Object
no
Detect Edge: E_{BG}, E_I, Extract Contour:C_{Blobs}
no
C_{Blobs} similar to E_{BG}
yes
Revealed BG $F_{revealed}$
no
Stopped FG $F_{stopped}$

Figure 3.3: Fusion of flux tensor and split Gaussian models. Images on the right hand side are corresponding to those elements in the flowchart on the left hand side. $\mathbf{F}_F$, $\mathbf{F}_B$ stand for flux tensor motion segmentation mask and split Gaussian background subtraction mask respectively.

Gaussian foreground and background models to generate a more complete and accurate foreground object detection method.


(a) Input image (b) BGSG foreground detection result

(c) Foreground model (d) Flux Tensor foreground detection result

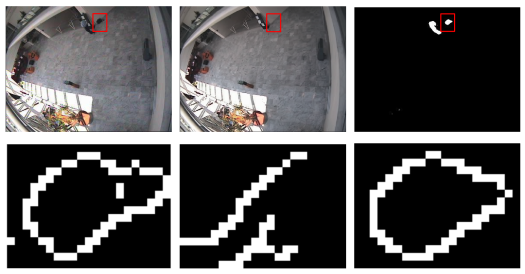
(e) Fusing result of Flux Tensor and BGSG

Figure 3.4: Light reflectance from the moving train causes suddent illumination changes in the scene as shown in (a). (b) shows that BGSG detects massive illumination changes as foreground. (d) shows that Flux Tensor is robust to illumination changes but detection result has holes inside object. By referencing foreground model (b), fused foreground detection result (e) eliminated illumination and fills hole inside foreground object.

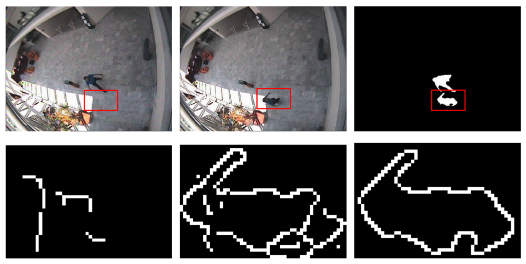
Figure 3.3 shows fusion flow chart and some examples of flux tensor and split Gaussian model fusion results. Pixels that are detected as foreground by both flux tensor and split

Gaussian background subtraction are classified as moving foreground objects. Pixels that are detected as foreground by background subtraction only and have a match in foreground model correspond to static foreground objects and those do not find a match in foreground model correspond to illumination changes. Figure 3.4 shows an example of fusion of flux tensor and split Gaussian models that handles illumination changes.

3.3 Stopped and Removed Object Classification



(a) Stopped object



(b) Revealed background by removed object

Figure 3.5: Classification of stopped objects vs. background revealed by removed objects. Images on the first row from left to right are current image, background model and foreground mask. Images on the second row are edge maps corresponding to the regions of interest marked by red rectangle in the images of the first row.

Fusion procedure classifies both stopped objects (true positives) and revealed background by removed objects (false positives) as static foreground. Distinguishing these two types of static foreground can effectively reduce the false positive rate and tackle deadlock problem. The method used for removed and stopped objects classification is based on [23], which basically has three steps: 1. Identify pixels corresponding to static regions; 2. Perform edge detection on static regions in current image, background generated by background subtraction and foreground detection mask; 3. Perform classification based on edge matching.

Static regions can easily be extracted by measuring the average weight of each connected component of foreground detection result in foreground model. The weight value in foreground model implies how recently a pixel is updated. For static foreground region, weight values will accumulate quickly so static foreground can be obtained by thresholding the average weight value of each connected component in foreground detection result. In addition, it is not necessary to classify those static regions that have already been correctly classified again, so static regions that have average weights larger then a pre-determined threshold will be classified as stopped foreground object directly.

We use canny edge detector to obtain edge maps in input image E_{in}, background scene subtract from BG-SG E_{MoG}. Edges of foreground detection mask are the boundaries of each connected components E_{mask}.

The classification of removed and stopped object is based on the intuition that the edge of stopped object in E_{mask} will be similar to E_{in} in the same location as shown in Figure 3.5 (a). Similarly, the edge of removed object in E_{mask} will be similar to E_{MoG} in the same area as shown in Figure 3.5 (b). In [2] chamfer matching is used directly to compare E_{mask} with E_{in} and E_{MoG}. However, since E_{mask} is only a closed boundary and the other two edge maps may contain complex interior edges or vague boundary edges around target object due to camouflage, it is very likely that chamfer matching will match E_{mask} to some interior edges in either E_{in} or E_{MoG} instead of the real object boundary. So instead of getting edge map of a whole region, we only extract edges that around the boundary of the detect region. This is done by using dilate and erode operation on the static region mask to get a ring shape region, as shown in figure. Then the edges only correspond to the ring like region in the input frame and background model are obtained. Therefore, matching E_{mask} to some interior complex edges is avoided and camouflage is also tolerated to some extend.

Figure 3.5 a, b show classification examples for stopped object (an abandoned bag) and revealed background by removed object (ghost effect due to background model initialization) respectively. Stopped object has higher edge similarity between current image and foreground mask, while revealed background by removed object has higher edge similarity between background model and foreground mask.

3.4 Model Update and Parameter Tuning

As we mentioned previously, an ideal update method should allow the background model to adapt to environment changes and at the same time not include foreground information into background model. Common update scheme can be divide into blind update and conservative update. Blind update, such as in GMM, includes all sample values into the background model, which will blend foreground information into background. So we use conservative update policy, pixels classified as foreground will never be updated into background model, in our update scheme. However, conservative update suffers from deadlock problem that wrongly classified background pixels will leave permanent ghost in detection results and prevent background model from updating. Some algorithm such as Vibe use spatial consistency, pixels classified as background will also update its randomly selected neighbor pixel models, to handle such deadlock problem. SACON [33] insert groups of pixels into background model if they are classified as foreground for sufficient long time. These methods all handled deadlock problem but will produce some false positive and false negative detections. We solved such deadlock problem by directly increase the detection accuracy, which is obviously a more desirable way.

Type	Flux Tensor (Temporal differencing)	SG-modeling (Background subtraction)	Similarity to foreground model	Edge based classification	Detection result
Moving object	$\surd$	$\surd$	*NA*	*NA*	FG
Stopped object	X	$\surd$	$\surd$	$\surd$	FG
Revealed backgroud	X	$\surd$	$\surd$	X	BG
Illumination and noise	X	$\surd$	X	*NA*	BG
Halo effect	$\surd$	X	*NA*	*NA*	BG
Static Background	X	X	*NA*	*NA*	BG
Dynamic BG-A	X	X	*NA*	*NA*	BG
Dynamic BG-B	X	$\surd$	$X/\surd$	*NA*	BG/FG
Dynamic BG-C	$\surd$	X	*NA*	*NA*	BG
Dynamic BG-D	$\surd$	$\surd$	*NA*	*NA*	FG

Table 3.1: Pixel type is determined by the combination of different modules.

In our FTSG framework, pixels are categorized into six different types. Pixels are firstly classified as foreground and background after pixel level classification by flux tensor chapter 2 and split Gaussian models section 3.1.1. In section 3.2, foreground pixels are further classified into moving foreground pixels, static foreground pixels and illumination changes. Then in section 3.3, small blobs are classified as dynamic background pixels, which reduces noise effect and integrates dynamic background pixels. Afterwards, static foreground pixels are categorized into stopped foreground object and revealed background. Table 3.1 is a summary of all types of pixels.

These six types of pixels are represented as following:

$$Type(x, y) \in \{FG_{moving}, FG_{static}, BG_{static}, BG_{illum}, BG_{revealed}, BG_{dynamic}\} \tag{3.4}$$

Due to dynamic nature of both the foreground objects of interest and background unpredictability, automatic parameter adaptation and model update are necessary. In our model update and parameter tuning scheme, except a few parameters that are set fixed, which are some well known parameters also used by many other methods such as background learning rate and foreground, background ratio threshold, other parameters are all adaptive changing. We represent those adaptively changing parameters and background model as following:

$$Param(t) = \{T_{Flux}, T_{BlobSize}\} \tag{3.5}$$

$$Model(t) = \{\mu_{BG}(x, y), \sigma_{BG}(x, y), \omega_{BG}(x, y)\} \tag{3.6}$$

Parameters and background models are updated with our model update and parameter tuning scheme shown in figure 3.6.

Thresholding is always a difficult problem in temporal differencing methods and most of them just used a fixed global threshold to classify foreground and background. In FTSG,

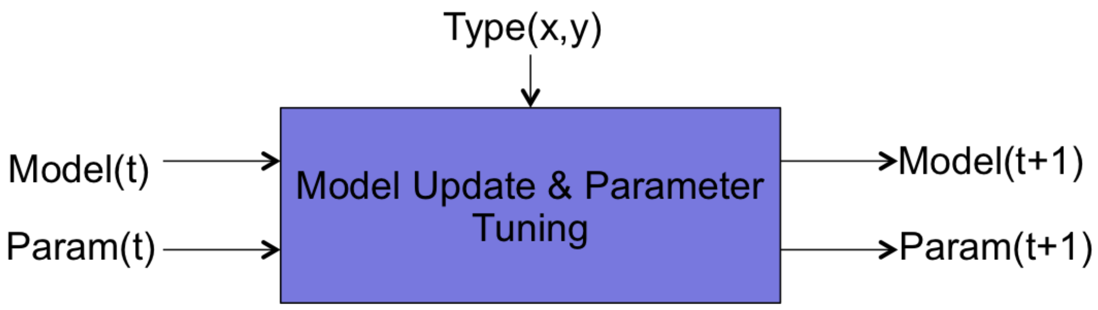


Figure 3.6: Model update and parameter tuning

instead of using a fixed global threshold, we adaptively change flux threshold T_{Flux} according to the number of background Gaussian models.

$T_{BlobSize}$ determines dynamic background blob size. It is also related to the number of background Gaussian models. For static background, $T_{BlobSize}$ is small so less blobs will be added to background model that the background model is more sensitive to changes. On the contrary, $T_{BlobSize}$ is larger in dynamic background so that more false positive detections due to dynamic background will be added to background model.

Details of background model update and parameter tuning are shown in figure 3.7.

Static background and illumination changes are updated into background model as:

$$\mu_t = (1-\alpha)M\mu_{t-1} + \alpha M I_t \tag{3.7}$$

$$\sigma_t^2 = (1-\alpha)M\sigma_{t-1}^2 + M\alpha(I_t-\mu)^T\alpha(I_t-\mu) \tag{3.8}$$

$$\omega_{i,t} = (1-\alpha)\omega_{i,t-1} + \alpha M \tag{3.9}$$

$$M = (1-\mathbf{F}_B) \cup (\mathbf{F}_{amb} - \mathbf{F}_S) \tag{3.10}$$

where α is a fixed learning rate set to 0.004 and M stands for update mask. Background revealed by removed objects and dynamic background are incorporated to background model as new Gaussian distributions. A new Gaussian is initialized with a high variance and low

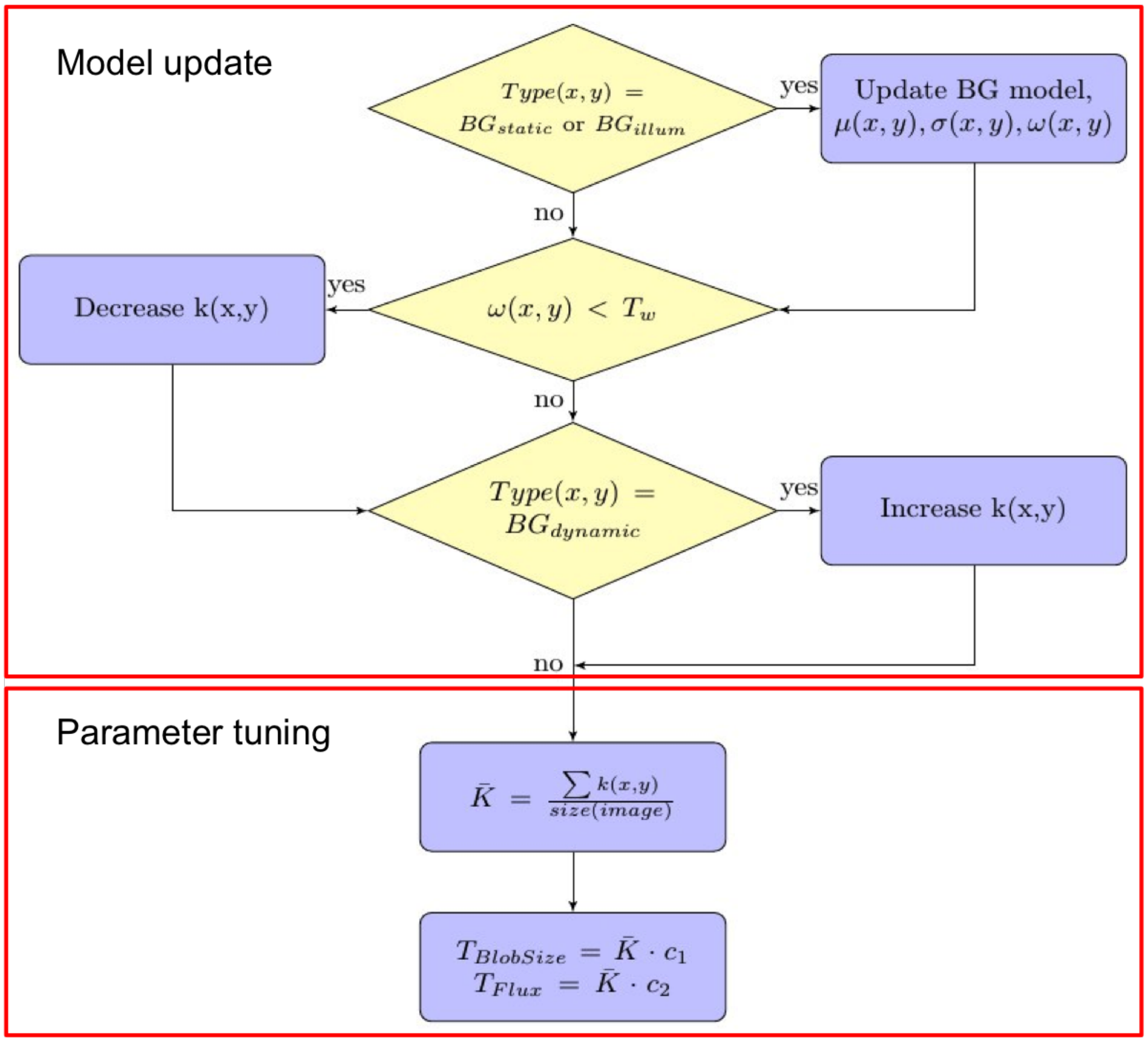

Model update
$Type(x,y) = BG_{static}$ or BG_{illum}
yes
Update BG model, $\mu(x,y), \sigma(x,y), \omega(x,y)$
no
Decrease k(x,y)
yes
$\omega(x,y) < T_w$
no
$Type(x,y) = BG_{dynamic}$
yes
Increase k(x,y)
no
Parameter tuning
$\bar{K} = \frac{\sum k(x,y)}{size(image)}$
$T_{BlobSize} = \bar{K} \cdot c_1$
$T_{Flux} = \bar{K} \cdot c_2$

Figure 3.7: Model update and parameter tuning system flow

weight, and its mean is set to the current pixel value.

If there is a large persistent change, a new model will be added to each pixel (i.e. in PTZ scenario [2], camera field of view change triggers large persistent change). Existing Gaussian models with weights less then a threshold T_l are discarded.

Foreground model update: As in the case of the background model, a conservative update strategy is used for the foreground model. Foreground model is only updated with the foreground regions indicated by the inverse of the background model update mask. In order to accommodate fast changing foreground, a high learning rate is used for foreground update.

3.5 Results and Analysis

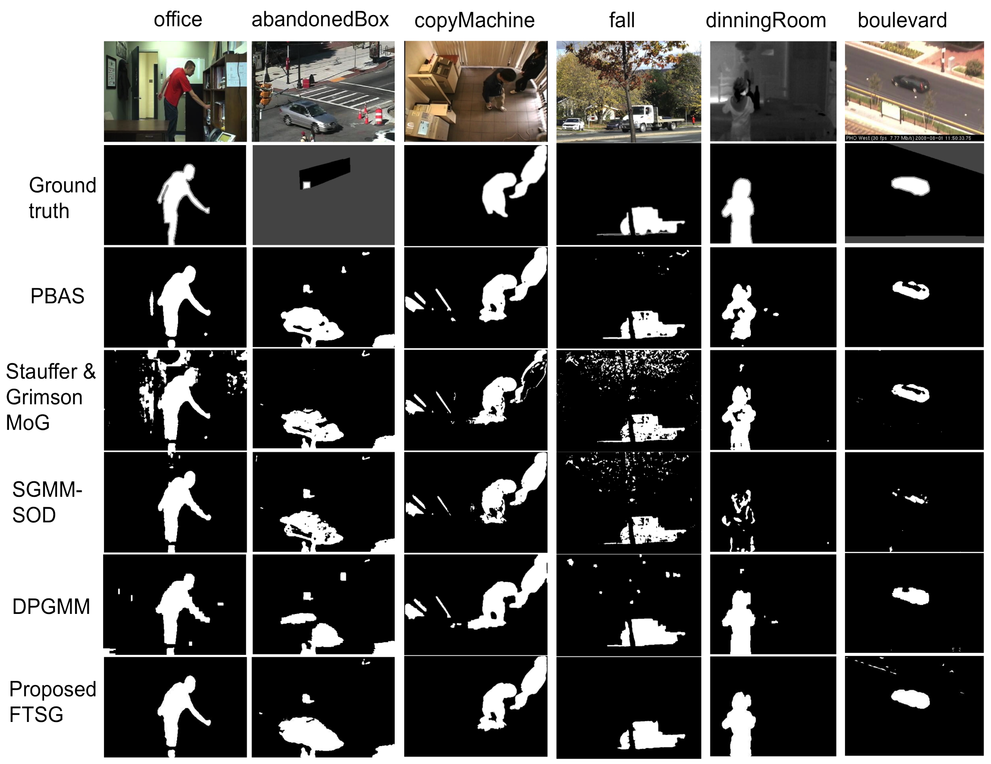


Figure 3.8: Selected foreground detection results from four state-of-the-art change detection algorithms and our FTSG method on CVPR2012 Changedetection dataset. See Table II for further quantitative details.

In this section, the proposed flux tensor with split Gaussian models system is evaluated using the dataset and evaluation metrics in ChangeDetection benchmark dataset for CVPR 2014 workshop [2]. The ChangeDetection dataset contains majority of the change detection difficulties including dynamic background, long-term static objects, removed objects,

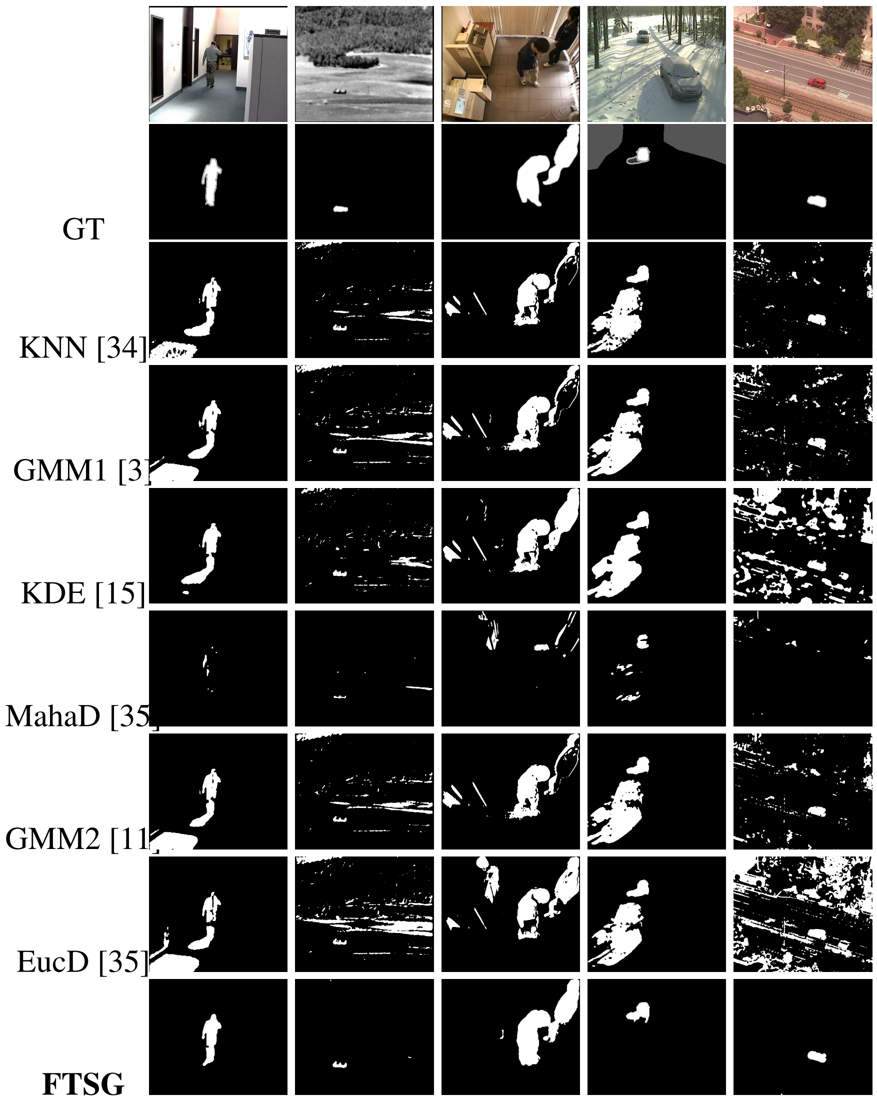


Figure 3.9: Selected foreground detection results from six state-of-the-art change detection algorithms and our FTSG method on CVPR 2014 Change Detection dataset [2]. See Table 3.3 for quantitative results.

sudden/gradual illumination changes, shadows, camouflage and unstable camera. There are eleven video categories and each of them contains four to six video sequences. ChangeDetection provides standard evaluation metrics so by using the same evaluation metrics, our FTSG system is compared to the state-of-the-art motion detection algorithms.

	Recall	Spec	FPR	FNR	PWC	F	Prec
Bad Weather	0.7457	0.9991	0.0009	0.2543	0.5109	0.8228	0.9231
Low Framerate	0.7517	0.9963	0.0037	0.2483	1.1823	0.6259	0.6550
Night Videos	0.6107	0.9759	0.0241	0.3893	4.0052	0.5130	0.4904
PTZ	0.6730	0.9770	0.0230	0.3270	2.5519	0.3241	0.2861
Turbulence	0.6109	0.9998	0.0002	0.3891	0.1987	0.7127	0.9035
Baseline	0.9513	0.9975	0.0025	0.0487	0.4766	0.9330	0.9170
Dynamic Background	0.8691	0.9993	0.0007	0.1309	0.1887	0.8792	0.9129
Camera Jitter	0.7717	0.9866	0.0134	0.2283	2.0787	0.7513	0.7645
Intermittent Object	0.7813	0.9950	0.0050	0.2187	1.6329	0.7891	0.8512
Shadow	0.9214	0.9918	0.0082	0.0786	1.1305	0.8832	0.8535
Thermal	0.7357	0.9960	0.0040	0.2643	1.1823	0.7768	0.9088
Overall	0.7657	0.9922	0.0078	0.2343	1.3763	0.7283	0.7696

Table 3.2: Comparison of the proposed FTSG system on all eleven scenarios using all seven measures.

	Recall	Spec	FPR	FNR	PWC	F	Prec
KNN [34]	0.6650	0.9802	0.0198	0.3350	3.3200	0.5937	0.6788
GMM1 [3]	0.6846	0.9750	0.0250	0.3154	3.7667	0.5707	0.6025
KDE [15]	0.7375	0.9519	0.0481	0.2625	5.6262	0.5688	0.5811
MahaD [35]	0.1644	**0.9931**	**0.0069**	0.8356	3.4750	0.2267	0.7403
GMM2 [11]	0.6604	0.9725	0.0275	0.3396	3.9953	0.5566	0.5973
EucD [35]	0.6803	0.9449	0.0551	0.3197	6.5423	0.5161	0.5480
FTSG	**0.7657**	0.9922	0.0078	**0.2343**	**1.3763**	**0.7283**	**0.7696**

Table 3.3: Quantitative comparison of the proposed FTSG system to several state-of-the-art methods.

3.5.1 Terms and Definitions

Following is the formal definition of the terms and equations used for evaluation.

Evaluation metrics	
Recall	$\frac{\text{TP}}{\text{TP+FN}}$
Precision	$\frac{\text{TP}}{\text{TP+FP}}$
Specificity	$\frac{\text{TN}}{\text{TN+FP}}$
False Positive Rate(FPR)	$\frac{\text{FP}}{\text{FP+TN}}$
False Negative Rate(FNR)	$\frac{\text{FN}}{\text{TP+FN}}$
Percentage of Wrong Classification(PWC)	$\frac{\text{100*(FN+FP)}}{\text{TP+FP+TN+FN}}$
F-Measure	$\frac{\text{2*Precision*Recall}}{\text{Precision+Recall}}$

Table 3.4: Evaluation metrics used for CVPR2012 Change Detection benchmark dataset where TP, FP, TN, FN stand for true positive, false positive, true negative and false negative respectively.

Among all, the PWC (Percentage of Wrong Classification) and F-measure use all four basic properties and thus give more comprehensive evaluation results on how well a change detection algorithm perform. In order to know the superiority of a change detection method, the average rank score with respect to the seven evaluation metrics in Table 3.4 is computed as

$$\begin{aligned} ranking= \quad & (rank{:}Recall+rank{:}Spec+rank{:}FPR+rank{:}FNR+ \\ & rank{:}PWC+rank{:}FMeasure+rank{:}Precision)/7 \end{aligned} \tag{3.11}$$

Then the superiority of a method is obtained by ranking the average rank score of all methods.

3.5.2 Parameter Selection

Besides those dynamic parameters that are mentioned in section 3.4, there are few other parameters in Flux Tensor Split Gaussian that are set to fixed values empirically.

Flux tensor

The parameters for flux tensor are spatial(Wx) and temporal(Wt) window sizes. In ChangeDetection datasets, Wt and Wx are both set to 7 frames for all sequences. Wt can be set according to video frame rate and speed of foreground objects. Smaller Wt for sequences with low frame rate and fast moving foreground objects, whereas larger Wt is suitable for sequences with high frame rate and slow moving foreground objects. Wx can be set according to average foreground object size. Larger Wx can fill larger holes of unit color interior regions of moving foreground objects, however, more halo effects will be created. Smaller Wx is preferable for small foreground objects.

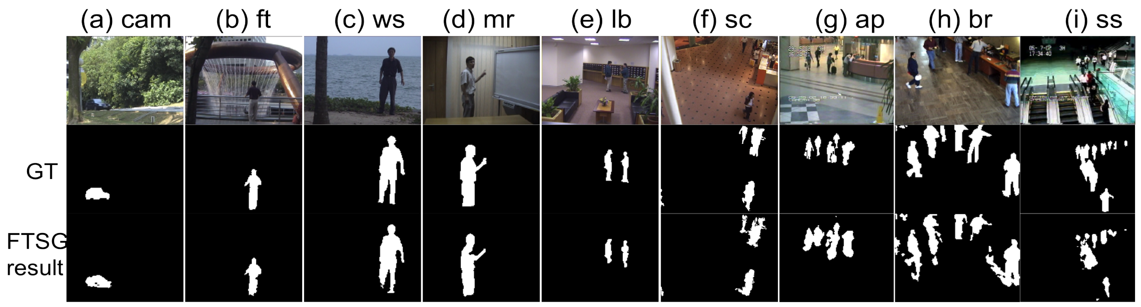


Figure 3.10: Selected foreground detection results on star dataset. Row 1 original images, Row 2 manual Ground Truth, Row 3 detection result from proposed FTSG method.

method	cam	ft	ws	mr	lb	sc	ap	br	ss	mean
Li 2 [29]	0.1596	0.0999	0.0667	0.1841	0.1554	0.5209	0.1135	0.3079	0.1294	0.1930
Stauffer [3]	0.0757	0.6854	0.7948	0.7580	0.6519	0.5363	0.3335	0.3838	0.1388	0.4842
Culibrk	0.5256	0.4636	0.7540	0.7368	0.6276	0.5696	0.3923	0.4779	0.4928	0.5600
Maddalena	0.6960	0.6554	0.8247	0.8178	0.6489	0.6677	0.5943	0.6019	**0.5770**	0.6760
DP-GMM	**0.7624**	0.7265	**0.9134**	0.8371	0.6665	0.6721	0.5663	**0.6273**	0.5269	0.6998
FTSG	0.7564	**0.7496**	0.8889	**0.8834**	**0.6910**	0.6475	**0.6346**	0.6015	0.5343	**0.7097**

Table 3.5: Quantitative evaluation of the proposed FTSG and five other methods on the star dataset. Results of other foreground detection methods are from [1].

Split Gaussian models

There are two sets of learning rate for BGSG (Split Gaussian Background subtraction), the learning rate for initialization stage is 0.09 and after that it is set to 0.004. The background model matching threshold T_b is set to 3. These parameters are fixed for all the sequences in ChangeDetection dataset. For foreground model FGSG the matching threshold is set to $T_f = 20$ and its learning rate is $\alpha_{fg} = 0.5$. The learning rate of Background Split Gaussian model(BG-SG) needs to be adjusted according to different frame rate. We set the learning rate alpha to 0.004 for all sequences in ChangeDetection datasets. The background model matching threshold Tb is set depending on image contrast, image dimension and background type. For images with low contrast, single channel and relatively static background, a small Tb is adopted. Larger Tb is used for sequences with dynamic background, higher image contrast and multiple channels. We use Tb = 3 for all image category except thermal, in which Tb = 2 is adopted, in ChangeDetection dataset.

3.5.3 Comparison Results

Table 3.2 shows results of the proposed approach on all eleven scenarios based on evaluation metrics described in Table 3.4. In the last row of table 3.2, the average scores of recall, specificity, FPR, FNR , PWC, F-Measure and Precision are obtained and the average ranking score is calculated based on them. On seven out of eleven scenarios and on the overall evaluation FTSG outperforms not only the listed state-of-the-art methods but also the new change detection challenge submissions in terms of average ranking.

Table 3.3 shows the comparison result of FTSG with state-of-the-art change detection methods. Evaluation scores of those methods are obtained from `http://www.`

changedetection.net. Best result of each metric is highlighted and in all the measures listed in Table 3.3. It can be seen that FTSG outperforms all the listed methods in five out of seven measures and has the second best score in the remaining two measures, specificity and FPR.

Figure 3.8 shows moving object detection results for various algorithms including proposed flux tensor with split Gaussian models (FTSG) on ChangeDetection 2012 dataset with some typical frames from each category. Flux tensor is robust to illumination changes, noises and shadow because it is based on spatial and temporal derivatives and it also has a local averaging step. Therefore the detection results of FTSG in office(Column 1) and copyMachine (Column 3) does not have any false positive detections corresponding to illumination or shadow effects, whereas all other methods have some false positive due to illumination or shadow problems. While temporal differencing based methods such as Flux tensor cannot detect slow moving or stopped objects. The proposed FTSG method can detect all static foreground objects including the box in abandonedBox (Column 2), the waiting person in copyMachine (Column 3) and the standing person in dinningRoom (Column 5) because FTSG incorporates Split Gaussian FG/BG modeling. In the abandonedBox and the copyMachine sequences (Col 2, 3) as we can see the classical MoG method absorbs the box and the waiting person into the background. Both Flux tensor and Split Gaussian models methods can handle dynamic background to some extend and the combination of these two methods (proposed FTSG) handles dynamic background even better. Fall (Column 4) is a visual result of dynamic background sequence and FTSG gives perfectly clean foreground detection result. In conclusion, the combination of two complementary change detection approaches, temporal differencing based Flux tensor and background subtraction based Split Gaussian models, produces a better change detection algorithm.

Figure 3.9 shows another set of moving object detection results for various algorithms including proposed Flux Tensor with Split Gaussian models (FTSG) on CVPR 2014 Change Detection dataset [2] with some typical frames selected from the 11 categories. The proposed FTSG is robust to illumination changes (col 1), it can detect long term static objects (col 3), and it also handles dynamic background (col 2). Image in col 4 demonstrates that FTSG can correctly identify revealed background by removed object, and image in col 5 shows that FTSG can adapt to scene changes quickly (sudden change of camera focus).

In order to show the generalization of FTSG and its capability of handling sudden illumination changes. Another dataset, star [17], is used to evaluate our FTSG method. This dataset contains difficulties including dynamic background, sudden illumination changes, shadow, crowded environment, systematic noise and camouflage. A different evaluation metric is proposed in [17] that for each sequence a similarity score, which is computed as similarity = TP/(TP+FN+FP), is used to evaluate the detection result. Table 3.5 shows the comparison results of FTSG with some state-of-the-art methods on star dataset. Evaluation scores of other methods in Table 3.5 are obtained from [1].With one frame initialization capability, FTSG detects sudden illumination changes and adapts to the new background scene quickly in sequence lb. However, due to low frame rate, flux tensor produces many false positive detections even with small time step window size, which causes FTSG get poor performance in br and sc sequences (Figure 3.10 f, h). Overall, FTSG still gets the 1^{st} rank in four out of the nine sequences, which shows its superiority.

A prototype of the proposed system implemented in Matlab runs at 10 fps for a 320×240 video. Matlab implementation of Flux tensor only detection runs at 50 fps. Flux tensor computation can be easily parallelized for different architectures as in [**?**] because of the fine grain parallelism of the filter operations.

Chapter 4

Robust Motion Detection Using PF Video Stabilization Combined with Flux Tensor Spatiotemporal Analysis

4.1 System Overview

Foreground object detection is almost impossible using either temporal difference or background subtraction methods in moving platform. However, by transforming all images into a same coordinate system makes such problem solvable using these methods. Image registration can be broadly categorized into feature based and area based registration[cite]. Feature-based registration methods tends to find correspondence between image features such as points, lines, and contours. These types of methods are more preferable when the local structural information is more significant than the information carried by the image intensities. Area based registration methods can register images without feature extraction or saliency detection. Image patches with fixed size or even entire images can be used for the correspondence estimation during the registration step of area based methods. In this chapter we describe several modules for robustly detecting moving objects in full motion video acquired from a moving airborne platform. The first module stabilizes the video with respect a base frame in the video sequence over a short time period of about 200 frames. This registration step is necessary to distinguish between the moving background due to camera motion and the foreground target motion that is of interest to be tracked. Stabilizing the video over short time periods requires frame-to-frame registration that is accomplished by firstly identifying prominent Beltrami color metric tensor features and Shi-Tomasi features that are matched between frames using a block matching approach. Secondly, featureless regions will be grouped into larger overlapping blocks. This combination of using feature extraction with local multi-scale region/block matching we term hybrid prominent feature-block matching. Once the features are available a RANSAC approach is used to find the best homography to remap one frame into the coordinate system of a base video frame within the chunk of 100 frames. The homography model assumes that a single plane is sufficient to model the 3D scene which may not be valid for complex video sequences. Once the frame homographies are available the video stabilization module is followed by the flux tensor motion detection module which uses local second derivative information efficiently to detect moving salient features. The flux tensor is able to filter out noisy responses efficiently and performs better than the mixture of Gaussians approach for motion detection. Spurious responses from the frame boundaries and other post-processing operations are applied to reduce the false alarms and produce accurate moving blob regions and blob statistics that will be useful for tracking.

4.1.1 Prominent Feature Block-based Region Selection

4.1.2 Hybrid Prominent Feature Block Detection

Feature based methods are usually more robust in terms of matching if significant structure information can be found. Therefore, the first step is to detect salient or prominent feature such as lines and corners. The image is divided into non-overlapping blocks as shown in figure 4.1. Prominent feature blocks are blocks that contain obvious line or corner features.

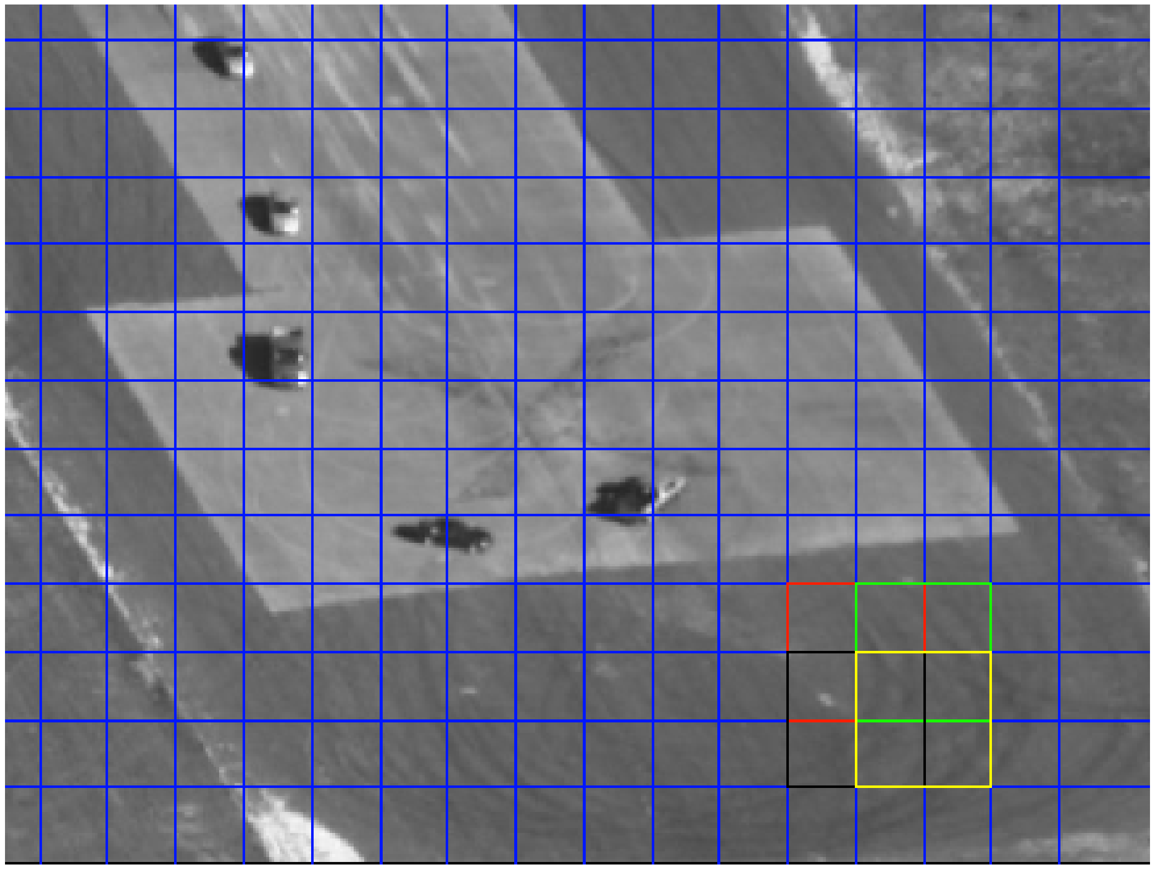


Figure 4.1: Divide image into non-overlapping blocks (blue grids). Use larger overlapping blocks at featureless regions (red, green, black and yellow blocks)

Many first and second derivative feature detectors and descriptors are available in the literature [36]. We use the 2D color structure tensor defined in terms of the outer product of spatial gradients in each channel is given in Eq.4.1 with C_i representing image channels ($i = 3$ for RGB color), and further described in [37, 38]. The 2D grayscale structure tensor matrix is also referred to as the second moment or autocorrelation matrix [36].

Local descriptors based on the two eigenvalues of the structure tensor provide information about the signal in orthogonal directions. Small eigenvalues are indicative of noise so the trace of $\mathbf{J_C}$ can filter these locations.

$$\mathbf{J_C} = \begin{bmatrix} \sum\limits_{i=C_1,C_2,\ldots} \int_{\mathbf{\Omega}} \left(\frac{\partial \mathbf{I}_i}{\partial x}\right)^2 d\mathbf{y} & \sum\limits_{i=C_1,C_2,\ldots} \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}_i}{\partial x}\frac{\partial \mathbf{I}_i}{\partial y} d\mathbf{y} \\ \\ \sum\limits_{i=C_1,C_2,\ldots} \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}_i}{\partial x}\frac{\partial \mathbf{I}_i}{\partial y} d\mathbf{y} & \sum\limits_{i=C_1,C_2,\ldots} \int_{\mathbf{\Omega}} \left(\frac{\partial \mathbf{I}_i}{\partial y}\right)^2 d\mathbf{y} \end{bmatrix} \tag{4.1}$$

The eigenvalues of $\mathbf{J}_C$ are correlated with the local image properties of edgeness and cornerness, defined as $\lambda_1 >> 0$, $\lambda_2 \approx 0$ and $\lambda_1 \approx \lambda_2 >> 0$ respectively. For a 2D multi-spectral image, the Beltrami operator defines a metric on a two-dimensional manifold $\{x, y, C_1(x, y), C_2(x, y), C_3(x, y)\}$ in the five-dimensional spatial-color space $\{x, y, C_1, C_2, C_3\}$:

$$\begin{aligned} Beltrami(\mathbf{I_{RGB}}) &= \mathbf{det}(\mathcal{I} + \mathbf{J_C}) \\ &= 1 + \mathbf{trace}(\mathbf{J_C}) + \mathbf{det}(\mathbf{J_C}) \\ &= 1 + (\lambda_1 + \lambda_2) + \lambda_1\lambda_2 \end{aligned} \tag{4.2}$$

After filtering, prominent features can be easily located. However, such features are usually clustered together instead of spread out the whole image. In this case, the correspondence established between frames may be biased. In addition, using fixed block size can only detect features in a small range of scale. However, multi-scale feature selection is computational expensive. Therefore, we use larger overlapping blocks to cover featureless regions as shown in figure , so that the features are guaranteed to be uniformly distributed all over the image and the two level block size simulates the multi-scale feature detection while it is less computational expensive. By using large blocks with NCC based matching, this is similar to intensity-based registration. We name such combination of prominent feature blocks and larger overlapping blocks hybrid prominent feature block and it has several advantages. It guarantees uniformly distributed features, it increase the feature scale range and it is more efficient than either multi-scale feature based registration or intensity-based registration. Figure 4.2 shows an example of hybrid prominent feature blocks. As we can see, the feature points are uniformly distributed over the image.

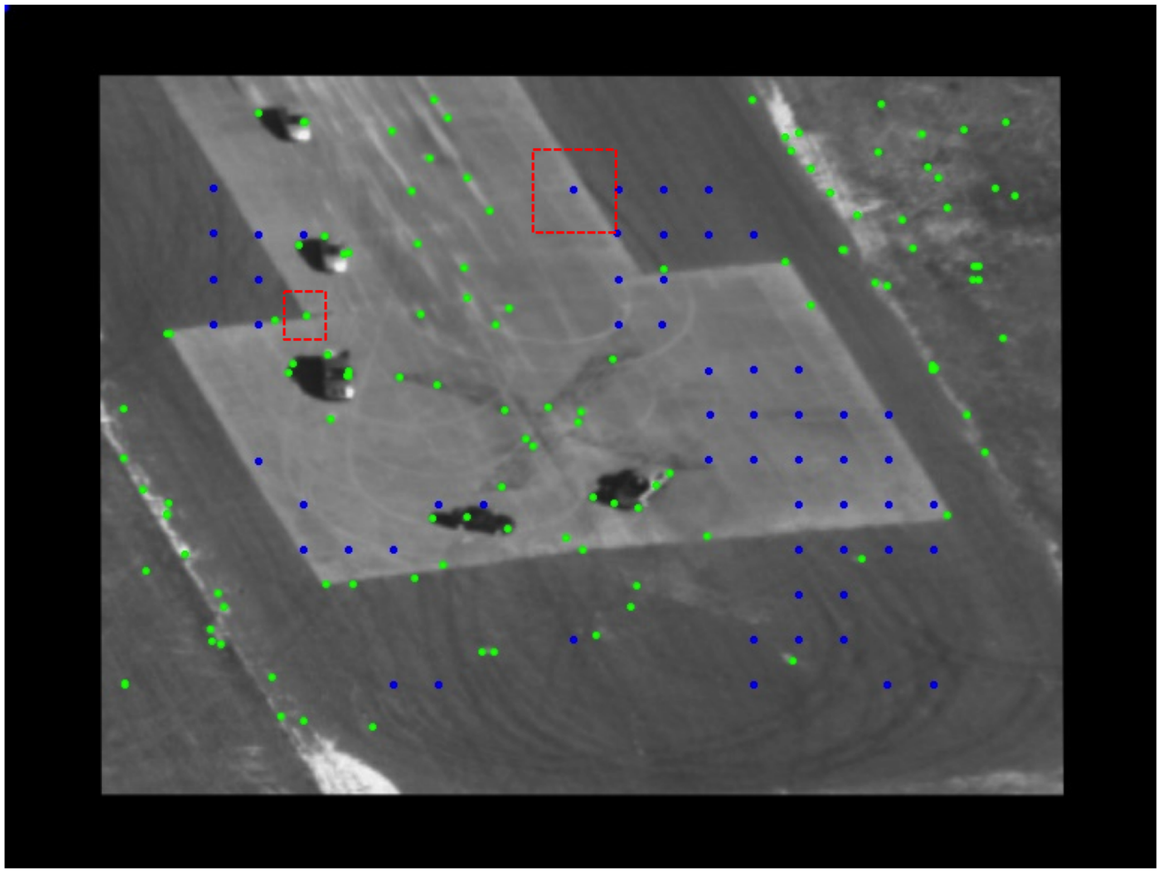


Figure 4.2: Hybrid prominent feature blocks. Points are the center of feature blocks. Green points are prominent feature blocks, blue points are larger overlapping blocks.

4.1.3 Hybrid PF Block Region-Correspondences

Once the hybrid prominent feature blocks are selected based on the strategy described in last section in both reference and input images, then the next step is to find the correspondence matching between these feature blocks. The matching process is using a hybrid prominent block(HPF) in the source image to searches for the best matching or most similar *overlapping* block contained within a search zone/window in the target image;

We use intensity based similarity measurements for both non-overlapping prominent feature blocks and overlapping larger featureless blocks. Normalized cross correlation (NCC) and sum of absolute difference (SAD) are the two of the most popular similarity measurements, which are adopted in our registration scheme. NCC is invariant to intensity changes between source and target images such as illumination changes while SAD is more computational efficient. These two methods can be used interchangeably for different purpose.

The minimum of the SAD measure can be defined as,

$$\Delta X_{opt} = \arg\min_{\Delta X} \sum_{X \in \Omega} |\mathbf{I}(X + \Delta X, t-k) - \mathbf{I}(X, t)| \tag{4.3}$$

The NCC between target (or reference) image $\mathbf{I}(X, t-k)$ and source (or template) image $\mathbf{I}(X, t)$ is defined as,

$$\gamma = \frac{\sum_{X \in \Omega}[\mathbf{I}(X + \Delta X, t-k) - \mu_{t-k}][\mathbf{I}(X, t) - \mu_t]}{\sqrt{\sum_{X \in \Omega}[\mathbf{I}(X + \Delta X, t-k) - \mu_{t-k}]^2 \sum_{X \in \Omega}[\mathbf{I}(X, t) - \mu_t]^2}} \tag{4.4}$$

where $\mu_{t-k} = \langle \mathbf{I}(X + \Delta X, t-k) \rangle$ and $\mu_t = \langle \mathbf{I}(X, t) \rangle$ are the local intensity means (averages) in the target and template image regions respectively and the denominator is the product of the local variances. The NCC for vector images (RGB color) can be appropriately extended.

We want to find the translation or displacement ΔX that maximizes the NCC measure,

$$\Delta X_{opt} = \arg\max_{\Delta X} |\gamma(\Delta X)| \tag{4.5}$$

The NCC can also be interpreted as the cosine of the angle between the two mean corrected region blocks. If we represent the mean subtracted pixels in the target and source windows as the vectors $\overrightarrow{W_T}$, $\overrightarrow{W_S}$, respectively then,$NCC \equiv \gamma(\Delta X) = (\overrightarrow{W_T} \bullet \overrightarrow{W_S})/(\|\overrightarrow{W_T}\|\|\overrightarrow{W_S}\|)$.

Figure 4.3 shows an example of the region-correspondences between two frames from a video sequence.

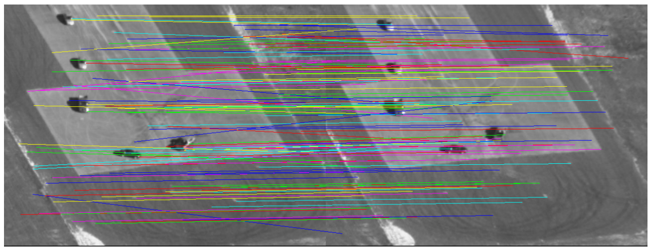


Figure 4.3: Region correspondences based on NCC matching.

4.1.4 Projective Transformation Estimation

Once region-based block correspondences are established, we need to compute the homography relating the the two coordinate systems. This enables image $\mathbf{I}(X, t)$ to be mapped into the coordinate system of the base frame for a given video segment $\mathbf{I}(X, t-k)$. Note that we are interested in finding a good solution for the homography, and *not* on finding the unique solution for the true 3D camera motion, as our goal is mainly to compensate for and remove the effects of the background or (dominant) ground plane motion. Since UAV imagery can have significant perspective effects an projective mapping is more accurate than a single global affine transformation. Other approaches include multiple local affine projections [39] and non-rigid transformations [40]. The projective mapping function or homography uses the coordinates of the corresponding PF block centroids (control points) to find a weighted least squares solution for the transformation matrix coefficients. The homography is used to warp the image at time t into the coordinate system of the base frame at time $(t-k)$. The two images, $\mathbf{I}(x,y,t)$ and $\mathbf{I}(x,y,t-k)$ can be related by a projective transformation (or homography) when the scene points are approximately planar. Let the image coordinates of the same scene point lying on the plane π be $P(x,y)$ and $P'(x',y')$, in the view at time t and $(t-k)$ respectively. The two views can be related by the following homogeneous relationships:

$$x' = \frac{ax+by+c}{gx+hy+w} \tag{4.6}$$

$$y' = \frac{dx+ey+f}{gx+hy+w} \tag{4.7}$$

The homography can be written in matrix notation as:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & w \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.8}$$

$$P' = \mathbf{A}_{(t-k,t)} P \tag{4.9}$$

This transforms position P observed at time t, to position P' in the coordinate system at time $(t-k)$ via the projective transformation matrix (a backward transformation from time t to time $(t-k)$). Usually we assume $w = 1$ in matrix $\mathbf{A}$.

Suppose we are given three images, $\mathbf{I}(x,y,t-2)$, $\mathbf{I}(x,y,t-1)$, $\mathbf{I}(x,y,t)$ with corresponding planar points, P'', P', P and homography transformation matrices $\mathbf{A}_{(t-1,t)}$ and $\mathbf{A}_{(t-2,t-1)}$ that projectively maps t to $(t-1)$ (i.e. Frame 2 to Frame 1) and $(t-1)$ to $(t-2)$

(i.e. Frame 1 to Frame 0), respectively. Without loss of generality we assume for simplicity of notation that the images are sequentially sampled at one unit time intervals, t, $(t-1)$, and $(t-2)$. We can then write the two respective projective transformations as,

$$P' = \mathbf{A}_{(t-1,t)}P \quad and \quad P'' = \mathbf{A}_{(t-2,t-1)}P' \tag{4.10}$$

and the composite or cumulative projective transformation relating pixels in frame t to pixels in frame $(t-2)$ (i.e. pixels in Frame 2 to pixels in Frame 0), as the product of two homographies or projective maps/transformations:

$$P'' = \mathbf{A}_{(t-2,t-1)}\mathbf{A}_{(t-1,t)}P \tag{4.11}$$

In the general case, mapping pixel positions from frame t to corresponding pixel positions in the coordinate system of frame $(t-k)$, we have

$$P(t-k,t) = \mathbf{A}_{(t-k,t)}P(t,t) \tag{4.12}$$

$$\mathbf{A}_{(t-k,t)} = \mathbf{A}_{(t-k,t-k+1)}\mathbf{A}_{(t-k+1,t-k+2)}....\mathbf{A}_{(t-2,t-1)}A_{(t-1,t)} \tag{4.13}$$

We also need to specify the coordinate system in which we reference or measure a pixel's position. Since the prime notation is limited, $P(t-k,t)$ denotes pixel position/geometry from image $\mathbf{I}(x,y,t)$ mapped to the coordinate system of image frame $\mathbf{I}(x,y,t-k)$ and $P(t,t)$ is the pixel position measured in its original coordinate system $\mathbf{I}(x,y,t)$. The elements of matrix $\mathbf{A}$ in Eq. 4.8 and 4.9 can be solved using weighted least squares, robust statistics such as LMedS or combinatorial methods such as RANSAC. Each pair of corresponding points provides three linear constraints that can be written in a matrix form

$\mathbf{B}_i\mathbf{a} = 0$ as shown below,

$$\mathbf{B}_i\mathbf{a} = \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & -w_i'\mathbf{x}_i^{\mathrm{T}} & -y_i'\mathbf{x}_i^{\mathrm{T}} \\ w_i'\mathbf{x}_i^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -x_i'\mathbf{x}_i^{\mathrm{T}} \\ y_i'\mathbf{x}_i^{\mathrm{T}} & x_i'\mathbf{x}_i^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \\ \mathbf{a_3} \end{bmatrix} = \mathbf{0} \tag{4.14}$$

where $\mathbf{a}_i^{\mathrm{T}}$ is the i^{th} row of $\mathbf{A}$ in Eq. 4.8, [41]. This above equation, $\mathbf{B}_i\mathbf{a} = 0$, is an equation *linear* in the unknown vector $\mathbf{a}$ [Hartley2003]. The matrix $\mathbf{B}_i$ is a 3×9 matrix, and $\mathbf{a}$ is a 9×1-vector made up of the entries of the matrix A,

$$\mathbf{a} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix}, \mathbf{a}_i = \begin{bmatrix} A(i,1) \\ A(i,2) \\ A(i,3) \end{bmatrix} \tag{4.15}$$

Notice that there are three equations in (4.14), however just two of them are linearly independent since the thrid row is obtained up to scale. Therefore each point correspondence provides two equations in the entries of A. Based on this, (4.14) can be written as

$$\mathbf{B}_i\mathbf{a} = \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & -w_i'\mathbf{x}_i^{\mathrm{T}} & -y_i'\mathbf{x}_i^{\mathrm{T}} \\ w_i'\mathbf{x}_i^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -x_i'\mathbf{x}_i^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \\ \mathbf{a_3} \end{bmatrix} = \mathbf{0} \tag{4.16}$$

where now $\mathbf{B}_i$ is the 2×9 matrix of (4.16). (4.16) can be solved for $mathbfa$ (9 unknown elements) using the normalized Direct Linear Transformation (DLT) approach (Alg. 2). DLT provides for improved numerical stability and accuracy when solving for $\mathbf{A}$.

Notice that for improving the accuracy of the results in the DLT algorithm, a normalization process (Alg. 1) has to be applied beforehand. This step is very important for less

Algorithm 1 Calculating similarity transformation to be used for normalization. It Computes a similarity transformation T, consisting of a translation and scaling, that takes point $\mathbf{x}_i$ to a new set of $\tilde{\mathbf{x}}_i$ such that the centroid of the points $\tilde{\mathbf{x}}_i$ is the coordinate origin, and their mean distance from the origin is $\sqrt{2}$

```
Input : A set of $n$ 2D points $\mathbf{x}_i$
Output : Similarity transformation $S$
1: Calculate the centroid of the points: $\bar{\mathbf{x}} \leftarrow \frac{1}{n}\sum_i \mathbf{x}_i$
2: Calculate scale factor: $s \leftarrow \dfrac{\sqrt{2}}{\frac{1}{n}\sum_i \sqrt{(\mathbf{x}_i - \bar{\mathbf{x}})^\top(\mathbf{x}_i - \bar{\mathbf{x}})}}$
3: Calculate the similarity transformation: $S \leftarrow \left[\begin{array}{cc|c} s & 0 & -s\bar{\mathbf{x}} \\ 0 & s & \\ 0 & 0 & 1 \end{array}\right]$
```

Algorithm 2 The nomalized DLT for homography estimation

```
Input : A set of $n$ 2D point correspondences $(\mathbf{x}_i, \mathbf{x}'_i)$, where $n >= 4$
Output : Homography matrix A such that $x'_i = Ax_i$
1: Calculate similarity transformation $S$ for $\mathbf{x}$ using Alg. 1
2: Normalization of $\mathbf{x}$: $\tilde{\mathbf{x}}_i \leftarrow S\,\mathbf{x}_i$
3: Calculate similarity transformation $S'$ for $\mathbf{x}'$ using Alg. 1
4: Normalization of $\mathbf{x}'$: $\tilde{\mathbf{x}}'_i \leftarrow S'\,\mathbf{x}'_i$
5: Assemble the $n$ $2 \times 9$ matrices $\mathbf{B}_i$ (using normalized points $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}'_i$) into a single $2n \times 9$ matrix $\mathbf{B}$
6: Calculate the SVD of $\mathbf{B}$. The unit singular vector corresponding to the smallest singular value is the solution for $\mathbf{a}$
7: The matrix $\tilde{A}$ is determined as $\tilde{A} \leftarrow \begin{bmatrix} \mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \mathbf{a}_3^\top \end{bmatrix}$
8: Denormalization: $A \leftarrow (S')^{-1}\tilde{A}\,S$
```

well conditioned problems such as DLT. Apart from improved accuracy of results, normalizing data has one more advantage, namely that an algorithm which incorporates an initial data normalization step will be invariant with respect to arbitrary choices of the scale and coordinate origin. As mentioned in [41], this is because the normalization step cancels out the effect of reference frame changes, by effectively choosing a canonical coordinate system for the measurement data. Therefore, algebraic minimization is carried out in a fixed canonical frame, and the DLT algorithm practically becomes invariant to similarity transformations. In order to give an idea of the importance of the normalization step in homography estimation using DLT, we performed a simulation using perturbed synthetic feature points. Figure 4.4 shows a set of 100 points $\mathbf{x}$ that were randomly generated representing some feature points in the first image image (I_1). Then a homography matrix A,

$$A = \begin{bmatrix} 1.4219 & 0.3067 & 5.2096 \\ 0.3153 & 1.2816 & 1.1919 \\ 0.0007 & 0.0007 & 1.0000 \end{bmatrix} \tag{4.17}$$

is randomly generated which maps $\mathbf{x}$ in I_1 to $\mathbf{x}' = A\,\mathbf{x}$ in the second image I_2. The image size is considered to be standard definition size of 640×480 pixels. Some noise (white Gaussian noise with zero mean and standard deviation one) are added to the feature points in I_2. Then the homography transformation between the points in I_1 and points (noise added) in I_2 has been estimated using the DLT algorithm, once using normalized points and the other without normalization. The geometric errors using the estimated homography for both cases are computed. For this experiment, we got 14.33 average pixel error for the non-normalized one and 4.48 for the normalized one. The estimated homography matrices in two cases of directly applying DLT or using a normalization method before DLT are as following, respectively:

$$A_{direct} = \begin{bmatrix} 1.3032 & 0.2211 & 29.4906 \\ 0.2714 & 1.1579 & 17.3624 \\ 0.0006 & 0.0005 & 1.0000 \end{bmatrix} \tag{4.18}$$

$$A_{normalized} = \begin{bmatrix} 1.4245 & 0.2972 & 5.6929 \\ 0.3189 & 1.2733 & 0.5711 \\ 0.0007 & 0.0007 & 1.0000 \end{bmatrix} \tag{4.19}$$

As can be seen, the estimated homography after applying a normalization step (matrix of Eq.(4.19)) numerically is very close to the original used homography (matrix of Eq.(4.17)) and moreover gives a better result.

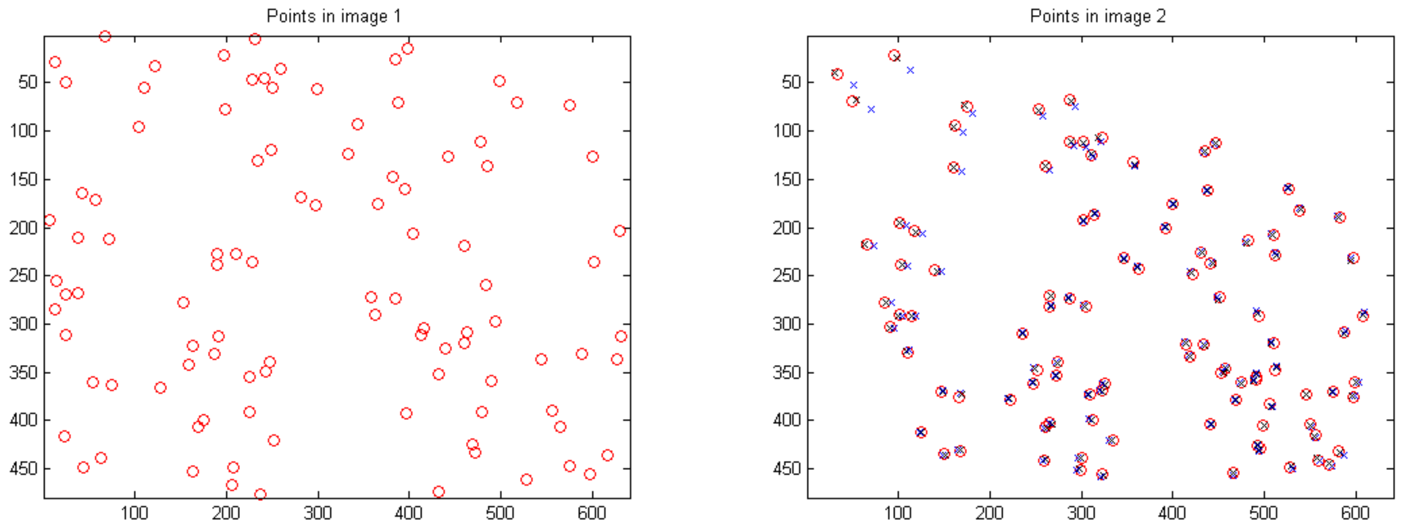


Figure 4.4: Simulated results to demonstrate the importance of using normalization step in DLT homography estimation algorithm. The blue-cross and black-cross marks in the right figure indicate the transformed points from I_1 to I_2 using non-normalized and normalized cases, respectively. The actual feature points in I_2 are drawn in red-circles. For this experiment, we measured 14.33 average pixel error for the non-normalized one and 4.48 for the normalized one. The assumption is that there is no matching error and the noise is equal additive Gaussian added to both I1 and I2

Robust Homography Estimation Using Normalized DLT RANSAC

The DLT method, described in Algorithm 2, is used to solve Eq. 14. The DLT method is robust only if the dominant source of the noise is in the location measurement of corresponding feature points. The DLT method is not appropriate when there are mismatches, that is the two putative feature point correspondences do not correspond to the same real world object at all. For this purpose we use a method, based on RANSAC (Random Sample Consensus), to robustify the estimate with respect to false matches. The RANSAC-based homography estimation incorporating normalized DLT is described in Algorithm 3.

The performance of RANSAC-based homography estimation using Algorithm 3 depends on the proportion of inliers and the number of iterations. The probability that after N iterations of RANSAC we have not picked a set of inliers is given by $(1 - g^4)^N$, with $g = m/n$ being the proportion of the inliers; the behavior of this curve is shown in Fig. 4.5 for three values of N ($N = 10, 100, 1000$). For the RANSAC iteration the initial value of N in Algorithm 3 is initialized with a large value that is then adaptively updated on each iteration using the Equation in Step 15 as shown in Figure 4.6.

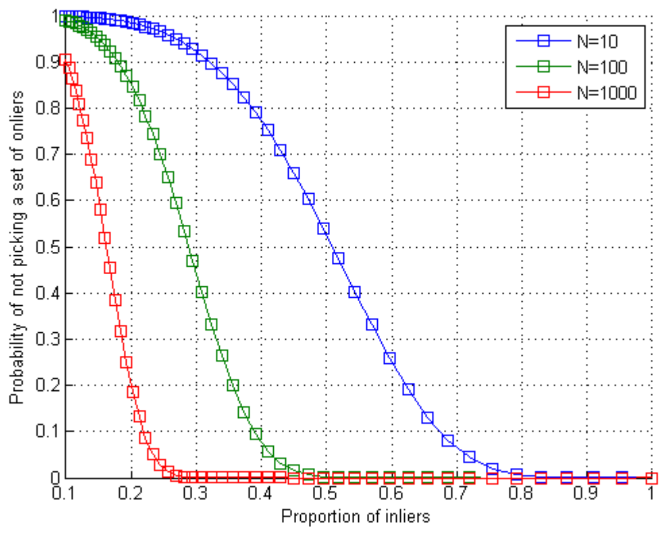


Figure 4.5: Evaluation of the robustness of the proposed RANSAC-based homography estimation in Alg. 3. The horizontal axis indicates the proportion of inliers ($g = m/n$) and the vertical axis shows the probability that after N RANSAC iterations we have not picked a sufficient set of inliers based on the function $(1-g^4)^N$ for three values of N.

An alternative to RANSAC is Least Median of Squares (LMedS) estimation, in which the model is selected using the median of the distances of all points in the dataset (whereas in RANSAC a minimum size subset of samples are randomly selected). As indicated in [41], LMS has the advantage of not requiring anyt thresholds; however it fails if more than

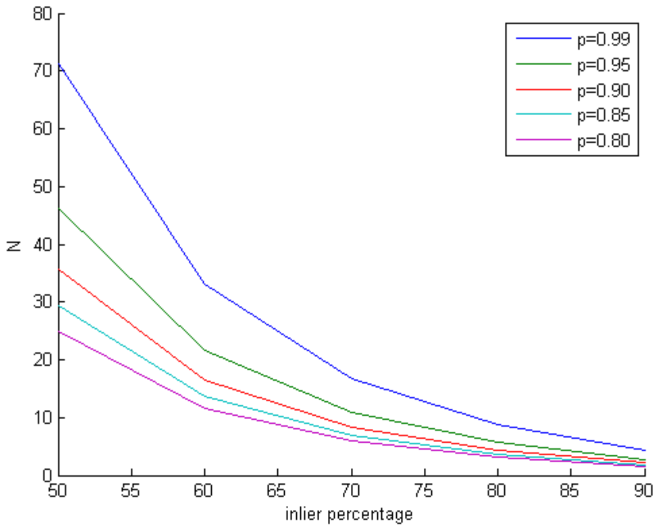


Figure 4.6: Plot of $N = \frac{log(1-p)}{log(1-(1-g)^4)}$, which gives an update for the number of iterations in Algorithm 3. N is adaptively determined in Step 15 of Algorithm 3. The horizontal axis indicates the inlier percentage. Plots for different values of p (see Step 4) that is related to the quality of the estimate is shown in different colors. For higher values of p more iterations are needed.

50% of the data are outliers. There are other variations of RANSAC such as Adaptive-Scale Kernel Consensus (ASKC) which can be used as alternative robust estimators [42]. Alternatively some other methods which consider the hohomraphy estimated by Alg. 3 as an initilization and then iteratively try to minimize the error using Levenberg-Marquardt method in order to optimize the initial estimation [41].

4.1.5 UAV Video Registration Algorithm

A segment of video whose length is adaptively determined based on the amount of scene change as well as parallax effects, and is typically one to a few seconds in length or about 30 to 100 frames, is registered using Algorithm 4 given below. Once a video segment has been registered then it can flow into a processing chain for object detection, tracking

Algorithm 3 RANSAC-based homography estimation

```
Input : A set of $n$ 2D point correspondences $(\mathbf{x}_i, \mathbf{x}'_i)$, where $n >= 4$
Output : Homography matrix A such that $\mathbf{x}'_i = A\mathbf{x}_i$
 1: $N \leftarrow 1000$   {a temporary number of iterations}
 2: $maxIterations \leftarrow 1000$   {maximum number of iterations}
 3: $d_\epsilon \leftarrow 0.005$   {distance threshold between data point and the model}
 4: $p \leftarrow 0.99$   {probability of choosing at least one sample free from outliers}
 5: $trialcount \leftarrow 0$   {trial counter}
 6: $inlier\_best \leftarrow 0$  {max of inliers so far}
 7: while $(N > trialcount)$ and $(trialcount < maxTrials)$ do
 8:     Randomly choose four correspondences $\{(\mathbf{x}_i, \mathbf{x}'_i) | i = 1..4\}$ from the list of putative matches
 9:     Check whether these points are co-linear, if so, redo Step 8
10:     Compute the homography $A$ by using normalized DLT (Alg. 2) from the four correspondences
11:     Compute $m$ as number of inliers where $\|\mathbf{x}'_i - A\mathbf{x}_i\| + \|\mathbf{x}_i - A^{-1}\mathbf{x}'_i\| < d_\epsilon$
12:     if $m > inlier\_best$ then
13:         $inlier\_best \leftarrow m$
14:         $\epsilon \leftarrow 1 - m/n$
15:         $N \leftarrow \frac{log(1-p)}{log(1-(1-\epsilon)^4)}$
16:     end if
17:     $trialcount \leftarrow trialcount + 1$
18: end while
19: Least squares estimate of $A$ from all correspondences classified as inliers using normalized DLT (Alg. 2)
```

and verification [43]. Chunks of video segments can be analyzed in a similar fashion, interconnected and summarized.

4.2 Implementation Details

4.2.1 Minimizing Accumulation of Transformation Errors

In the straightforward Method 1 illustrated in the figure 4.7 (based on 4.9) adjacent transformation matrices are multiplied to determine the cumulative transformation from Frame t to Frame t-k. The problem with Method 1 is that error accumulation is very rapid even with multiplying as few as 10 adjacent frame transformations. We have observed several pixel

Algorithm 4 UAV video registration algorithm.

```
1: Compute image spatial gradients $I_x$, $I_y$ and trace of the color structure tensor matrix
   $\mathbf{J}_C$ at every pixel (Eq.4.1)
2: Threshold on $trace(J_C(X)) > th$ to remove noise pixels, low confidence pixels and
   homogeneous regions.
3: For each remaining (non-zero) feature pixel, compute the Beltrami color metric tensor
   (i.e. determinant term) at each potential feature point $X_i$ using Eq. 4.2.
4: Establish highly confident PF regions for registration or tracking. Divide the image
   into $16 \times 16$ non-overlapping macroblocks, $B_k$, and classify prominent feature blocks
   based on a high PF value, which measures the percent cornerness that can be used as a
   block confidence measure for weighted least squares.
5: For each prominent feature block in current frame, $t$, find the best match in the previous
   frame, $(t-1)$, by maximizing NCC or minimizing SAD search (in intensity).
6: optional: Compute the direction histogram of the motion vectors obtained in Step 5,
   and apply motion filtering.
7: Estimate the homography between adjacent frames $t$ and $(t-1)$, $A_{(t-1,t)}$ by using the
   RANSAC (Alg. 3), DLT (Alg. 2) and normalization (Alg. 1) algorithms.
8: Compute the homography to warp from frame $t$ to frame $(t-k)$, $A_{(t-k,t)}$, and associate
   it with frame $t$.
9: Warp current image $I(x,y,t)$ into the coordinate system of the active video segment
   base (or reference) frame $I(x,y,t-k)$ using the homography from Step 8.
```

errors over less than 10 frames using double precision matrix multiplication with manually selected highly precise feature point matching between pairwise adjacent frames.

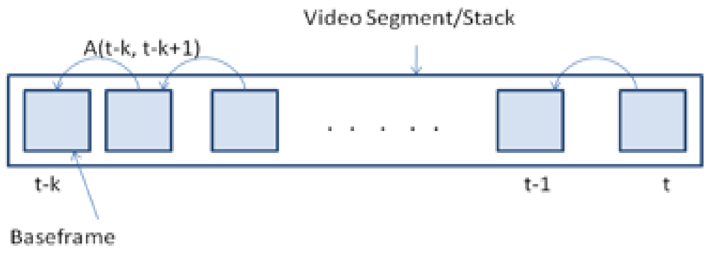


Figure 4.7: Method 1: Transformation across adjacent set of frames

To prevent rapid accumulation of errors we recommend using a transformation frame groups. As shown in the Figure 4.8, with say $N = 10$ there are at most $\lfloor k/N \rfloor + 1$ matrix multiplies instead of k matrix multiplies which significantly reduces the numerical error accumulation. In method 2, within a group first a transformation is obtained with respect

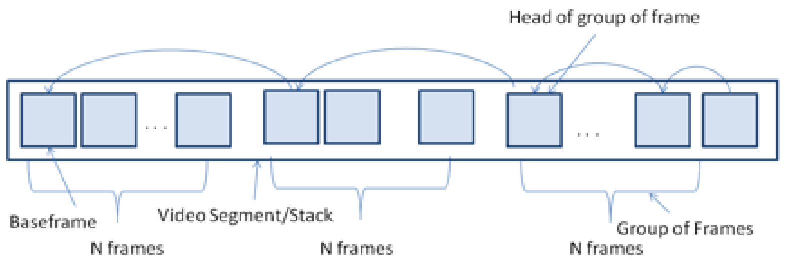


Figure 4.8: Method 2: Transformation across groups of frames

to the adjacent frame, then to the baseframe followed by successive transformations with respect to the baseframe of the previous groups.

From the idea of method 2, we further developed an adaptively changing reference frame scheme. Using a transformation frame group can prevent transformation error from being accumulated too quickly, however, the selection of reference frame (the first frame in a transformation group) and a proper interval between transformation groups become crucial. A bad reference frame, eg. highly blurred, will result in bad registration result in all successive frames. With fixed reference frame interval, its hard to find a balance between accuracy and how much camera motion is allowed. Small frame interval can deal with large camera motion but needs more matrix multiplication and large frame interval cannot handle big camera motion.

We solved such difficulties by dynamically selecting reference frame based on camera motion and current frame quality. In section 4.1.3, the displacement of each feature block with respect to reference feature block can be measured using either NCC or SAD. Therefore, the average displacement can be a measurement of camera motion. When the average displacement is larger than a threshold, then we change the reference frame. Therefore, for those images with smaller camera motion, the reference frame interval will be large, which can reduce the number of matrix multiplication, and for images with large camera motion, smaller frame interval will allow the registration program to keep working well. In section 4.1.4, we use RANSAC algorithm to filter out outliers so the ratio of inliers/outliers can be easily obtained. The selection of reference frame will also based on this inliers/outliers ratio. Low ratio means the matching between current frame and the reference frame is not trustworthy and we should not select current image as new reference frame. In this case, we will look at the next frame and the previous frame, either of the frame that has a higher inlier/outlier ratio larger than a threshold will be selected as new reference frame and this step continues until a new reference frame is found.

4.2.2 Single Global Coordinate System vs Multiple Sliding Coordinate Systems

A single global system as shown in the Figure 4.9 is best for tracking especially if it is georectified to a basemap. The dashed lines shown the location of image frames within the single global coordinate system and the solid line marks the ground projected trajectory of a single moving target within this coordinate system. But transforming imagery from an arbitrary camera pose on a moving platform to a georectifed coordinate system with adequate accuracy is difficult to do, especially if the motion is large, field of view is changing rapidly, or if registration cannot be done for some subset of frames due to sensor noise, dropped frames or other imaging distortions. In such cases a sliding or translating coordinate system is preferred and is easier to maintain in realtime. In this current work we use a moving coordinate system and maintain the homographies to move between the local or raw coordinate system and any other camera pose. This can be used in the future to move to a single georectified or georeferenced global coordinate system.

An important requirement for the flux algorithm is that there should be overlapping frames equal to the temporal filter size between two adjacent stacks as shown in the figure below. This implies that the registration module has to output the overlapping frames in the coordinate systems of both the previous and the new stack. The temporal filter size will be given as input to the registration module so that it knows how many overlapping frames to use between adjacent stacks for motion estimation.

Moving to a new coordinate system is controlled by two factors: (1) the current frame when transformed into the common coordinate system is too close to the edge of this image mosaic common/reference system, or (2) the maximum number of frames per video segment is reached.

4.2.3 Reducing False Alarms by Removing Border Effects in Flux Tensor Detection Results

Registered images under same coordinate system are shifting in the local coordinate system. Therefore, there will be significant changes near image borders, which causes border effects(false positive detections) in flux tensor detection results, as shown in Figure 4.10(b) and Figure 4.11(b). In order to reduce such border effects, only overlapping regions inside

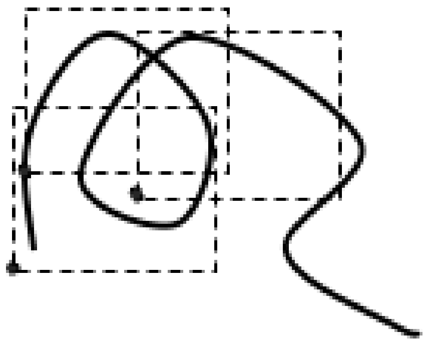


Figure 4.9: Registering or mosaicing to a single global coordinate system.

each flux tensor time step window are used for moving object detection. In addition, since foreground objects in aerial images are usually small, connected component analysis is also used to remove large false positive detection blobs, which further improves the detection accuracy, as shown in Figures 4.10 and 4.11.

4.2.4 Transform Flux Tensor Foreground Detection Result Back to Original Coordinate System

Foreground detection using flux tensor is performed using registered images. So foreground detection results are in transformed coordinate system. In order to obtain foreground detection results of original images, inverse transformations are performed on flux detection results to transfer them back to the original coordinates systems. As mentioned previously, we are using transformation frame groups with 10 frames per group to reduce the number of matrix multiplications. In addition, in order to prevent the accumulation of numerical errors, we also change the base frame every 30 frames. Therefore, to transform flux detection result back to origin, there will be at most 5 matrix multiplications and 1 inverse transform. At the begining, the papped images from image registration are directly used for motion detection and inverse transformation to obtain motion detection result in original coordinate system. However, the homography are calculated from original images. Therefore applying the homography on padded images produces wrong inverse transformation result that the inverse transform errors keeps on accumulating as shown in Figures 11(d), 12(d) and 16. After realizing that, in order to transform flux tensor foreground detection results back to the correct original coordinate systems, the four corners and their coordinates of the registered images are recorded instead of the four corners of the padded image. The results after correction are shown in Figures 11(e), 12(e) and 17.

4.3 Experimental Results

The proposed HPF-based image registration and flux tensor motion detection algorithm is tested using sequences from VIRAT dataset. Figure 4.10 (a) and 4.11 (a) show visual results of registered image. By applying inverse transform, we transform the moving objects detection results from registered image coordinate system back to the original image coordinate system so that they can be overlapped with original images. Results are shown in Figures 4.12 and 4.13 where the red blobs are corresponding to foreground detection results.As we can see, the detection blobs are shifted after applying inverse transform, which is due to accumulated numerical errors from sequential homography multiplications described in Sections 4.1.4 and 4.2.1. We also show a sequence of foreground motion detection results on a VIRAT ground-based stationary camera video dataset in Figure 4.15.

In order to measure the residual differences between the original image I and the inverse transformed registered frame I_{inv}, RMSE(root mean square error) is used as

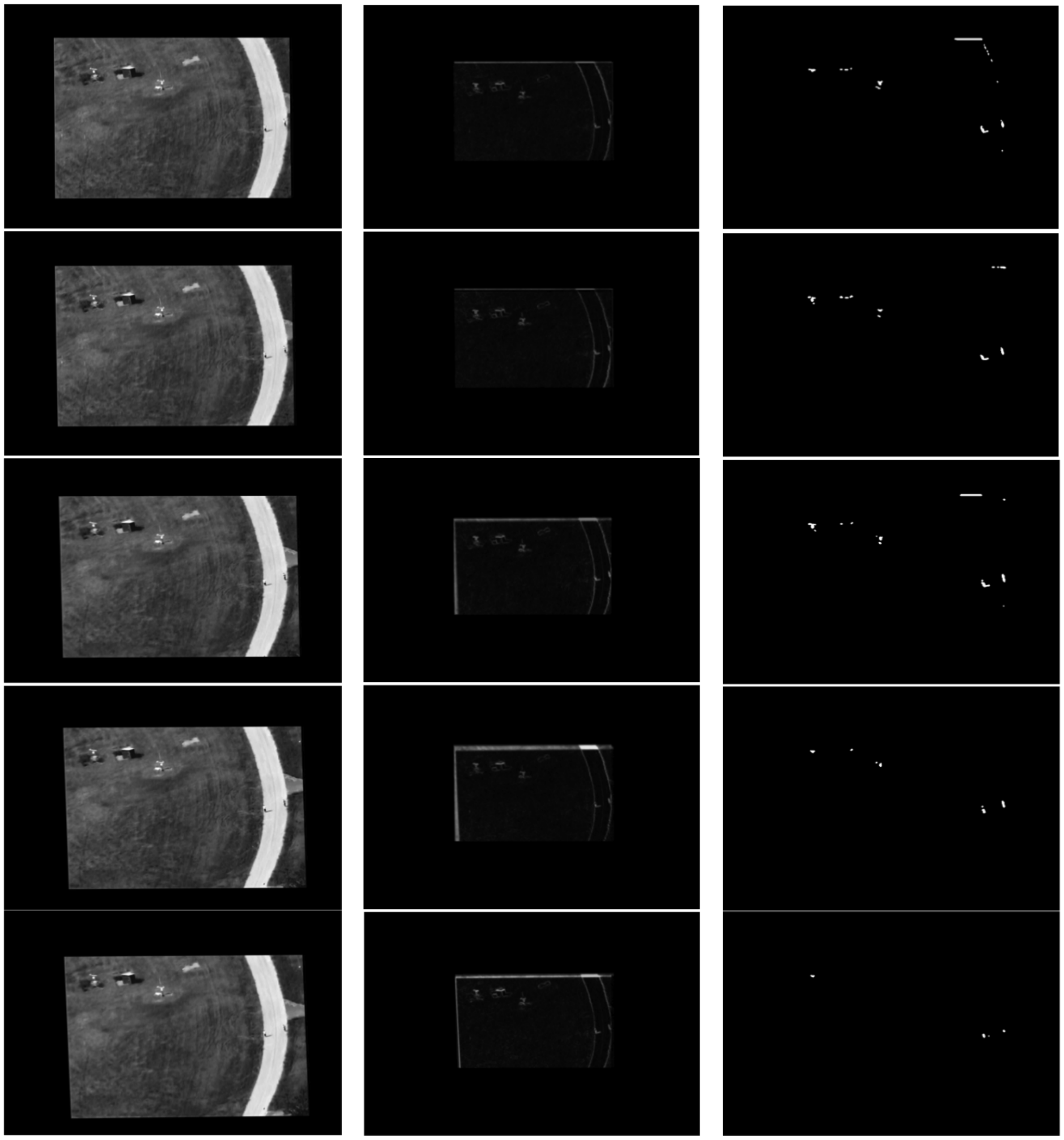
$$\text{RMSE} = \sqrt{\frac{1}{\#pixels}\sum_{x,y\in I}[I(x,y) - I_{inv}(x,y)]^2} \tag{4.20}$$

Another measurement showing how the accumulation of numerical errors affects the foreground object detection results is done by measuring the Euclidean distance between the centroid of detected object and the corresponding true object location(from groundtruth). The centroid-to-centroid Euclidean distance (CCED) is calculated in both registered image coordinate systems and original coordinate systems. The CCED errors of all objects to their true locations in each frame are shown in Figure 4.17. When flux tensor fails to detect moving object, the CCED error will raise to some large number, eg. larger than 60 pixels.

As we mentioned in section 4.2.1 the registration accumulation error can be minimized by using dynamic changing reference frame instead of using fixed reference frame interval.

To show the superiority of dynamic changing reference frame, we measured the CCED error between registered frame and base frame using both dynamic changing reference frame and fixed reference frame interval techniques. Figure 4.18 shows the comparison result.

To further reduce the accumulation error, the registration is restart after around 100 frames or when the registered image is out of the canvas. Figure 4.19 shows that by changing base frame, we can avoid the accumulation of registration error effectively.



(a) Registered frames (b) Flux trace (c) Thresholded flux

Figure 4.10: (a) Sample registered images from VIRAT Sequence 1 corresponding to frame numbers (from top to bottom): 96,100,104,108,112. (b) Corresponding frame motion energy based on flux trace response. (c) Corresponding frame motion-based foreground detection result after thresholding.

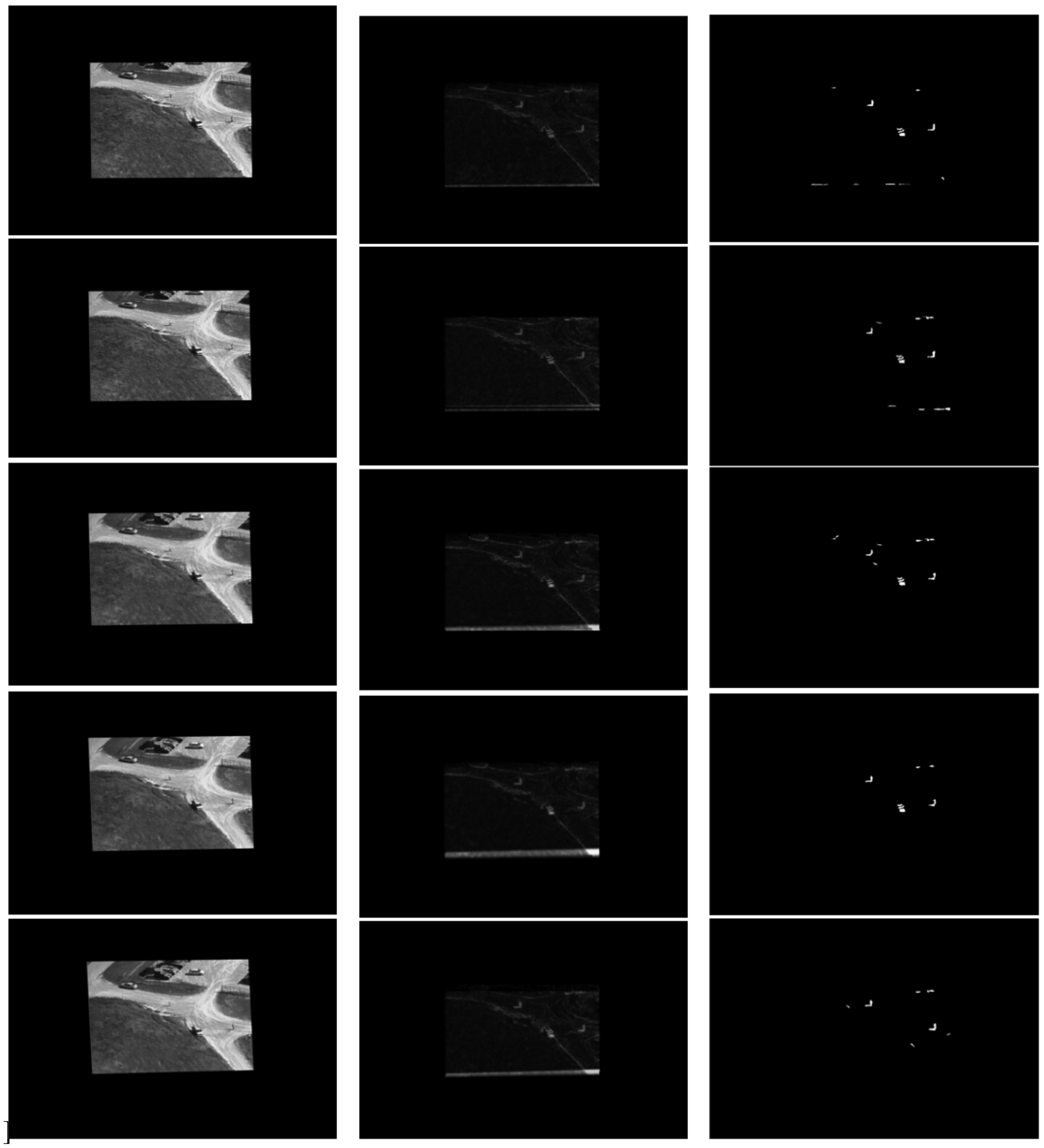
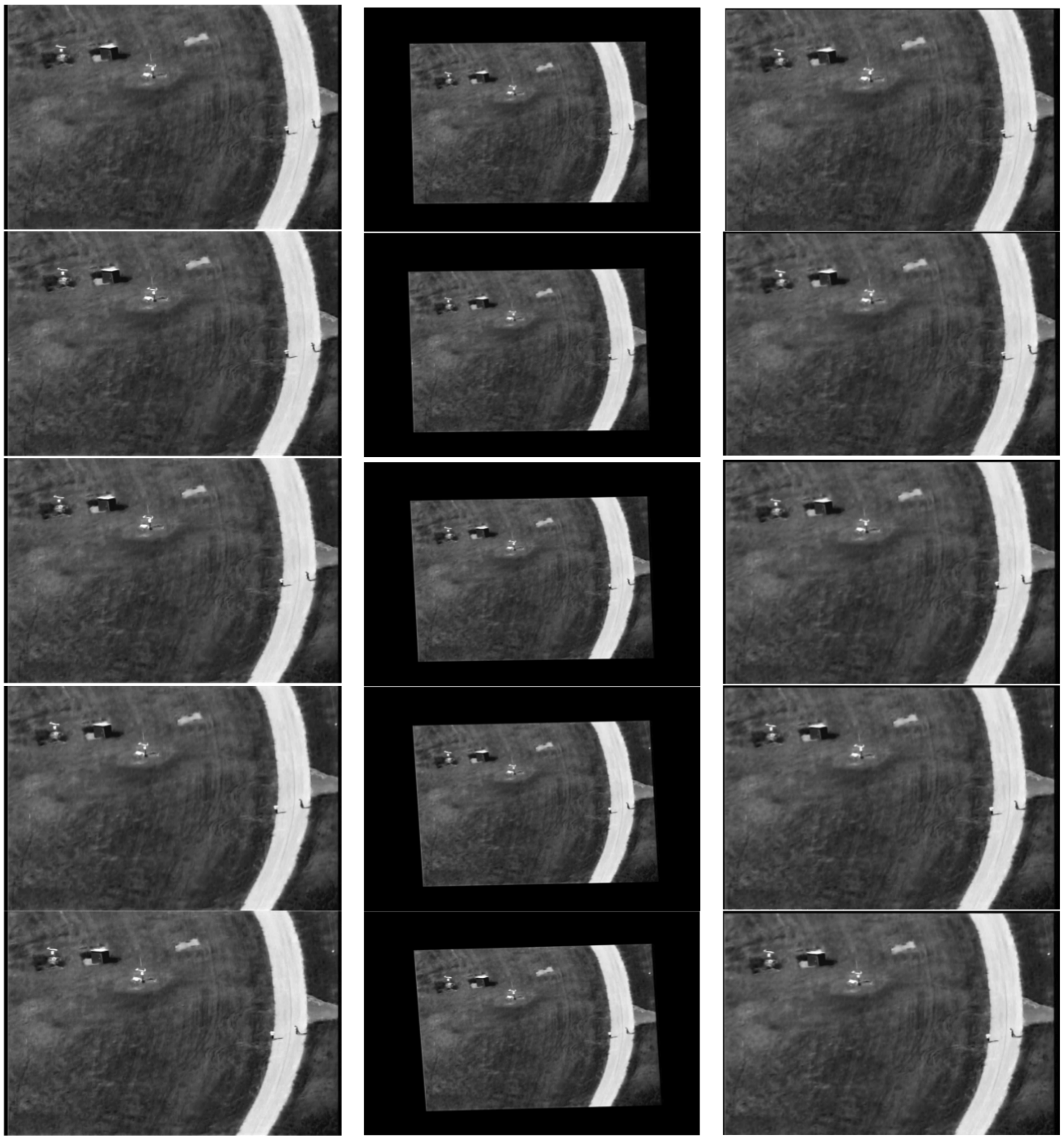
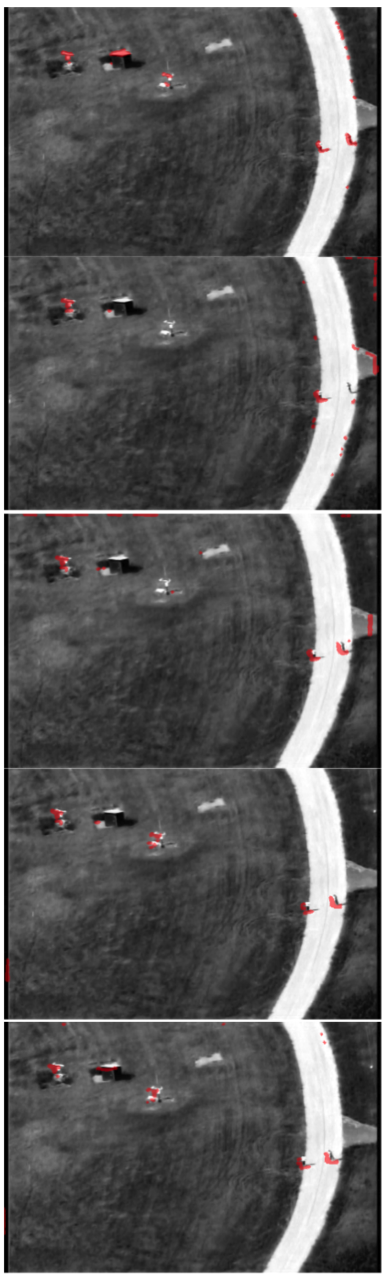

(a) Registered frames (b) Flux trace (c) Thresholded flux

Figure 4.11: (a) Sample registered images from VIRAT Sequence 3 corresponding to frame numbers (from top to bottom): 186,190,194,198, 202. (b) Corresponding frame motion energy based on flux trace response. (c) Corresponding frame motion-based foreground detection result after thresholding.



(a) Original frame (b) Registered frame (c) Inverse transform of registered frame

Figure 4.12: (a) Sequence of images from VIRAT Sequence 1 for frame numbers (from top to bottom): 139, 143,147, 151, 155. (b) Registered images in the base frame (global) coordinate system. (c) Inverse transformation back to raw original (local) coordinate system.


(d) Incorrect inverse transform of flux

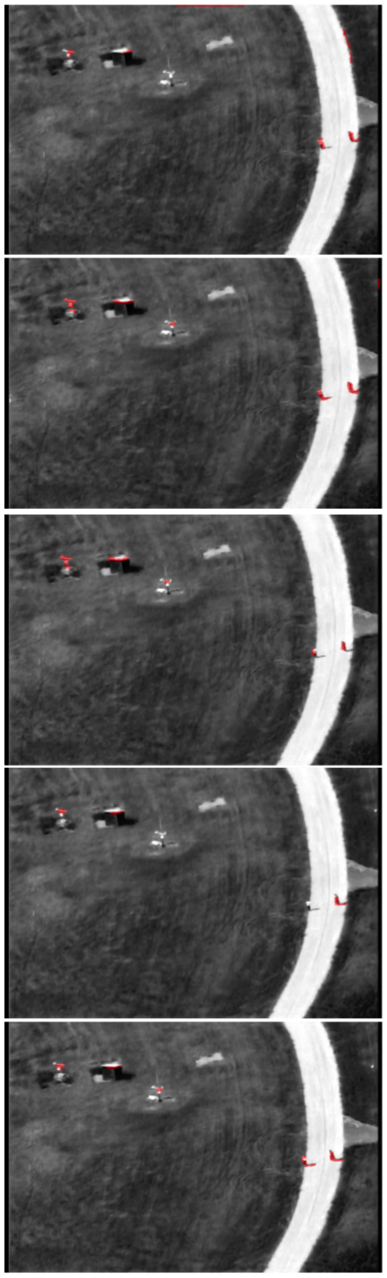
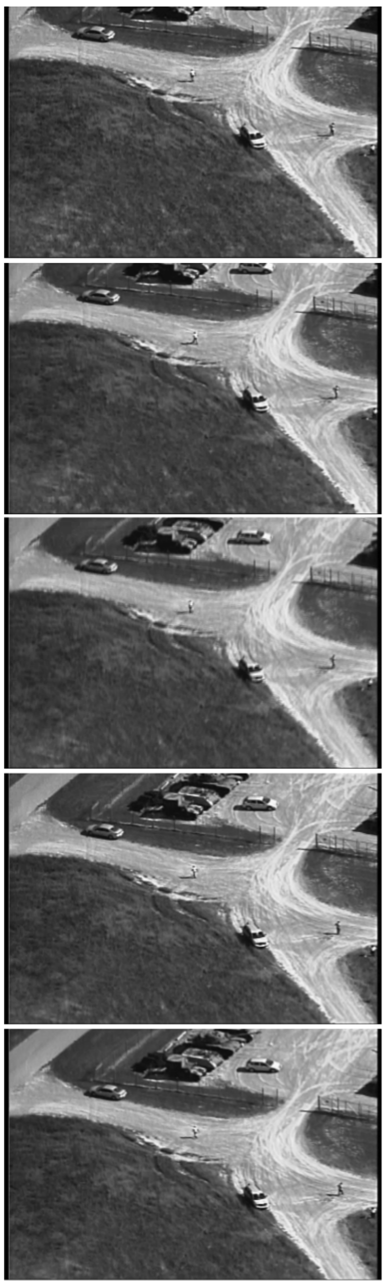
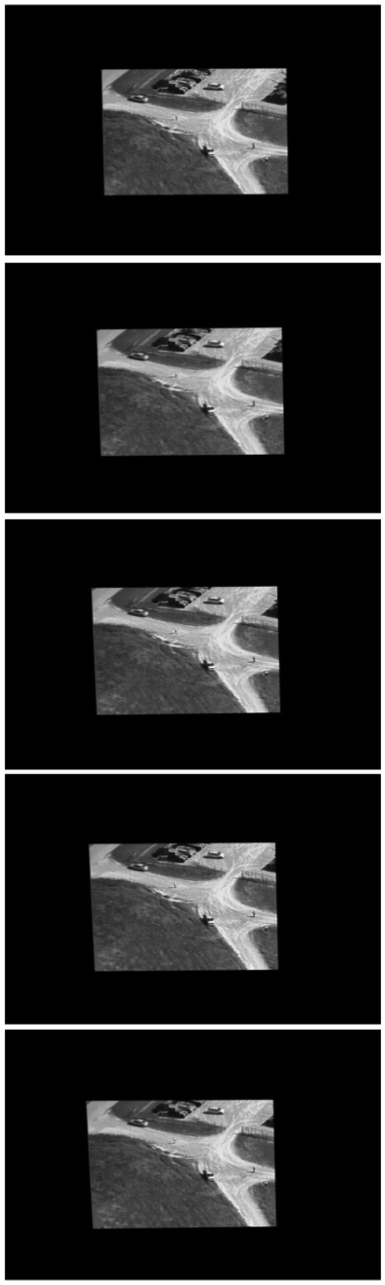

(e) Corrected inverse transform of flux

Figure 4.12: (d) Flux detection results superimposed on original frames for VIRAT Sequence 1 (frames 139, 143,147, 151, 155) with an incorrect offset in the inverse transformation. (e) Flux detection results after applying correct offset inverse transformation. The corrected inverse transform has a higher SNR and so a higher threshold was used which produces cleaner flux response and tighter more compact blobs. Note that both incorrect and correct offset transformation flux results have missed detections in second and fourth rows respectively. For missed detections the centroid-to-centroid error is based on the closest blob.


(a) Original frame


(b) Registered frame

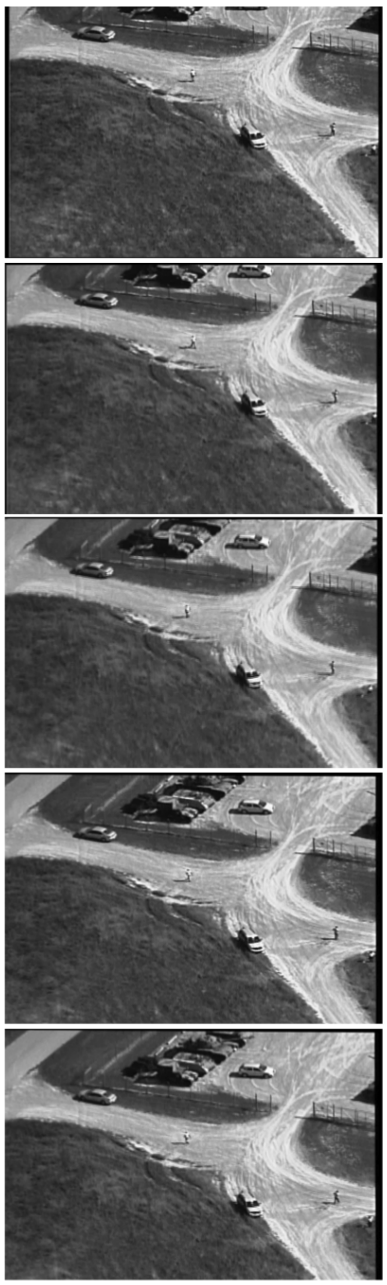
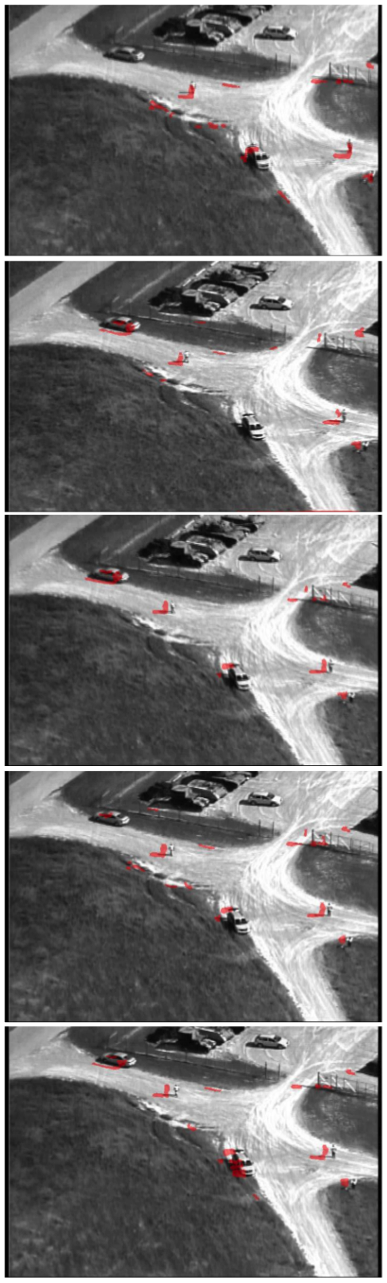

(c) Inverse transform of registered frame

Figure 4.13: (a) Sequence of images from VIRAT Sequence 3 for frame numbers (from top to bottom): 184, 191,198, 205, 212. (b) Registered images in the base frame (global) coordinate system. (c) Inverse transformation back to raw original (local) coordinate system.


(d) Incorrect inverse transform of flux result

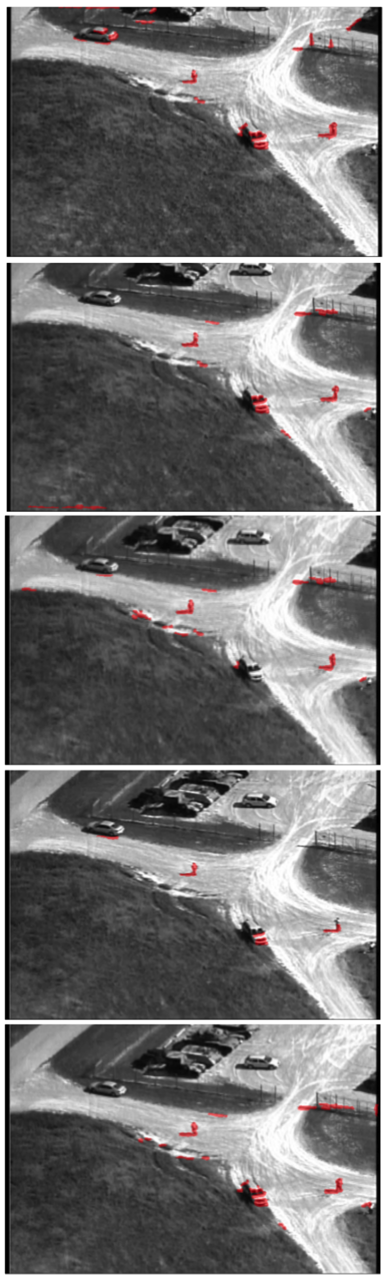
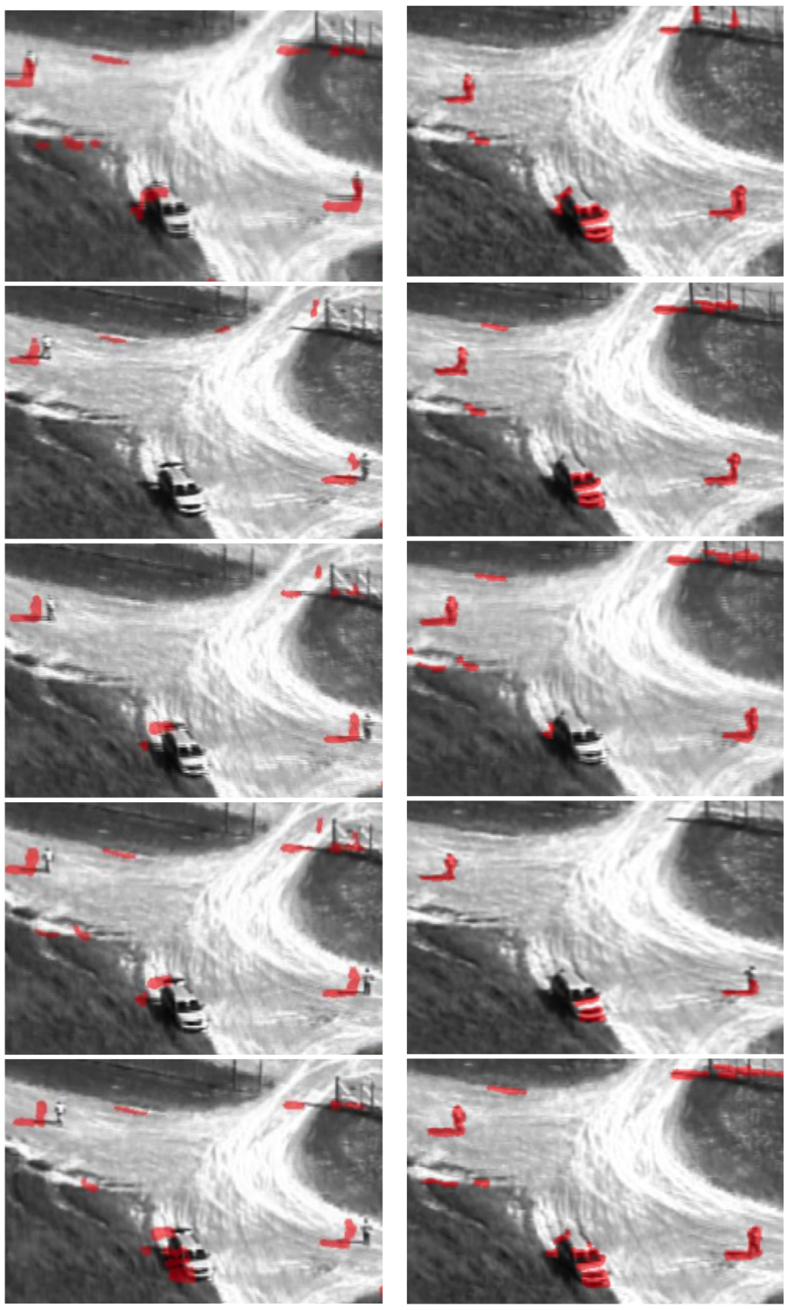
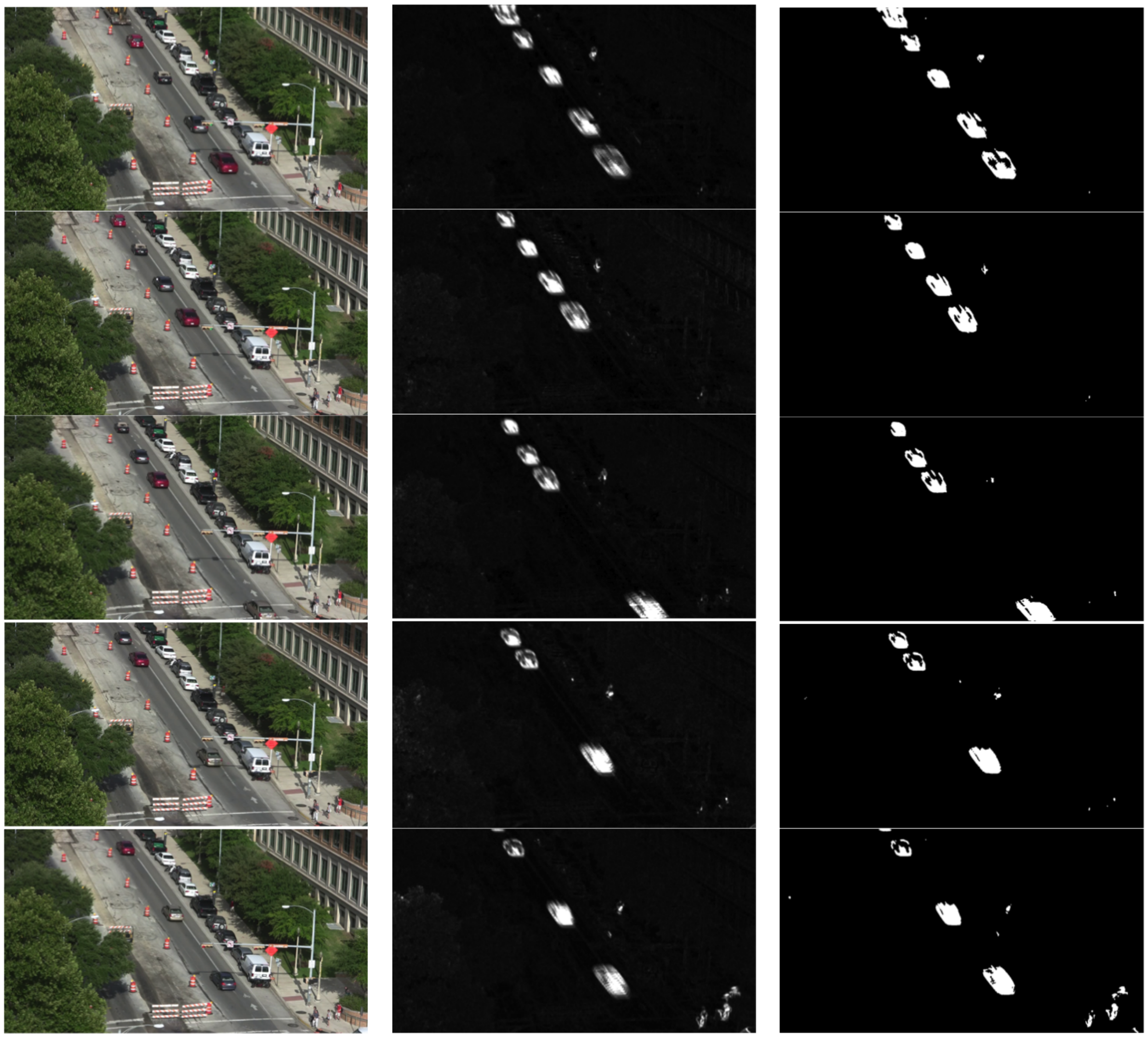

(e) Correct inverse transform of flux result

Figure 4.13: (d) Flux detection results superimposed on original frames for VIRAT Sequence 3 (frames 184, 191,198, 205, 212) with an incorrect offset in the inverse transformation. (e) Flux detection results after applying correct offset inverse transformation. The corrected inverse transform has a higher SNR and so a higher threshold was used which produces cleaner flux response and tighter more compact blobs. Note that in this example with two walking dismounts there are extra false detections but no missed detections.



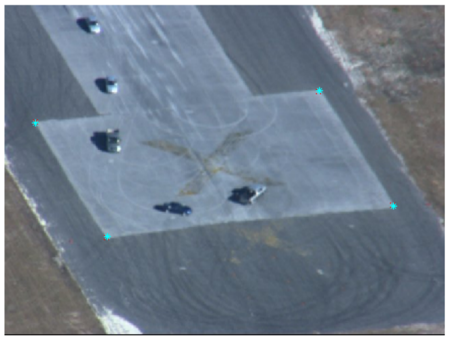
(a) Original frame (b) Registered frame

Figure 4.14: Zoom in of inverse transformed foreground detection result overlay with original image before and after correct inverse transformation.

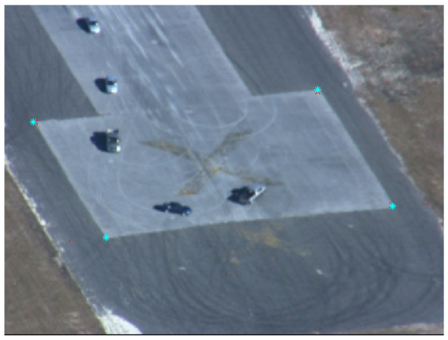


(a) Original frames (b) Unthresholded flux results (c) Motion detection results

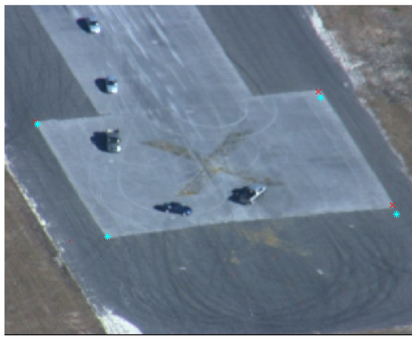
Figure 4.15: Moving object detection in DARPA VIRAT ground-based stationary camera video sequence VIRAT_S_050000_05_000696_000732 showing sample frames (top to bottom): 136,166, 196, 226, 256.



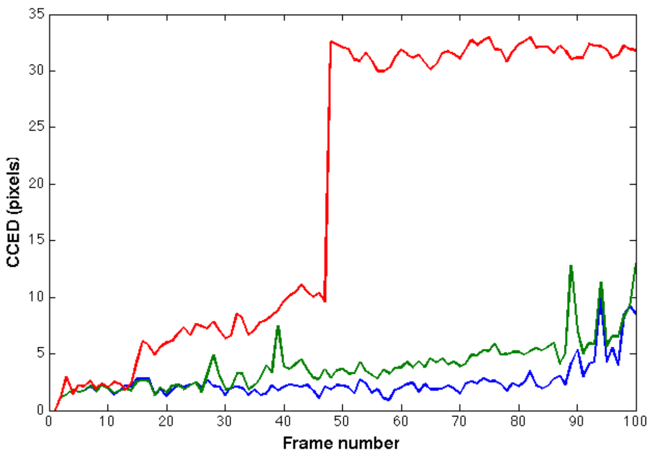
(a) Frame 1-10



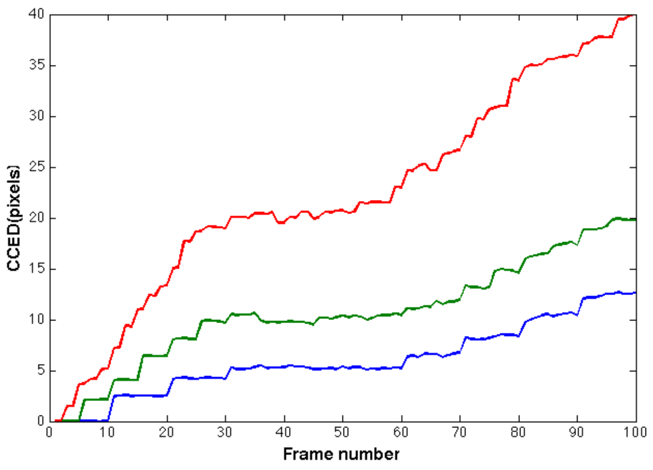
(b) Frame 1-50



(c) Frame 1-100

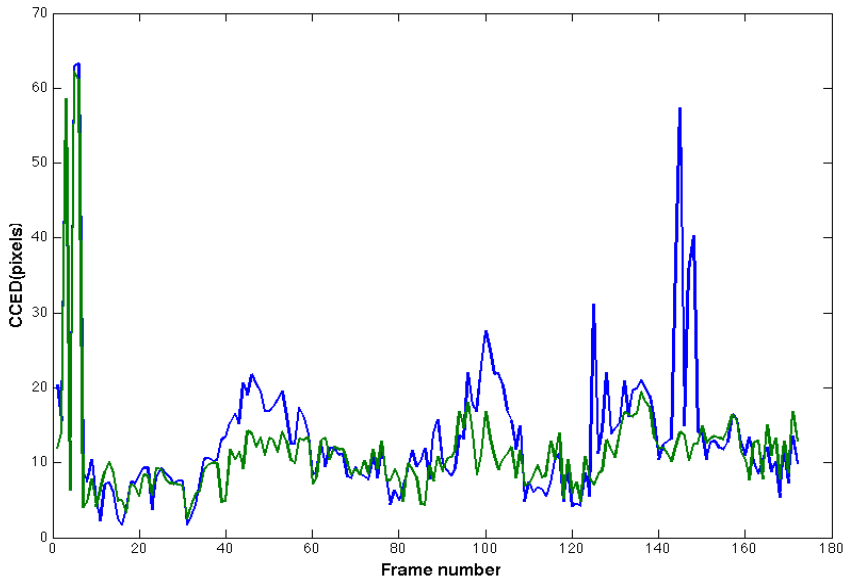


(d) Estimated homography

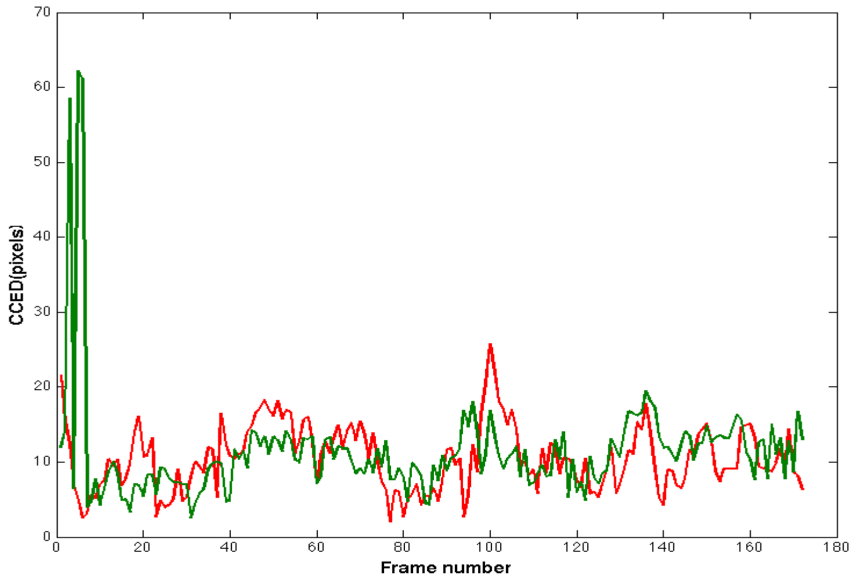


(e) Manually computed homography

Figure 4.16: (a) Reference points in base frame shown as red crosses and registered points transformed back to the base frame shown as blue stars using estimated homographies from the RANSAC algorithm. (b) RMSE in pixels between reference points and registered points using **estimated homographies**. Blue curve is when the reference base frame is updated every 10 frames, the green curve is when the reference base frame is updated every 5 frames and red curve is when the reference base frame is updated every 2 frames. (c) RMSE in pixels between reference points and registered points using **ground-truth homographies**. Blue curve is when the reference base frame is updated every 10 frames, the green curve is when the reference base frame is updated every 5 frames and red curve is when the reference base frame is updated every 2 frames. **Fewer homography matrix multiplications or larger interval between base frames leads to better accuracy.**

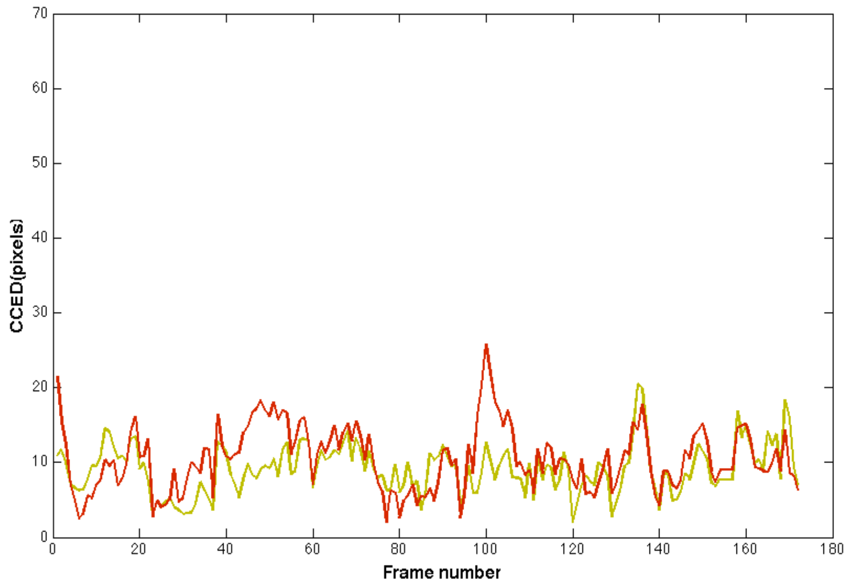


(a) Before offset correction



(b) After offset correction

Figure 4.17



(c) After offset correction

Figure 4.17: Centroid to centroid Euclidean distance error over 180 frames for VIRAT Seq 1. Green curve is for the stabilized (or registered) frames in the base frame coordinate system; the base frame coordinate system changes every 30 frames and the homography multiplication reference frames is updated every 10 frames. The green curve corresponds to the middle row in Figure 21. The five to ten pixel error is due to the imprecision in the flux tensor blob centroid localization shifting/variability and not the homography. The blue curve is measured in the original coord system after inverse transformation from the registered to the original frame. The large errors in the first few frames is due to the object blobs being removed from the post processing so that there is a missed fg object detection. The deviation of the blue curve which is in the original coordinate system (inverse transformed) from the green curve is due to the incorrect offset transformation. After the offset transformation was corrected the errors are essentially eliminated as shown in the graph in the second row. The red curve is corresponding to correctly inverse transformed sequence. The green curve is the same curve in the first row. The third row shows the CCED based on the motion detection sequence using correct inverse transformaiont. The yellow curve is for the stablized frames same as the green curve in (a) but in different detection results. Larger errors in red curve are because of numerical errors when doing inverse transformation, which is smaller than using incorrect inverse transformation shown in (a). The average error of the blue curve in (a) is 13.38, the average error of the green curve in (a) and (b) is 11.28, the average error of the red curve in (b) and (c) is 10.46 and the average error of the yellow curve in (c) is 9.05.

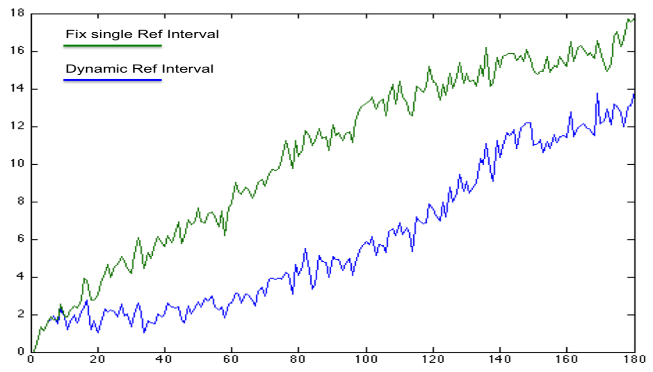


Figure 4.18: Dynamic changing reference frame vs. fixed reference frame interval

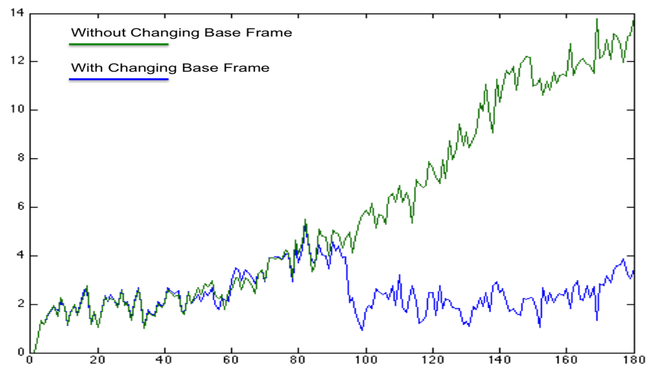


Figure 4.19: Reduce registration accumulation error by changing base frame.

Chapter 5

Summary and Concluding Remarks

We described a moving object detection system that combines spatio-temporal tensor-based motion estimation with a novel background modeling scheme. Use of tensor-based motion segmentation results in coherent detections robust to noise and illumination artifacts, while the proposed background subtraction process handles detection of static objects. The final multi-cue object level classification distinguishes stopped objects from background revealed by removed objects and thus reduces false positives. We experimentally show that the proposed system outperforms most state-of-the-art methods on the CVPR2014 challenge dataset [2].

In addition, we also developed a moving object detection framework for aerial images, which get significant results. However, there are still a lot of limitations for our system. The accumulation of numerical errors produced by sequential homography multiplication makes the registered images not fully stable, which causes false positive detections in flux tensor results. The foreground objects sizes in aerial images are usually very small, which makes them very hard to be detected. In addition, the motion of the camera results in blurred or interlaced images, which causing problems in both image registration and motion detection. In order to obtain better knowledge of how numerical errors accumulates, we will produce synthetic dataset with known camera motion to help quantify the errors. Multi-scale flux tensor will be implemented to handle foreground objects with different sizes in future work.

Bibliography

[1] Tom SF Haines and Tao Xiang. Background subtraction with dirichlet processes. In *Computer Vision–ECCV 2012*, pages 99–113. Springer, 2012.

[2] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection.net: A new change detection benchmark dataset. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 1–8, June 2012.

[3] Chris Stauffer and W. E L Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages –252 Vol. 2, 1999.

[4] R.J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: a systematic survey. *Image Processing, IEEE Transactions on*, 14(3):294–307, 2005.

[5] Sebastian Brutzer, Benjamin Hoferlin, and Gunther Heidemann. Evaluation of background subtraction techniques for video surveillance. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1937–1944. IEEE, 2011.

[6] Yannick Benezeth, Pierre-Marc Jodoin, Bruno Emile, Hélène Laurent, and Christophe Rosenberger. Review and evaluation of commonly-implemented background subtraction algorithms. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[7] Y.Z Hsu, H.-H Nagel, and G Rekers. New likelihood test methods for change detection in image sequences. *Computer Vision, Graphics, and Image Processing*, 26(1):73 – 106, 1984.

[8] V. Mahadevan and N. Vasconcelos. Spatiotemporal saliency in dynamic scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):171–177, 2010.

[9] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 255–261 vol.1, 1999.

[10] Stephen J. McKenna, Sumer Jabri, Zoran Duric, Azriel Rosenfeld, and Harry Wechsler. Tracking groups of people. *Computer Vision and Image Understanding*, 80(1):42 – 56, 2000.

[11] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31 Vol.2, 2004.

[12] Marko Heikkilä, Matti Pietikäinen, and Janne Heikkilä. A texture-based method for detecting moving objects. In *BMVC*, pages 1–10, 2004.

[13] Shengping Zhang, Hongxun Yao, and Shaohui Liu. Dynamic background modeling and subtraction using spatio-temporal local binary patterns. In *Image Processing,*

2008. ICIP 2008. 15th IEEE International Conference on, pages 1556–1559. IEEE, 2008.

[14] Shengcai Liao, Guoying Zhao, Vili Kellokumpu, Matti Pietikainen, and Stan Z Li. Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1301–1306. IEEE, 2010.

[15] Ahmed Elgammal, David Harwood, and Larry Davis. Non-parametric model for background subtraction. In *Computer VisionECCV 2000*, pages 751–767. Springer, 2000.

[16] Yosuke Nonaka, Atsushi Shimada, Hajime Nagahara, and Rin-ichiro Taniguchi. Evaluation report of integrated background modeling based on spatio-temporal features. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 9–14. IEEE, 2012.

[17] Liyuan Li, Weimin Huang, I.Y.-H. Gu, and Qi Tian. Statistical modeling of complex backgrounds for foreground object detection. *Image Processing, IEEE Transactions on*, 13(11):1459–1472, 2004.

[18] Du-Ming Tsai and Shia-Chih Lai. Independent component analysis-based background subtraction for indoor surveillance. *Image Processing, IEEE Transactions on*, 18(1):158–167, 2009.

[19] Olivier Barnich and Marc Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *Image Processing, IEEE Transactions on*, 20(6):1709–1724, 2011.

[20] Martin Hofmann, Philipp Tiefenbacher, and Gerhard Rigoll. Background segmentation with feedback: The pixel-based adaptive segmenter. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 38–43. IEEE, 2012.

[21] Robert T Collins, Alan Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, Osamu Hasegawa, Peter Burt, et al. *A system for video surveillance and monitoring*, volume 2. Carnegie Mellon University, the Robotics Institute Pittsburg, 2000.

[22] Paolo Spagnolo, TD' Orazio, Marco Leo, and Arcangelo Distante. Moving object segmentation by background subtraction and temporal analysis. *Image and Vision Computing*, 24(5):411–423, 2006.

[23] Rubén Heras Evangelio and Thomas Sikora. Complementary background models for the detection of static and moving objects in crowded environments. In *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, pages 71–76. IEEE, 2011.

[24] Filiz Bunyak, Kannappan Palaniappan, Sumit Kumar Nath, and Gunasekaran Seetharaman. Flux tensor constrained geodesic active contours with sensor fusion for persistent object tracking. *Journal of Multimedia*, 2(4):20, 2007.

[25] S. Nath and K. Palaniappan. Adaptive robust structure tensors for orientation estimation and image segmentation. In *LNCS-3804: Proc. ISVC'05*, pages 445–453, Lake Tahoe, Nevada, Dec. 2005.

[26] H.H. Nagel and A. Gehrke. Spatiotemporally adaptive estimation and segmentation of OF-Fields. In *LNCS-1407: ECCV98*, volume 2, pages 86–102, Freiburg, Germany, June 1998. Springer-Verlag.

[27] B.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, Aug. 1981.

[28] K. Palaniappan, Hai Jiang, and Tobias I Baskin. Non-rigid motion estimation using the robust tensor method. In *IEEE Comp. Vision and Pattern Recog. Workshop on Articulated and Nonrigid Motion*, Washington, DC, June 2004.

[29] J. Zhang, J. Gao, and W. Liu. Image sequence segmentation using 3-D structure tensor and curve evolution. **11**(5):629–641, May 2001.

[30] H. Scharr. Optimal filters for extended optical flow. In *LNCS: First Int. Workshop on Complex Motion*, volume 3417, pages 66–74, Berlin, Germany, Oct. 2004. Springer-Verlag.

[31] H. Scharr, I. Stuke, C. Mota, and E. Barth. Estimation of transparent motions with physical models for additional brightness variation. In *13th European Signal Processing Conference, EUSIPCO*, 2005.

[32] Jing-Ming Guo, Chih-Hsien Hsia, Yun-Fu Liu, Min-Hsiung Shih, Cheng-Hsin Chang, and Jing-Yu Wu. Fast background subtraction based on a multilayer codebook model for moving object detection. *Circuits and Systems for Video Technology, IEEE Transactions on*, 23(10):1809–1821, Oct 2013.

[33] Hanzi Wang and D. Suter. Background subtraction based on a robust consensus method. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 223–226, 2006.

[34] 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.

[35] Yannick Benezeth, Pierre-Marc Jodoin, Bruno Emile, Christophe Rosenberger, and Hlne Laurent. Comparative study of background subtraction algorithms. *Journal of Electronic Imaging*, 19(3):033003–033003–12, 2010.

[36] K. Mikolajczyk and *et. al* . A comparison of affine region detectors. *Int. J. Computer Vision*, 65(1/2):43–72, 2005.

[37] F. Bunyak, K. Palaniappan, S. Nath, and G. Seetharaman. Geodesic active contour based fusion of visible and infrared video for persistent object tracking. In *8th IEEE Workshop on Applications of Computer Vision (WACV 2007)*, page Online, Austin, TX, Feb. 2007.

[38] F. Bunyak, K. Palaniappan, S. Nath, and G. Seetharaman. Fux tensor constrained geodesic active contours with sensor fusion for persistent object tracking. *J. Multimedia*, 2(4):20–33, August 2007.

[39] G. Seetharaman, G. Gasperas, and K. Palaniappan. A piecewise affine model for image registration in 3-d motion analysis. In *IEEE Int. Conf. On Image Processing*, pages 561–564, Vancouver, BC, Canada, Sep. 2000.

[40] L. Zhou, C. Kambhamettu, D. Goldgof, K. Palaniappan, and A. F. Hasler. Tracking non-rigid motion and structure from 2D satellite cloud images without correspondences. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(11):1330–1336, Nov. 2001.

[41] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[42] Hanzi Wang, D. Mirota, and G.D. Hager. A generalized kernel consensus-based robust estimator. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):178–184, 2010.

[43] Z. Yue, D. Guarino, and R. Chellappa. Moving object verification from airborne video. In *IEEE Int. Conf. Computer Vision Systems (ICVS)*, page Online, 2006.

VITA

I was born in Baoji, Shaanxi, China. I attended University of Electric Science and Technology of China (UESTC) in 2008 and I participated in a 2+2 program to enter University of Missouri, Columbia (MU) in 2010. I got my Bachelors' degree from both UESTC and MU in DEC 2012. I began doing research with Professor Kannappan Palaniappan since Feb 2011.

I will join a PhD program in University of North Carolina, Chapel-Hill after my graduation from MU.