MOTION DETECTION AND OBJECT TRACKING IN GRAYSCALE VIDEOS BASED ON SPATIOTEMPORAL TEXTURE CHANGES

A Dissertation
Submitted
to the Temple University Graduate Board

In Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

By
Roland Miezianko
January, 2006

ABSTRACT

MOTION DETECTION AND OBJECT TRACKING IN GRAYSCALE VIDEOS BASED ON SPATIOTEMPORAL TEXTURE CHANGES

Roland Miezianko

DOCTOR OF PHILOSPHY

Temple University, January, 2006

Dr. Longin Jan Latecki, Chair

Automatic detection and tracking of moving objects are the fundamental tasks of many video-based surveillance systems. Higher level security assessment and decision making procedures rely upon these essential video analysis tasks. Robust motion detection and object tracking provide the basis for detection of increased activity, entry into a restricted area, detection of objects left behind, tracking of optical flow against established motion patterns, and other similar surveillance requirements. A common method for real-time segmentation of moving regions in image sequences involves modeling each pixel as a mixture of Gaussians and using K-means approximation to update the model. Each Gaussian distribution is assigned to represent the background or a moving object in the adaptive mixture model. Every pixel is then evaluated and classified as part of a moving region or as a background. This method does not adapt

well to graylevel videos since the color information is limited to a single value. The proposed motion detection and object tracking method is particularly suitable for grayscale videos, such as infrared, thermal, and converted color image sequences. Detection of moving objects in grayscale videos is based on changing texture in parts of the field of view. The proposed method estimates the speed of texture change by measuring the spread of texture vectors in the feature space. It robustly detects very fast and very slow moving objects. The theoretical and experimental results show that measuring spread of texture vectors significantly outperforms the Stauffer-Grimson approach based on Gaussian mixture model. The proposed spatiotemporal motion detection method does not require any post-processing, which is a necessary step required by the Gaussian mixture model. Additionally, real-time software designed to detect motion based on the spatiotemporal texture changes is evaluated using color and infrared cameras. The proposed selective hypothesis tracking method is fundamentally based on the location of spatiotemporal texture motion regions. It uses predicted motion vectors, sub-pixel image registration, and minimum cost estimation using distance, direction, size, and persistence. This method is capable of tracking fast and slow moving objects, objects that disappear and later reappear, and objects that merge and split.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Longin Jan Latecki for his support and invaluable guidance during this research. His mentorship and advice were instrumental in the course of this exploration. Much thanks also goes to Dr. Dragoljub “Pokie” Pokrajac for sharing the details and results of his research on motion detection. I am also very grateful to the members of the Examining Committee: Dr. Rolf Lakaemper, Dr. Slobodan Vucetic, and Dr. Marc J. Sobel.

To Monika.

TABLE OF CONTENTS

PAGE

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Motion detection and object tracking algorithms are an important research area of computer vision and comprise building blocks of various high-level techniques in video analysis that include tracking and classification of trajectories. It is an obvious and biologically motivated observation that the main clue for detection of moving objects is the changing texture in parts of the view field. All optical flow computation algorithms use derivative computation to estimate the speed of texture change [20]. However, derivative computation may be very unstable in finite domains of images. Therefore, a proposed motion detection method is introduced that does not require any derivative computation. An approach is proposed to perceive motion and detect activity based on statistical properties of texture vectors [4].

Let us focus on a fixed position in a video plane and observe the sequence of texture vectors representing a patch around this position over time. Each texture vector describes the texture of the patch in a single video frame. An assumption of a stationary camera is made. In the observed patch that corresponds to part of the background image, the texture vectors will not be constant due to various factors (e.g., illumination changes, errors of the video capture device), but combined effect is merely a small spread of texture vectors over time. Also a repetitive background motion like tree

branches waving in the wind yields a relatively small spread of texture vectors. Since similar texture repeats frequently, the texture vectors in this case are highly correlated.

On the other hand, if a moving object is passing through the observed location, it is very likely that object will have a different texture from the background patch. Therefore, the texture vectors are very likely to have a large spread. Even if different parts of the moving object have the same texture that is the same as the background texture, the texture vectors will have large spread at the observed location, since different texture parts will appear in the patch. This holds under the assumption that the texture is not completely uniform, since then different texture parts have different texture vectors. To summarize, the proposed approach can identify moving objects even if their texture is identical with the background texture, due to the fact that our classification is based on measuring the amount of texture change and texture structure is extremely unlikely to be perfectly uniform.

Observe that the spread of texture vectors is measured in the texture space. Because of this, the direct optical flow computation is not possible, i.e., to estimate the directions and speed of moving objects. However, we are able to perform robust detection of moving objects. In comparison to the existing motion detection algorithms [6, 7, 14], no model of the background is computed. Only the measurement of the amount of texture change is computed and then classified it into two categories: moving and stationary objects. The afore-mentioned situation in which the background texture and the texture of moving object are similar illustrates a typical situation in which the proposed approach outperforms any background modeling method. In such cases, in the

background modeling approaches the texture of a moving object can be easily misclassified as background texture.

Instead of color, graylevel, or infrared values at pixel locations, the values of all pixels in spatiotemporal regions represented as 3D blocks are considered. These 3D blocks are represented through compact spatiotemporal texture vectors to reduce the influence of noise and decrease computational demands. As shown in [11] the use of such texture vectors in the framework of Stauffer and Grimson [14] can improve the detection of moving objects while potentially cutting back the processing time due to the reduction of the number of input vectors per frame. Thus, we go away from the standard input of pixel values for motion detection that are known to be noisy and the main cause of instability of video analysis algorithms. The proposed motion detection technique is independent of any particular texture representation used.

To represent texture, consider the values of all pixels within spatiotemporal regions represented as 3D blocks. A 3D block (e.g., $8\times8\times3$) consists of a few successive frames (e.g., 3) at the same quadratic patch (8×8) of a scene. To compactly represent these values and to reduce the influence of noise, a dimensionality reduction technique is applied by using principal components projection (PCA). As the result, texture is represented by a vector containing only the most significant projected components of texture, while less significant components and noise are filtered out through the process of feature extraction. The most significant projected components represent a small subset of all the projections. The obtained texture vectors provide a compact low dimensional joint representation of texture and motion patterns in videos,

and are used as primary inputs to a motion detection technique. As mentioned above, texture at a given location in video plane is very likely to considerably vary while a moving object is passing through this location. Measure this variance by estimating the covariance matrix of the texture vectors from the same location within a window of a small number of successive frames is computed, and determine the texture spread as the largest eigenvalue of the covariance matrix. This allows to indirectly determine the magnitude of texture variability in the direction of its maximal change. Finally, the decision whether a moving object or a stationary background is identified at a given spatiotemporal location is made by dynamic distribution learning of the obtained largest eigenvalue.

The proposed technique can use a variety of video sequences as input, ranging from monochromatic grayscale, thermal, infrared (IR) videos to multispectral videos in visible or IR spectral domain. To demonstrate the usefulness of the proposed method, several benchmark videos from PETS workshop are used. The robust performance of the proposed motion detection method provides the basis to detect increased activity in videos. Motion amount is defined as a sum of motion activates of all blocks in a given frame (Chapter 5). By applying a simple statistical learning of the motion amount, increased activities may be detected and classified. Learning of the distribution of the total motion amount in all previous frames is necessary, under the assumption that mostly normal activities are present. An increased activity is detected as outlier of the learned distribution.

The proposed approach to increased activity detection does not include any specific domain knowledge about the monitored objects. Such knowledge can be incorporated in the framework, e.g., focusing on monitoring only human or vehicle activities. By adding a classifier that is able to label moving object categories, it can restrict system's attention to particular object categories, e.g., see [18].

The usefulness of dimensionality reduction techniques to compactly represent 3D blocks has already been recognized in video compression. There, 3D discrete cosine and 3D wavelet transforms are employed to reduce the color or graylevel values of a large number of pixels in a given block to a few quantized vector components, e.g., [15]. However, these techniques are not particularly suitable for detecting moving objects, since the obtained components do not necessarily provide good means to differentiate the texture of the blocks. Namely, these transformations are context free and intrinsic in that their output depends only on a given input 3D block. In contrast, the proposed technique allows obtaining an optimal differentiation for a given set of 3D blocks. To reach this goal, an extrinsic and context sensitive transformation such that a representation of the given block depends on its context is needed. The Principal Component Analysis (PCA) [8] satisfies these requirements. Namely, for a given set of 3D blocks PCA assigns to each block a vector of the components that maximize the differences among the blocks. Consequently, PCA components are very suitable to detect changes in 3D blocks.

1.1 Related Work

A good overview of the existing approaches to motion detection can be found in the collection of papers edited by Remagnino et al. [13] and in the special section on video surveillance in IEEE PAMI edited by Collins et al. [2]. A common feature of the existing approaches for moving objects detection is the fact that they are pixel based. Some of the approaches rely on comparison of color or intensities of pixels in the incoming video frame to a reference image. Jain et al. [7] use simple intensity comparison to reference images so that the values above a given threshold identify the pixels of moving objects. A large class of approaches is based on appropriate statistics of color or graylevel values over time at each pixel location. (e.g., the segmentation by background subtraction in W4 [6], eigenbackground subtraction [10], etc). Wren et al. [16] were the first who used a statistical model of the background instead of a reference image.

One of the most successful approaches for motion detection was introduced by Stauffer and Grimson [14]. It is based on adaptive Gaussian mixture model of the color values distribution over time at each pixel location. Each Gaussian function in the mixture is defined by its prior probability, mean and a covariance matrix. It is shown that the proposed local variation is not only a much simpler but also a more adequate model for motion detection for graylevel videos. It can significantly reduce the processing time in comparison to the Gaussian mixture model, due to smaller complexity of the local variation computation, thus making the real time processing of high resolution videos as well as efficient analysis of large scale video data viable.

Moreover, the local variation based algorithm remains stable with higher dimensions of input data, which is not necessarily the case for an Expectation-Maximization (EM) type algorithm (used for Gaussian model estimation). This makes the proposed technique potentially appealing for moving detection in higher dimensional domains, such as multispectral remote sensing imagery.

As argued in [9], the application of region level techniques can lead to increased stability when detecting objects in adverse conditions. However, [9] and related approaches (e.g., [1]) aimed to improve the Stauffer-Grimson algorithm [14] still perform motion detection on pixel level. Only the post processing of pixel-based motion detection results is region based. In contrast, the motion detection in the proposed approach is solely region-based.

1.2 Background Subtraction Model

The most basic form of motion detection is the method of subtracting know background image containing no objects from an image under test. There are several methods to background subtraction, including averaging background frames over time and statistical modeling of each pixel. Example of background subtraction of two frames is shown in Figure 1.1. Whereas Figure 1.2 shows some of the difficulties in using this method for motion detection. It shows the difficulty in selecting an appropriate threshold. All above the mean pixel values are scaled to the maximum value of 255. Figure 1.2 shows that other methods must be used to threshold the image.

Figure 1.1: Temple 1 background subtraction images. Two consecutive frames from Temple1 video showing motion, frame 219 and frame 220.

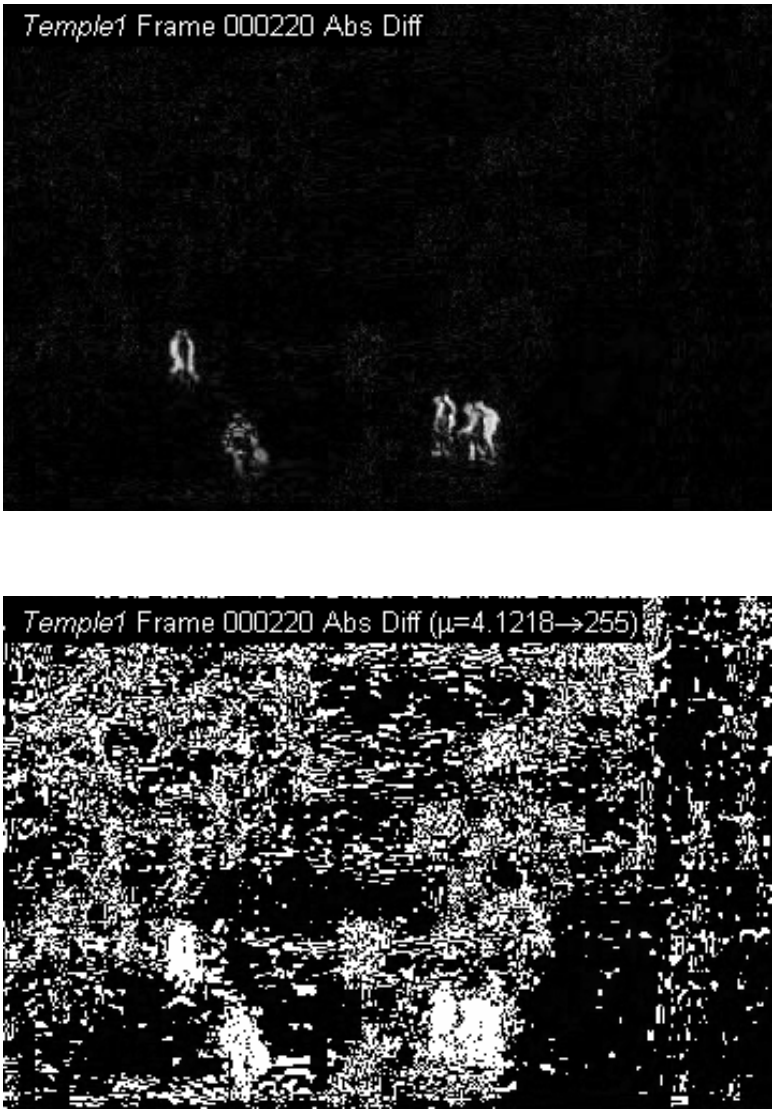

Figure 1.2: Temple 1 background subtraction absolute difference. Difference between frame 220 and 219 from Temple 1 video where each pixel greater than mean difference value is scaled to 255, showing the unreliable nature of the background subtraction model.

1.3 Gaussian Mixture Model

Model the values of a particular pixel as a mixture of Gaussian distributions. Multiple adaptive Gaussians are necessary to cope with acquisition noise, lighting changes, and other natural occurrence. Pixel values that do not fit the background distributions are considered foreground. This is a common method for real-time segmentation of moving regions in image sequences. Model Gaussians are updated using *K*-means approximation method. Each Gaussian distribution is assigned to represent the background or a moving object in the adaptive mixture model. Every pixel is then evaluated and classified as a moving region or as a background. This method also requires post-processing for graylevel videos whereas the method proposed in Chapter 2 does not.

Incremental model of Gaussian distribution is defined as

$$p(x) = \prod_{i=1}^{K} w_i \frac{1}{\sqrt{(2\pi)^d \det(\sum_i)}} e^{-\frac{(x-\mu_i)\Sigma_i(x-\mu_i)^T}{2}}$$

where

μ_i are the mean vectors

Σ_i are the covariance matrices

w_i are the priors.

Example of Gaussian model showing and classifying a pixel as motion is shown in Figure 1.3. Actual distributions of one pixel 185x213 from Campus 1 video is shown in Figure 1.4 (Red-Green) and Figure 1.5 (Red-Green-Blue), were more than one background is very likely.

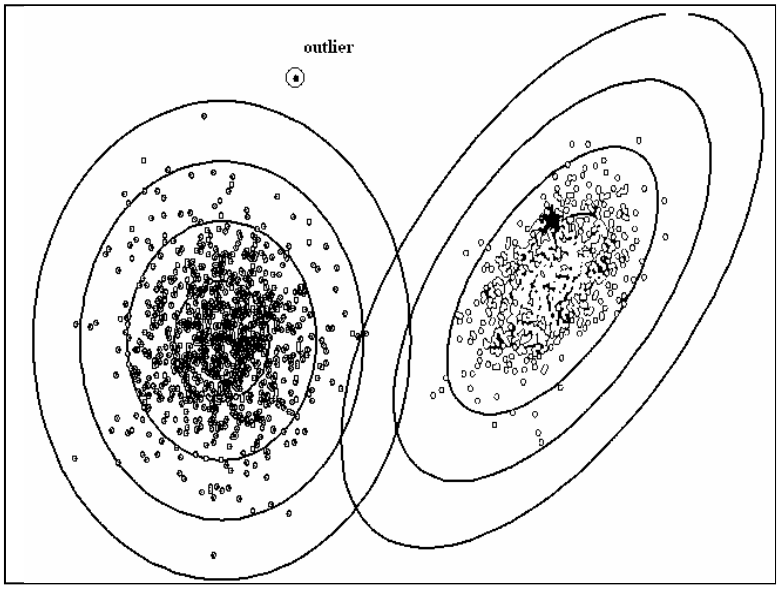

Figure 1.3: Outlier pixel in the Gaussian distribution. Distributions are shown as ellipses and the pixel outside the distribution is classified as motion.

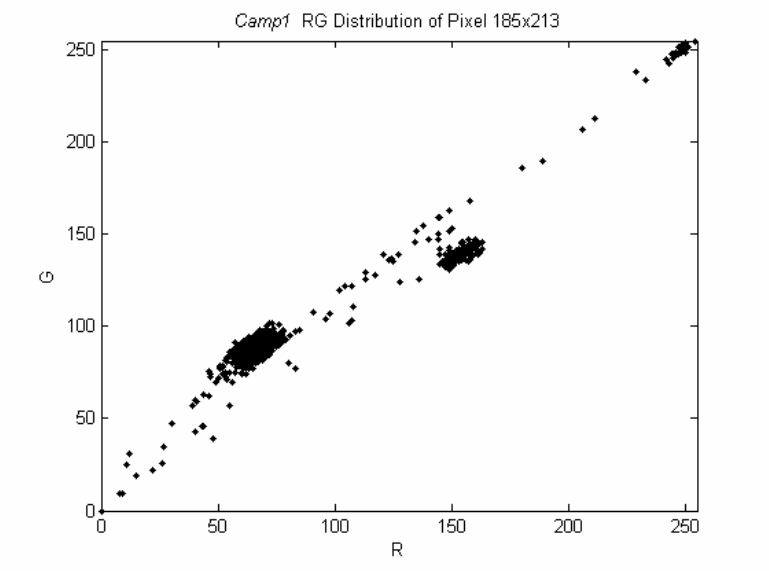

Figure 1.4: Campus 1 RG pixel distribution. Red-Green locations of pixel 185x213 from Campus 1 video showing more than one possible background.

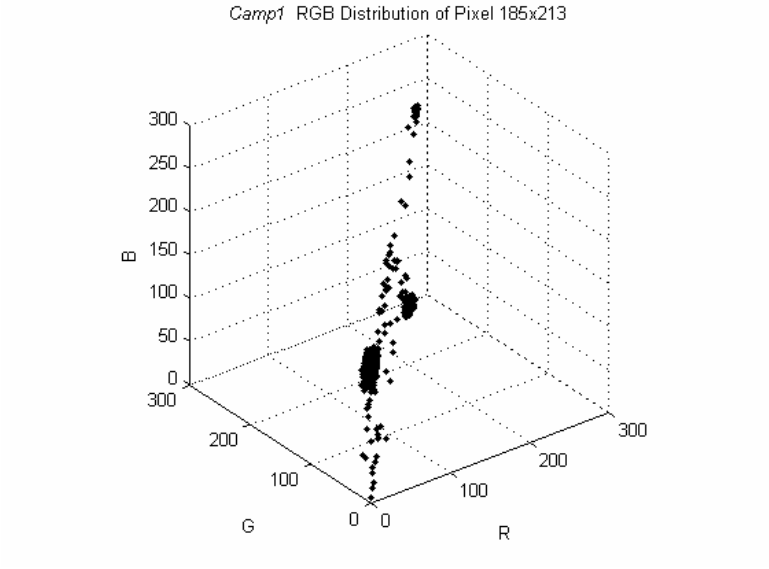

Figure 1.5: Campus 1 RGB pixel distribution. Locations of pixel 185x213 from Camp1 video showing more than one possible background.

CHAPTER 2

MOTION FEATURE REPRESENTATION

2.1 Spatiotemporal Texture Vectors

Videos are represented as three-dimensional (3D) arrays of monochromatic (infrared or graylevel) pixel values $g_{i,j,t}$ at a time instant t and a pixel location (i, j). A video is characterized by temporal dimension Z corresponding to the number of frames, and by two spatial dimensions, characterizing number of pixels in horizontal and vertical direction of each frame. Each image is divided in a video sequence into disjoint $N_{BLOCK} \times N_{BLOCK}$ squares (e.g., 8×8 squares) that cover the whole image. Spatiotemporal 3D blocks are obtained by combining squares in consecutive frames at the same video plane location. All experiments reported here use $8 \times 8 \times 3$ blocks that are disjoint in space but overlap in time, i.e., two blocks at the same spatial location at times t and $t+1$ have two squares in common. The fact that the 3D blocks overlap in time allows us to perform successful motion detection in videos with very low frame rate, e.g., in experimental results, videos with 2 to 30 frames a second are included. The obtained 3D blocks are represented as 192-dimensional $(8 \times 8 \times 3)$ vectors of monochromatic pixel values [21].

In general the blocks are represented by N-dimensional vectors $b_{I,J,t}$, specified by spatial indexes (I, J) and time instant t. Vectors $b_{I,J,t}$ contain all graylevel values $g_{i,j,t}$ of pixels in the corresponding 3D block. To reduce dimensionality of $b_{I,J,t}$ while preserving information to the maximal possible extent, we compute a projection of the normalized block vector to a vector of a significantly lower length $K \ll N$ using a PCA [8] projection matrix $P_{I,J}^{K}$ computed for all $b_{I,J,t}$ at video plane location (I, J). The resulting spatiotemporal texture vectors $b_{I,J,t}^{*} = P_{I,J}^{K} \bullet b_{I,J,t}$ provide a joint representation of texture and motion patterns in videos and are used as input of algorithms for detection of motion and objects tracking. Value of $K = 10$ is used in all experiments and value of $K = 3$ is used for simplicity in creating motion orbit graphs. The obtained K-dimensional vectors form a compact spatiotemporal texture representation for each block. It is important to notice that a different projection matrix $P_{I,J}^{K}$ is used for each video plain location. This assures that the obtained texture vectors are able to optimally distinguish different textures that appear in a given block. The initial projection matrix is trained on the first t_0 frames under the assumption that only background is present in all block locations. The projection matrices are then updated during the time periods in which no motion is detected in a given block location.

2.2 Detection of Moving Features by Measuring Texture Spread

The spread of texture vectors over time indicates whether the corresponding object texture is stationary or moving. Recall that each spatiotemporal vector represents

texture of the corresponding block. Hence, by observing the characteristics of spatiotemporal vectors change over time, we are able to detect whether a particular block belongs to a moving object or to a background. Consider a single block position in a video plane. We can observe the trajectory of its spatiotemporal vectors, i.e., the loci of spatiotemporal vectors in successive time frames, which we call motion orbits. For example, see Figure 2.1, where each point represents the first three PCA components of the texture vectors.

If during an observed time interval there is no moving object in the block, i.e., a stationary background is only present in the block, the spatiotemporal vectors will be close to each other. The background texture is represented by the large cluster of points as seen in Figure 2.1. In contrast, if there is a moving object passing through this block, the spatiotemporal texture vectors will change fast, that is, the spatiotemporal vectors will be spread in the space of their coordinates.

To summarize, it can be observed that frames with only stationary objects are visible in the observed block location correspond to regions where spatiotemporal vectors are clustered into fairly spherical shapes with small spread. In contrary, when moving objects are passing through this block location, the trajectory of spatiotemporal vectors is typically elongated and the variance is relatively large. A simple way to determine the speed of spatiotemporal vector change would be to compute the norms of their first derivatives. However, computing finite differences of consecutive spatiotemporal vectors may be unreliable. In order to determine whether the consecutive

vectors belong to elongated trajectories, we need to observe whether they are making a consistent progress in one particular direction within a certain time interval.

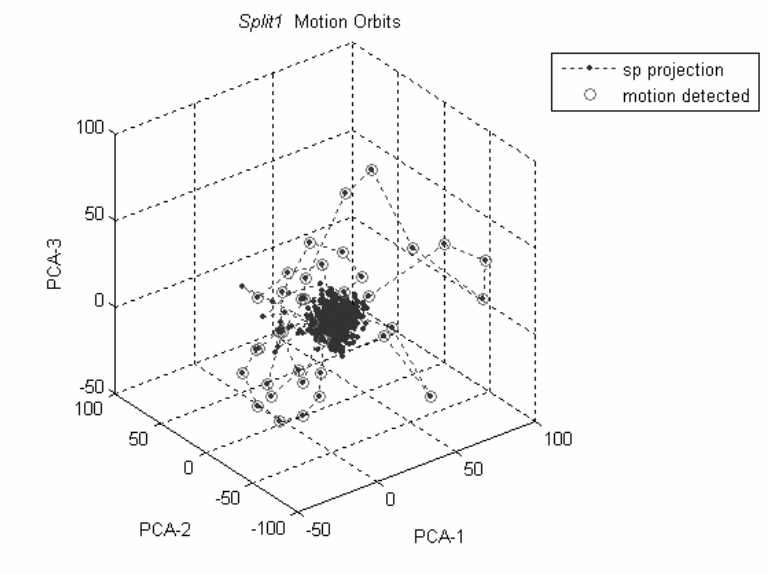

Figure 2.1: Split 1 motion orbits. Motion and background vectors of block (41,42) in Split 1 video.

This requires the assessment of the spatiotemporal vector spread in the direction of maximal variance. To measure the variance of spatiotemporal vectors, we compute the covariance matrix of spatiotemporal vectors corresponding to the same block location for a pre-specified number of consecutive frames. We use the maximal

eigenvalue as the measure of trajectory elongation. More formally, for each location (x, y), and temporal instant t, we consider vectors of the form

$$\left\langle b^*_{x,y,t-W}, b^*_{x,y,t-W+1}, \ldots, b^*_{x,y,t}, \ldots, b^*_{x,y,t+W} \right\rangle$$

corresponding to a symmetric window of size $2W+1$ around the instant t. For these vectors, we compute the covariance matrix $C_{x,y,t}$, and assign the largest eigenvalue of $C_{x,y,t}$, denoted as $\Lambda_{x,y,t}$, to a given spatiotemporal video position to define a local variance measure. This local variance measure is also called motion measure

$$mm(x, y, t) = \Lambda_{x,y,t}$$

The larger the motion measure $mm(x, y, t)$, the more likely is the presence of a moving object at position (x, y, t). An example graph of $mm(x, y, t)$ is shown in Figure 2.2. The large values (spikes) correspond to time intervals when moving objects where observed at this video location. The large values exactly correspond to the elongated motion orbits, while the small values correspond to the texture vectors within the background cluster.

For comparison, the infrared value of a pixel within block location 44x16 is shown in Figure 2.3. Due to a significant amount of noise, detection of moving objects

seems to be a very challenging if not impossible task when based on infrared pixel values. Distinct advantage to spatiotemporal block processing is evident from these figures, where motion is detected in block 44x16, yet the pixels inside that block show no relevant texture changes.

As the graph in Figure 2.2 suggests, we can label video position (x, y, t) based on the history of $mm(x, y, t)$ values over time (frames $(t, \ldots, t-1)$) as moving, by applying an outlier detection method to $mm(x, y, t)$ values, i.e., a position is labeled as moving if motion measure value at a given time is classified as an outlier. To perform the outlier detection, we first learn the nominal distribution of $mm(x, y, t)$ values over some initial time period $(t = 1, \ldots, t_1)$. This requires that the amount of unusual activity is relatively small in the initial time period, i.e., the part of the scene we mostly view at this location in the initial time period is stationary (background) .Then we use running average to update the mean and standard deviation of this distribution. The update is not performed if the position is classified as moving. A particular $mm(x, y, t)$ is classified as an outlier if it is further away from the mean than a certain number of standard deviations. The distribution learning and outlier detection algorithm is described in Section 2.3.

The motion measure value provides a classification method for each block as moving or stationary. This is the basis for activity detection, noise detection, and the selective hypothesis tracking algorithm.

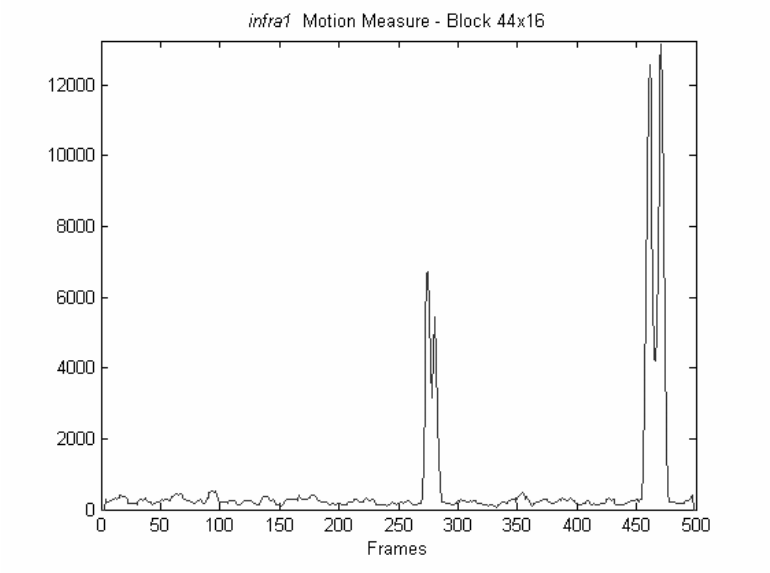

Figure 2.2: Infra 1 motion measure. Value for block 44x16 showing motion around frames 275 and 475.

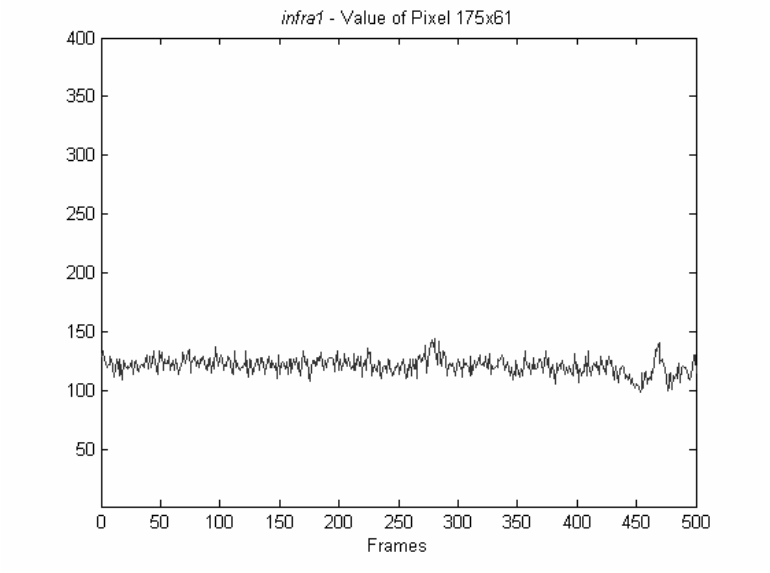

Figure 2.3: Infra 1 pixel values. Intensity of pixel 175x61 inside block 44x16 not showing any significant change.

2.3 Dynamic Distribution Learning and Outlier Detection

Consider labeling each video position as moving or stationary (background) based on whether the motion measure $mm(x, y, t)$ is larger or smaller than a suitably defined threshold. Dynamic distribution learning is used to determine the threshold value at position (x, y, t) based on the history of $mm(x, y, t)$ values over time (at frames $t, \ldots t-1$). Since $mm(x, y, t)$ is a function of one variable t for a fixed position (x, y) (see Figure 2.2), the task reduces to dynamic estimation of the mean and standard deviation of mm. The only assumption that is made about the distribution of values of function f is that it has a prominent right tail (such as a general Gaussian distribution).

Given a function f of one variable, we compute initial values of $mean(t_0)$ and variance $\sigma^2(t_0)$ of all values $f(t)$ in some initial interval $t = 1, \ldots, t_0$ [3,5]. For $t > t_0$, we update the estimates using the technique described in the next paragraph. An outlier is detected at time $t > t_0$ if the standardized feature value is sufficiently large, i.e., when

$$\frac{f(t) - mean(t-1)}{std(t-1)} > C_1$$

where C_1 is a constant and

$$std(t) = \sqrt{\sigma^2(t)}$$

Once an outlier is detected at time t_1, value $f(t_1)$ is labeled as an outlier. We update the nominal state at time t, if the standardized feature value drops below a threshold $C_2 < C_1$, i.e.,

$$\frac{f(t) - mean(t-1)}{std(t-1)} < C_2$$

We update the estimates of mean and standard deviation only when the outliers are not detected (nominal state), i.e., at the beginning of the execution of the algorithm and when feature value drops below a threshold. Then, $mean(t)$ and $std(t)$ are updated using running averages:

$$mean(t) = u \cdot mean(t-1) + (1-u) \cdot f(t)$$

$$\sigma^2(t) = u \cdot \sigma^2(t-1) + (1-u) \cdot (f(t) - mean(t-1))^2$$

$$std(t) = \sqrt{\sigma^2(t)}$$

For example, we use $C_1 = 9, C_2 = 3, u = 0.99$ in the case of the detection of moving blocks for $f = mm$. As stated before, the only assumption that we make about the distribution of values of function f is that it has a significant right tail. This

assumption clearly applies to the Gaussian distribution, but is significantly more general.

Figure 2.4 shows the detection of noisy frames in Temple 1 video [12] using the derivative of motion amount using the dynamic distribution learning algorithm. Figure 2.5 shows motion measure values of block 47x54 from Campus 1 video. In Figure 2.6 motion is detected also using dynamic distribution learning algorithm. The motion measure value is very small around frame 1500 relative to frames 800 or 2600, however, the dynamic distribution correctly identified motion around this location.

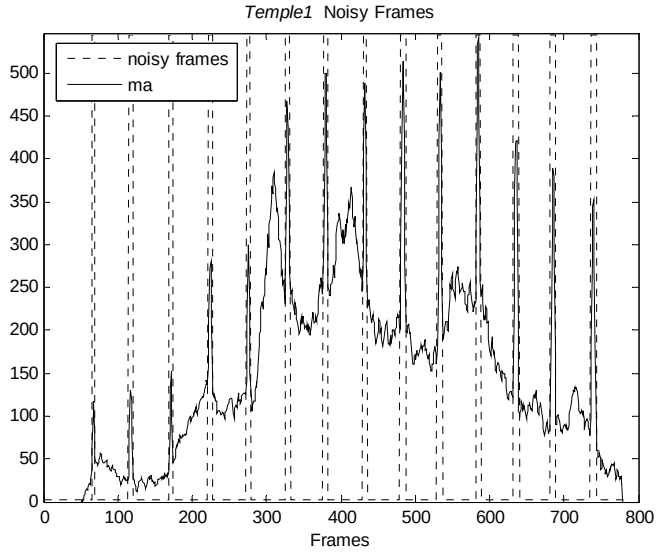

Figure 2.4: Temple 1 detected noisy frames. Motion measure with periodic spikes due to compression errors. Spikes are detected using dynamic distribution learning.

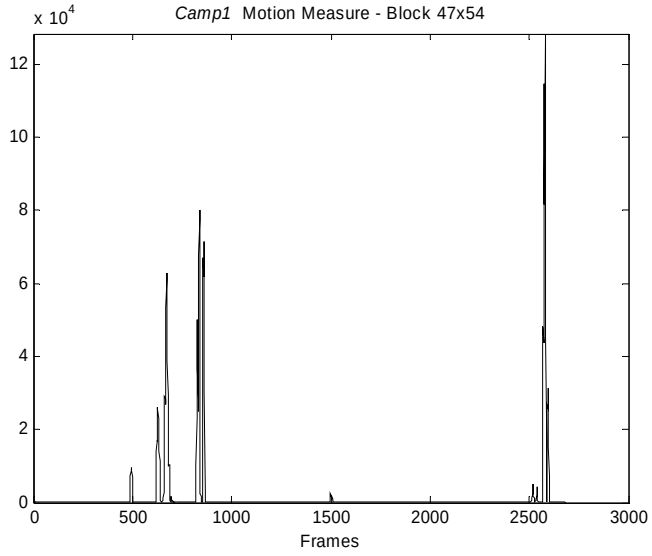

Figure 2.5: Campus 1 motion measure. Motion measure of block 47x54 showing large motion spikes.

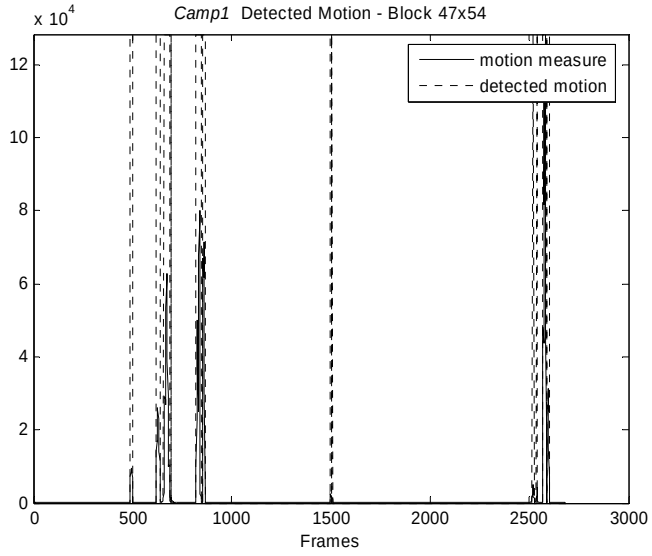

Figure 2.6: Campus 1 detected motion. Detected motion based on motion measure of block 47x54 and dynamic distribution learning.

CHAPTER 3

OBJECTIVE PERFORMANCE EVALUATION

3.1 Motion Orbits in Texture Space

The most common method to evaluate the performance of motion detection is simply to view the videos with moving objects marked by the applied algorithm as discuss in Chapter 2. However, a more objective method of performance evaluation is also possible. This section introduces and uses such a method to compare the proposed spread measure of texture vectors to the Gaussian mixture model introduced in [14]. To make the comparison more realistic, the Gaussian mixture model to texture vectors is applied. Hence, both compared techniques are based on the same spatiotemporal blocks that represent texture and motion patterns. We also show that the Gaussian mixture model on texture vectors significantly outperforms the original representation used in [14], RGB color values on a pixel level.

A motion orbit is defined as a path that the texture representation at the fixed video plane location traverses over time. Recall that texture vectors are composed of first 3 PCA components of each spatiotemporal block vector. Hence, the motion orbit at video plane location (x, y) is a sequence of points in the 3D Euclidean space $v_{x,y,1}, v_{x,y,2}, \ldots, v_{x,y,T}$ where $v_{I,J,t} = b^*_{I,J,t}$ and T is the total number of frames.

For instance, in Figure 3.1, orbits for the block (24,28) of the Campus 1 PETS video [25] are shown. Frames identified as moving using our local variation method are marked with gray dots while stationary frames are marked with black dots. The distribution of black dots is multimodal globally. We observe two main modes that represent the background blocks. They are identified as two 3D blobs that correspond to two different background textures that appeared in the course of this video at block position (24,28): a part of parking lot and a parked car. Around these blobs we see one dimensional orbits marked with gray dots corresponding to moving objects.

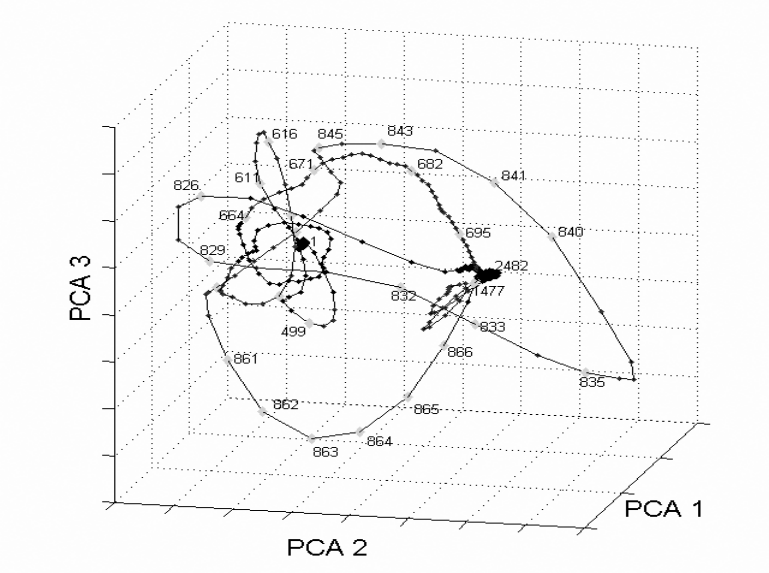

Figure 3.1: Campus 1 motion frames. Orbits of block (24,28) vectors marked with motion frames, using ‘reset’ and ‘hold’ mechanisms identified by the EM algorithm.

We can view the proposed local variance method as orbit classification algorithm. The reason is that elongated one dimensional orbits that identify motion have higher spread than the stationary background objects.

The dot labeling as shown in Figure 3.1 was computed by the proposed method for detection of moving objects. Observe that the gray dots perfectly correspond to the one dimensional motion orbits that identify moving blocks. Thus, the proposed algorithm correctly detected moving objects. In contrast, for the same Campus 1 video the incremental Expectation-Maximization type method failed to identify the motion orbit containing frames 633—663. In comparison to any pixel-based approaches (e.g., as originally proposed in [14]), motion detection based on 3D blocks performs better since it reduces noise in background and can extract information about temporal change of texture (since it is based on spatiotemporal texture representation of 3D blocks instead of pixels). To demonstrate how noisy RGB color values of a single pixel can be, we plot an orbit over time of RGB color values that occur at the pixel (185,217) which is one of the pixels in the block (24,28) of Campus 1 video (Figure 3.2). For better visualization, in Figure 3.2 we show the linearly transformed space of PCA projections of the original RGB color values (the trajectory in the space of original RGB colors is similar). To allow a proper comparison to the results in Figure 3.1 (computed by our local variance technique), the dot labels from Figure 3.1 are copied to Figure 3.2.

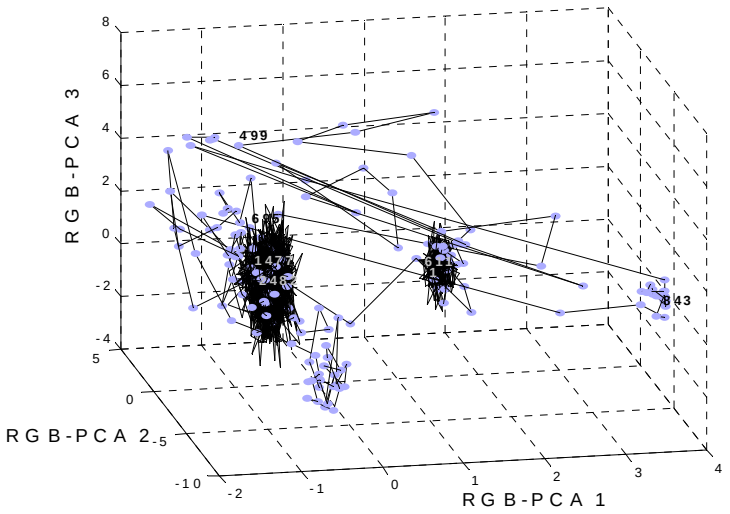

Figure 3.2: Campus 1 RGB pixel values. Standardized PCA components of RGB pixel values for Campus 1 at pixel location (185,217) that is inside block (24,28); allows a direct comparison to Figure 3.1.

By comparing Figure 3.1 to Figure 3.2 one can conclude that in both representations there are two distribution components corresponding to the background. However, using the block-based approach, the background variance is much smaller, since using block vectors that contain texture information results in effective noise reduction in comparison to using “raw” pixels. Hence, any technique to detect moving objects as outliers will perform much better using spatiotemporal blocks than when using the raw pixels. As it can be seen in Figure 3.2, the method from [14] has difficulties in properly detecting frames 611, 695, and 1477 belonging to the second and fourth moving objects that appear at the observed pixel. The gray dots incorrectly become parts of two background components, which imply that a pixel-based method [14] would classify the corresponding gray dots as belonging to a background distribution. The proposed local variation based technique can also be applied on pixel level. However, due to problems with large uniform texture regions as well as noise inherent to pixel values (shown above), the preferred technique is to apply local variance method on spatiotemporal texture block vectors.

Figure 3.3 shows a close up of motion orbits for Campus 1 video with distinct two backgrounds. Each background cluster is very close-fitting and motion is evident by the elongated vectors. A single background without any motion is shown in Figure 3.4. Same scale is used in Figure 3.4 as Figure 3.3 to compare background spread. The close up of Figure 3.4 shows the background as a compact representation of features and the noisy nature of Campus 1 video.

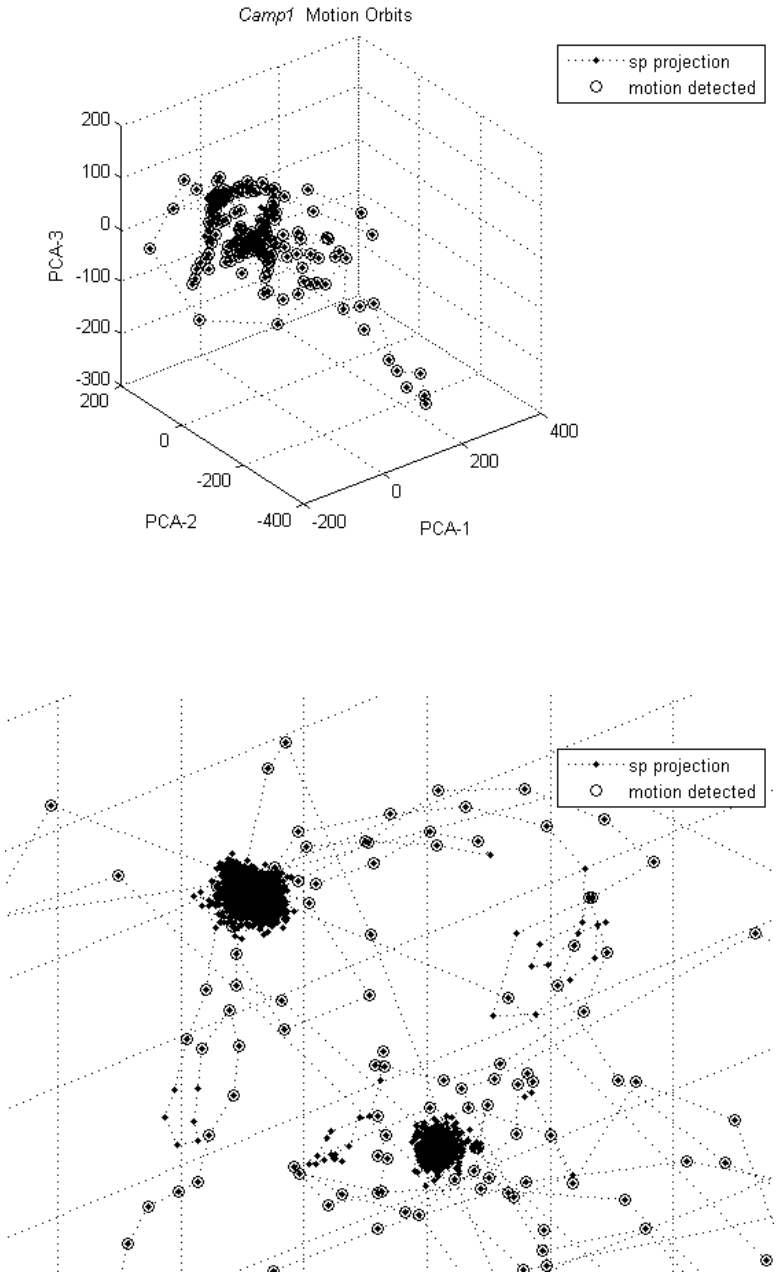

Figure 3.3: Campus 1 two backgrounds. Example of color motion orbits. Motion block 47x54 of Campus1 video showing two background clusters and detected motion orbits. Full scale and close-up of two background clusters is shown.

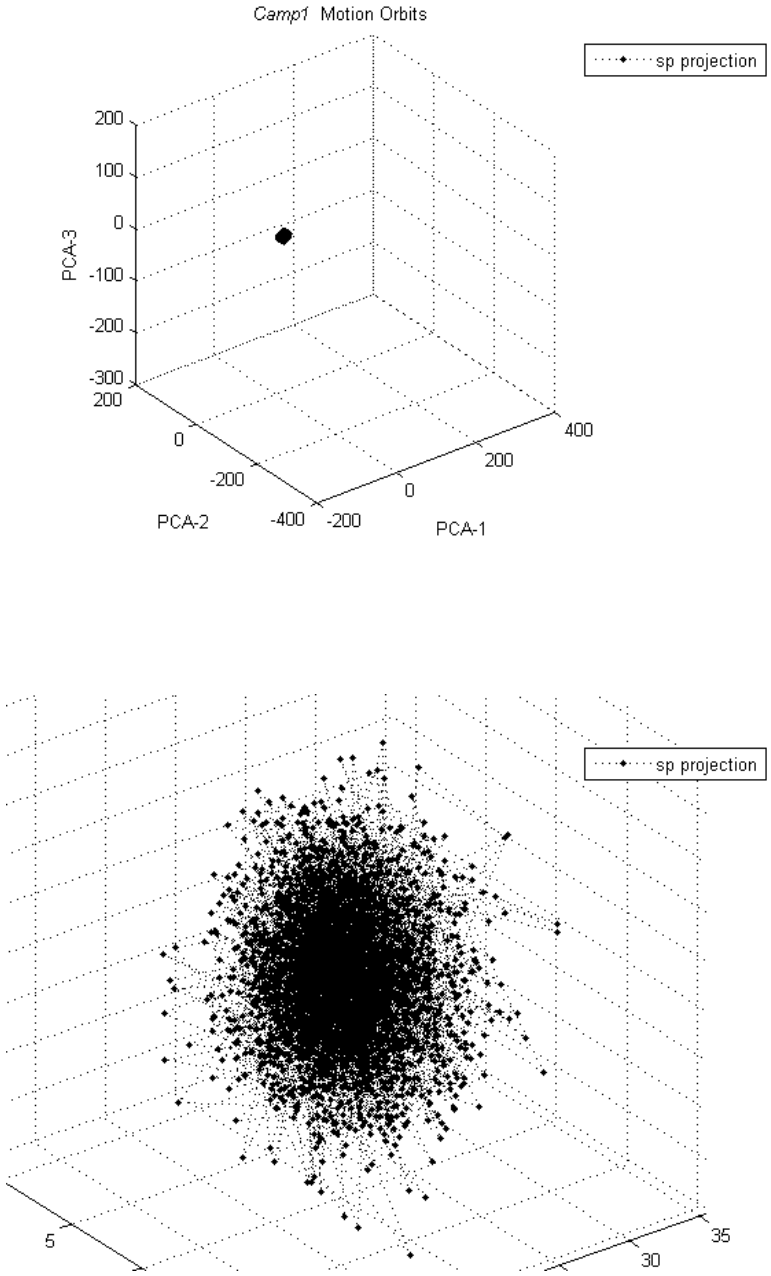

Figure 3.4: Campus 1 single background. An example of color static background. Full scale and close-up of single background cluster without any motion orbits.

3.2 Decreased Sensitivity to Noise

It is well-known fact that noise to signal ratio in thermal infrared videos is higher than in visible light videos. The infrared noise can be viewed as jitters in pixel values. Figure 3.5 illustrates the performance of the Gaussian mixture model [14] on infrared pixels values on Infra 2 video from [19]. A large number of false-positives, some have the size of actual moving objects. The usage of the proposed spatiotemporal texture vectors eliminates very effectively the infrared jitter noise as we can see in Figure 3.5.

A noisy color video sequence also poses a challenge to the Gaussian mixture model as is shown in Figure 3.6. The spatiotemporal motion detection method handles the frame shift well when the texture blocks contain only most significant spread of texture vectors. Frame 120 of the Split 1 video shows three moving objects as detected by spatiotemporal motion vectors, whereas many more objects are detected by the Gaussian mixture model.

Additionally, the visual inspection of frames is supplemented by ground truth data evaluation. Centroids of the motion regions as detected by spatiotemporal method and Gaussian mixture model method are projected onto single frame, and difference from ground truth data is computed (Section 3.3).

It should be noted, that large frame shifts also introduce false motion detection by the spatiotemporal motion detection method. If the frame motion is significant, and large sections of the frame change position, the spatiotemporal method will mark blocks as motion when there is only translation of the entire frame (Chapter 4).

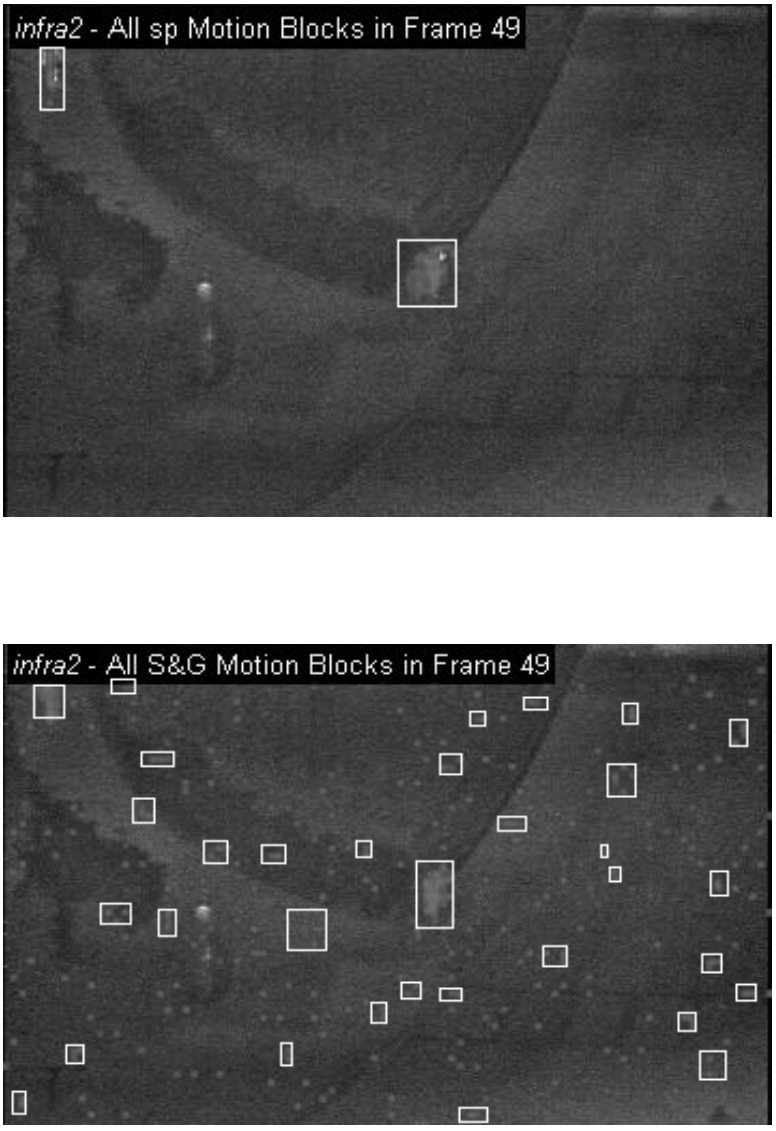

Figure 3.5: Infra 2 video frame 49 with detected motion: Example of sensitivity to noise in infrared video as proposed by the outlier detection based on spatiotemporal blocks and S&G Gaussian mixture model.

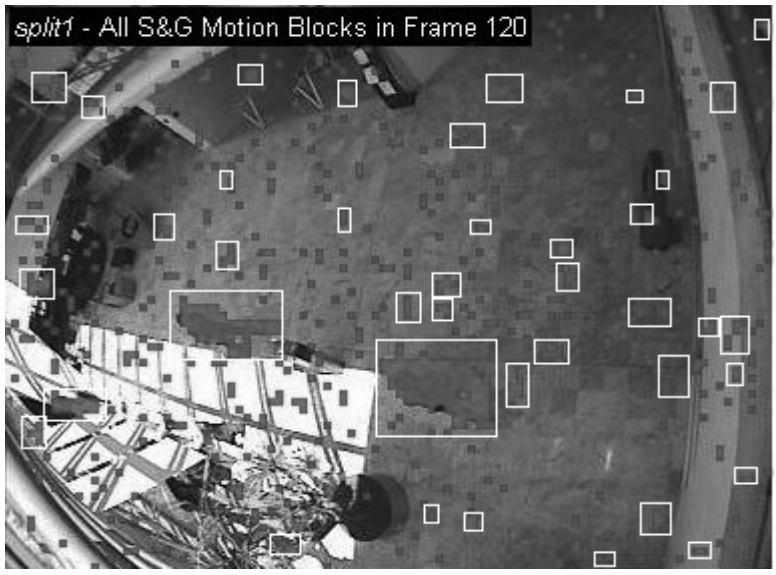

Figure 3.6: Split 1 video frame 120 with detected motion: Example of sensitivity to noise in color video as proposed by the outlier detection based on spatiotemporal blocks and Gaussian mixture model.

3.3 Ground Truth Data Evaluation

The video clips and corresponding ground truth data used in this evaluation were obtained from Ohio State University Thermal Pedestrian Database [19]. Video was captured using a Raytheon 300D thermal sensor core with 75 mm lens. Camera was mounted on an 8-story building overlooking a pedestrian intersection on the Ohio State University (OSU) campus. Ground truth data gives us number of objects and their centroids in each video frame. In order to compare the two methods to the ground truth data, we must detect motion, find objects from motion data, and compute their centroids. Process each video sequence to identify motion on block level and establish motion/no motion binary image as described in Chapter 2. The output from motion detection is fed into object labeling algorithm to measure the object's region of interest and centroid location. Connected components are used to establish motion regions of interests with a minimum of two blocks per region. We evaluate motion block components as 8-connected objects. It should be noted that the method of creating motion rectangles is only used to compare motion centroids to ground truth data. The proposed tracking algorithm in Chapter 6 uses a different method to create tracking regions. The comparison of tracking regions to ground truth data is presented in Chapter 6 as well.

Ground truth centroids for Infra 2 video are shown in Figure 3.7. All ground truth centroid are projected to visualize all motion paths simultaneously. Observe that the motion centroids coincide very well with the ground truth for the proposed spatiotemporal method. On average, our spatiotemporal motion centroid distance from

ground truth data was 4.62 pixels with standard deviation of 2.54 pixels for Infra 2 video. The infrared jitter noise on the pixel level makes the detected moving objects by Gaussian mixture model method (without post processing) to form a dense set in the video plane.

Ground truth data comparison for Walk 1 video is shown in Figure 3.8. It was obtained from the CAVIAR project [24], and it shows the spatiotemporal motion centroids and Gaussian model region centroids superimposed on ground truth data.

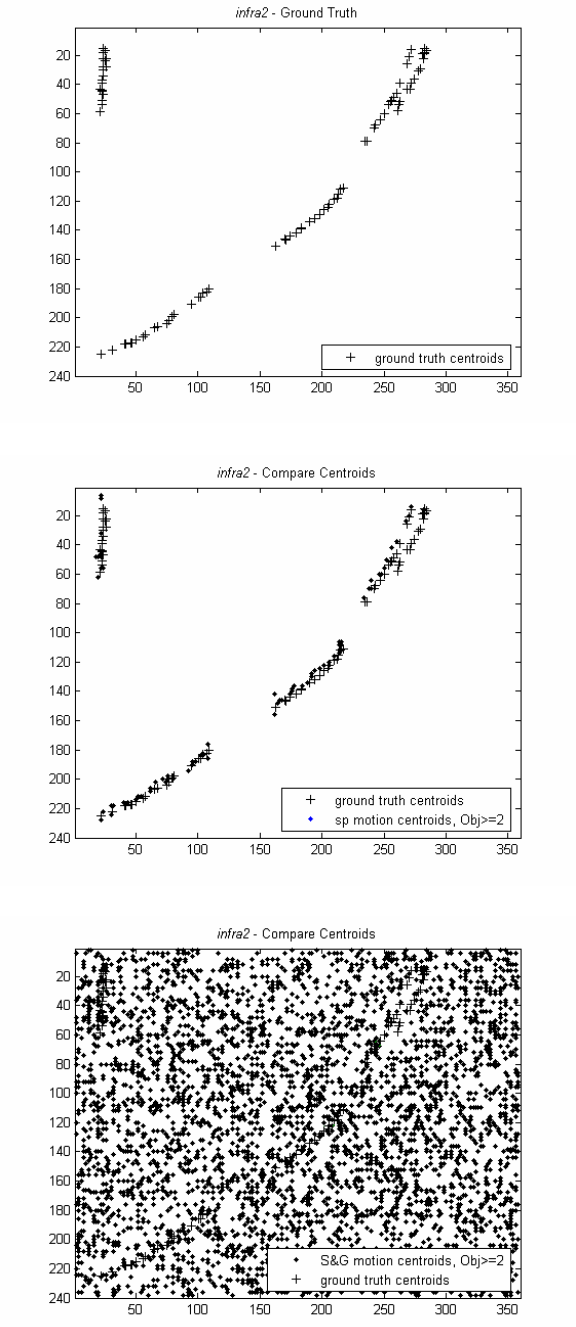

Figure 3.7: Infra 2 ground truth projection. Projection of all ground truth data for Infra2 video with objects size >= 2. Showing ground truth data, ground truth data and motion centroids, and ground truth data and Stauffer-Grimson Gaussian mixture model centroids.

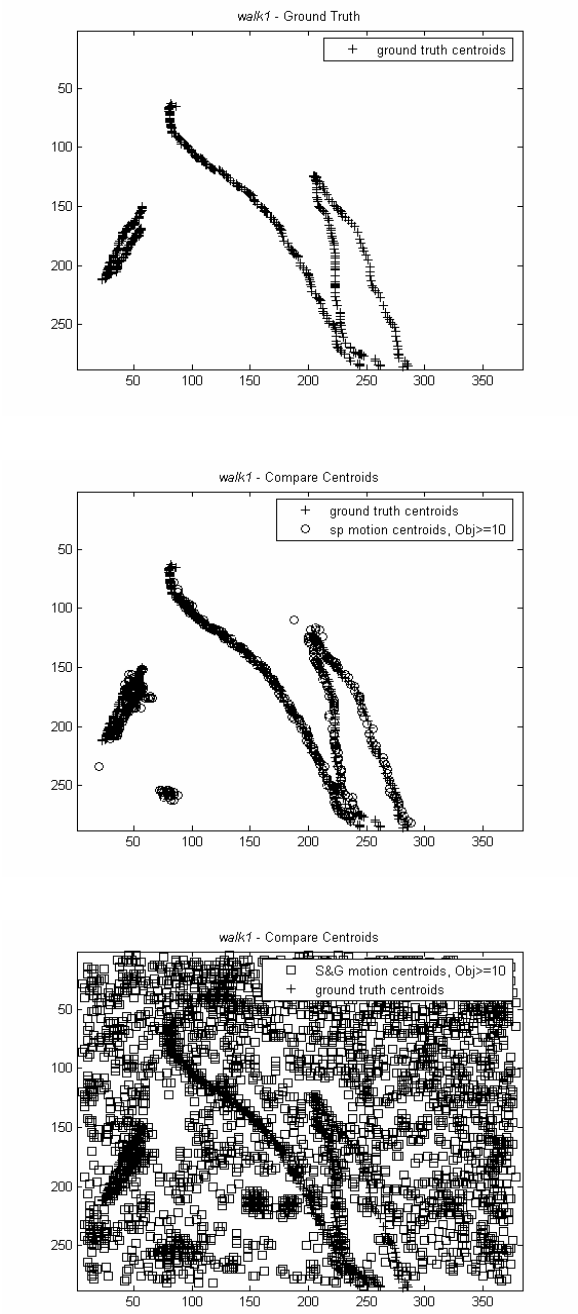

Figure 3.8: Walk 1 ground truth projection. Projection of all ground truth data for Walk 1 video with objects size >= 10. Showing ground truth data, ground truth data and motion centroids, and ground truth data and Stauffer-Grimson Gaussian mixture model centroids.

CHAPTER 4

EVALUATING RELIABILITY OF MOTION FEATURES

Two examples are shown of video content changes that cause the existing motion detection approaches to inaccurately detect the presence of substantial motion. Clearly, the detected motion is present in videos, but it is due to some content artifacts and is not due to the actual presence of moving objects. Consequently, human observant ignores such “motion” as irrelevant, while standard video analysis systems detect it as significant activity. It is shown that the proposed feature reliability methods identifies the unreliable motion features, and ignores the irrelevant artifacts. This is possible without reducing the detection rate of real moving objects. Consequently, the goal is to eliminate false alarms without reducing the detection rate. It is stressed that this is obtained without any direct video content analysis (e.g., using different features), but by monitoring the reliability of computed features. As stated before, direct video content analysis of features does not solve the problem, since these features may also become unreliable.

First example illustrates motion artifacts in Campus 3 video introduced by some reflections in windows that are a result of passing by cars. Figure 4.1 shows two frames from Campus 3 video, one showing real motion, and the second showing the motion artifacts in addition to the normal motion. Second example, Figure 4.2 shows motion

artifacts introduced by video compression and the same scene without such artifacts. This Temple 2 video was recorded in real-world environment by the video surveillance system of the Campus Police Division of the Temple University.

Figure 4.3 shows a video artifact produced by the wireless transmission of the video signal. Two frames of the Entry 1 video show a horizontal-scan band of noise, once when there is no real motion and again when there is actual motion in the field of view. When there is no real motion, the spatiotemporal algorithm detects motion where the noise band is located. This produces false-positive motion detection and is very difficult to differentiate from actual motion of a person. One of the solutions to this method involves the use of a tracking algorithm to distinguish between periodic motion and actual object motion.

Section 4.1 describes a temporal method to determine the reliability of motion detection. The temporal method monitors features computed by the proposed motion detection approach presented in Chapter 2. The motion features are computed for graylevel or infrared videos using 3D spatiotemporal blocks of spatial size 8×8 pixels, and temporal size of 3 frames. The blocks are disjoint in space and overlap by one frame in time. As a result we obtain motion activity values for each 8×8 block in each video frame. By thresholding the motion activity values, a binary feature produced, called motion detection, with 1 for ‘motion detected’ and 0 for ‘no motion detected’. Other statistical methods that handle interference suppression are proposed by [17].

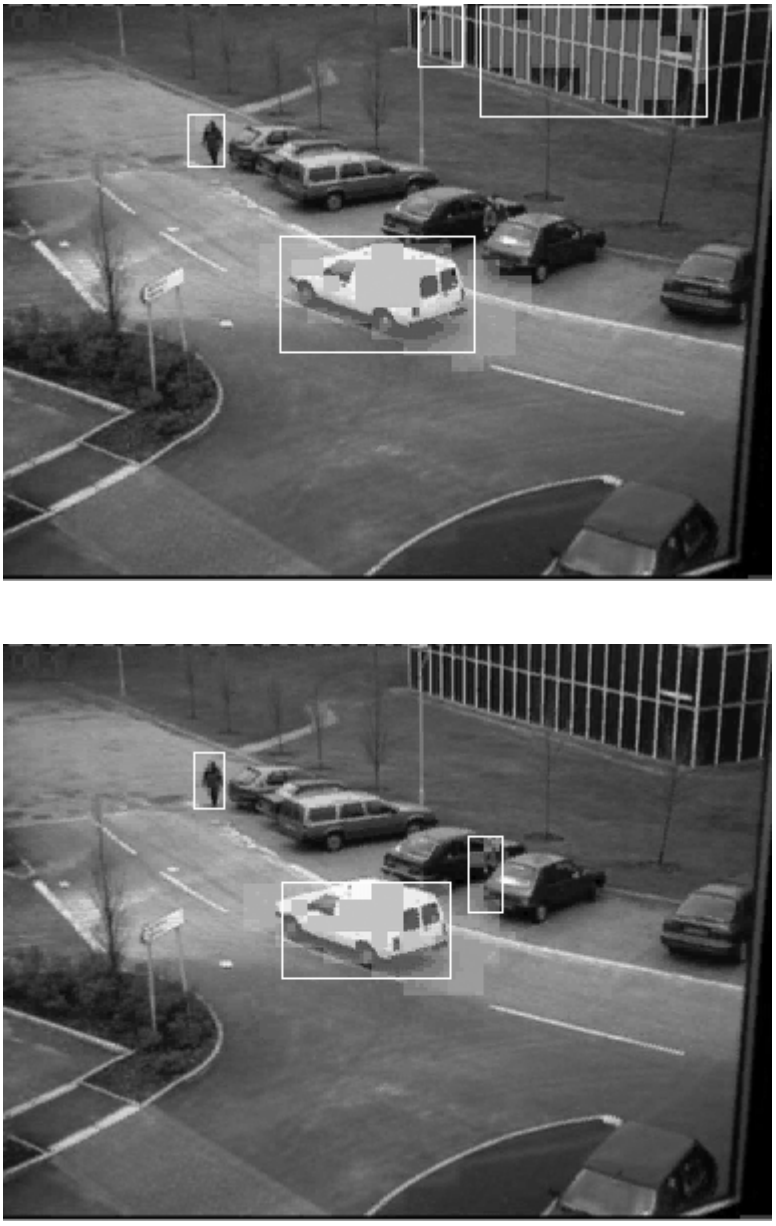

Figure 4.1: Campus 3 reflected noise. Two frames from Campus 3 video with moving blocks highlighted red: Shown are motion artifacts due to reflections in the windows, and the same scene (a few frames later) without the artifacts.

Figure 4.2: Temple 2 video compression noise. Two frames from Temple 2 video with moving blocks highlighted showing: motion artifacts introduced by video compression, and the same scene (a few frames later) without the artifacts.

Figure 4.3: Entry 1 video transmission noise. Two frames from Entry 1 video with moving blocks highlighted by bounding rectangle showing: motion artifacts introduced by video transmission, and detected motion with more artifacts due transmission.

4.1 Temporal Analysis of Feature Reliability

This section describes a temporal method to determine the reliability of motion features. The input motion feature has binary values for each 8×8 block of each video frame with 1 for 'motion detected' and 0 for 'no motion detected'. The algorithm described in Chapter 2 computes this feature vector. The 8×8 feature blocks are disjoint. Let $f(n)$ be the number of 1s in the frame number n, i.e., $f(n)$ is the number of moving blocks as function of frame number. We use the finite difference approximation of first derivative of f, $\partial f/\partial t$, to monitor the reliability of our motion detection. In other words, if the jump in values of $\partial f/\partial t$ is above a certain threshold for a given time interval, the binary feature is unreliable in this interval. The threshold necessary to detect the unreliable features is not static. The proposed dynamic thresholding algorithm described in Chapter 2 learns and varies this threshold. However, some other learning techniques could also be used.

This reliability property works under the assumption that there exists an upper bound on the size of moving objects whose motion we want to detect (measured in the number of moving blocks). This assumption holds for most surveillance videos. Now let us consider an example video, Temple 2, that satisfies this assumption. This video is recorded by a roof mounted, stationary camera, so that a certain minimal distance to moving objects is guaranteed. Typical moving objects there, humans and vehicles, cannot get arbitrarily large. Hence, the fraction of the scene occupied by a moving object is limited. Observe that the actual value of the upper bound on the size of moving objects needs not to be known, since the algorithm learns it automatically.

Figure 4.4 shows the motion amount of Temple 1 video with significant spikes where the compression errors occur. When looking at motion orbits of a single block from this video that does not have actual motion, it is evident that due to compression errors, the video exhibits two distinct backgrounds as shown in Figure 4.5. Due to the shift in frame position, a block without any motion has two distinct background clusters.

Figure 4.6 shows the graph of function f and first derivative of f for Temple 2 video were significant motion artifact are present. Time intervals with significant jumps of f that are correctly identified by the dynamic thresholding are marked with dotted lines in Figure 4.7. The graph of modified feature f, when f was set to previously known value within the time intervals when motion was detected as unreliable is shown in Figure 4.7. The proposed method is able to identify and exclude the unreliable results of motion detection. By excluding these time intervals from further processing, we not only eliminate false alarms, but make it possible to correctly detect alarm situations, although the input motion detection in not 100% reliable. For example, a significant increase in the number of motion blocks after the frame 1700 indicates an alarm situation, Figure 4.7. This is a correct prediction; since a street fight is recorded in the video after this frame.

As noted before, an interesting observation may be made when evaluating motion amount of the entire frame to the motion orbits of a single block without any motion. The relation of motion orbit vectors of the two backgrounds (Figure 4.5) it may be directly translated to the frame numbers of the motion amount graph in Figure 4.4.

Unfortunately, the thresholding constants C_1, C_2, and u of the dynamic distribution learning algorithm need adjustment to detect motion between the two orbits in Figure 4.5.

Activity detection is further discussed in Chapter 5 showing the exact method to detect increased activity around frame 1700 of the Temple 2 video, once the noise is identified and removed. The first derivative of the filtered motion amount is then used to detect increased activities using the same principles of dynamic distribution learning as presented for motion detection.

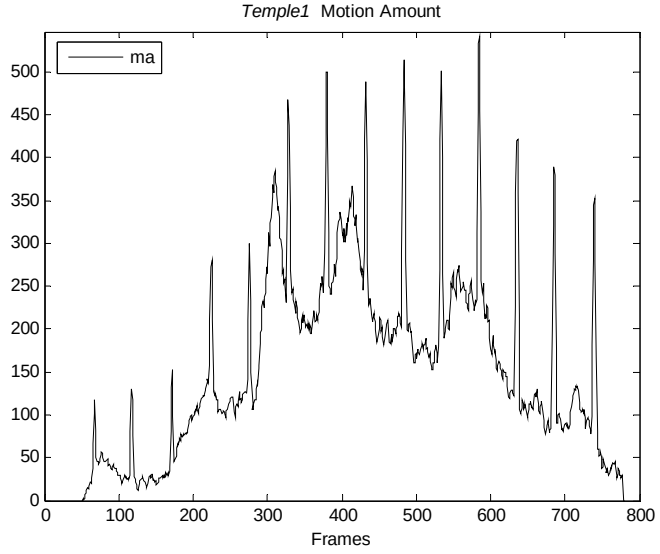

Figure 4.4: Temple 1 motion amount. The graph of $f(n)$, which is the number of moving blocks as function of frame number n showing periodic jumps.

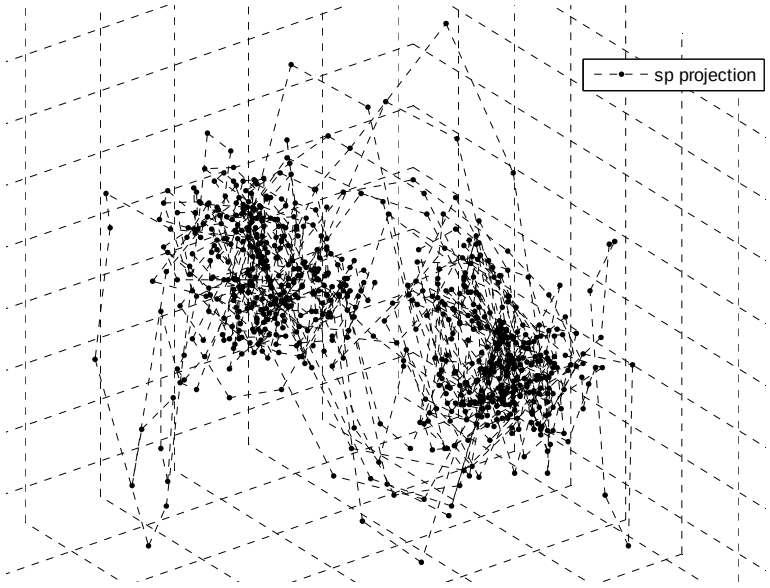

Figure 4.5: Temple 1 noisy background without motion. Projection of block 9x53 showing two background clusters corresponding to the periodic jumps in Figure 4.4.

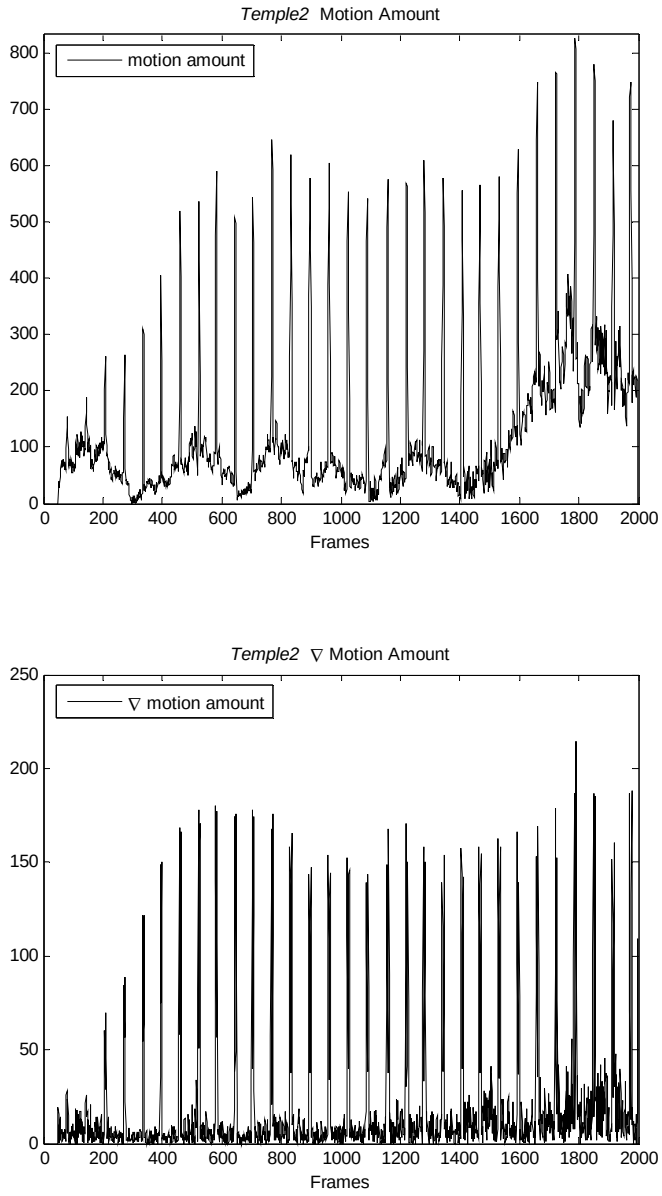

Figure 4.6: Temple 2 motion amount. The graph of $f(n)$, which is the number of moving blocks as function of frame number n, and gradient of $f(n)$ indicating the frames with large motion jumps.

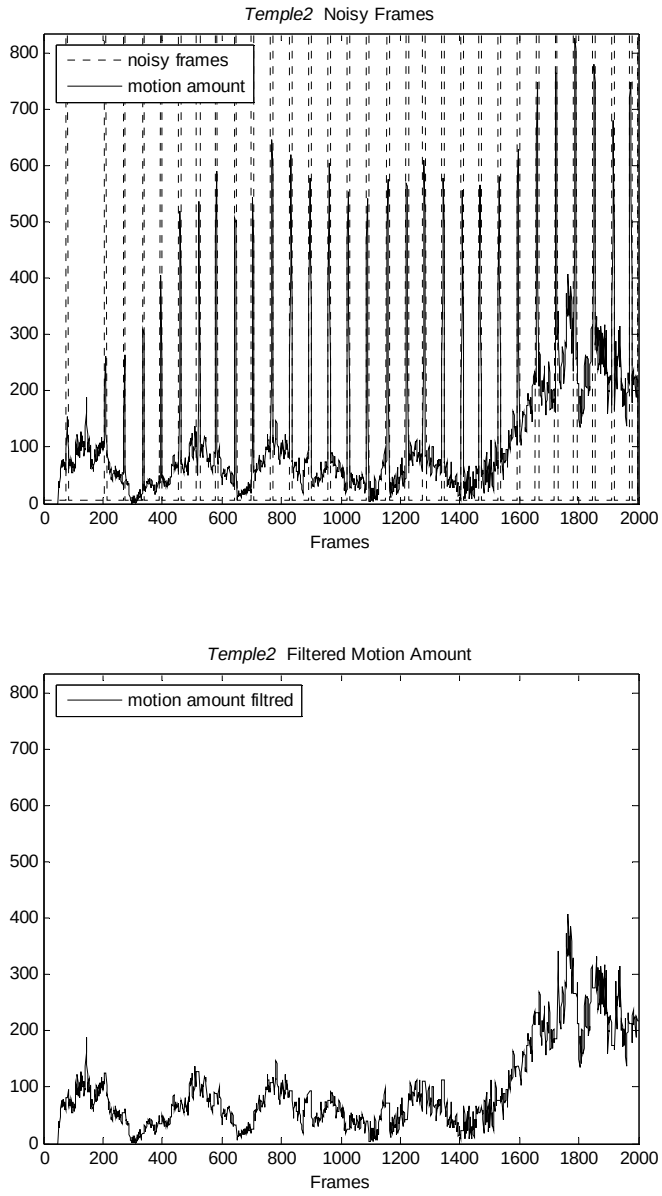

Figure 4.7: Temple 2 filtered motion amount. The graph of $f(n)$ with significant jumps of f (caused by feature unreliability) correctly identified by our dynamic thresholding, and the graph of $f(n)$ with unreliable motion frames removed.

CHAPTER 5

DETECTION OF INCREASED ACTIVITIES

The robust computation of the motion measure mm, allows to reliably estimate the motion amount in each video frame [23]. Motion amount can be defined as the sum of detected motion as a function of motion measures of all blocks:

$$ma(t) = \sum_{x,y} dm(x, y, t)$$

where

$$dm(x, y, t) = \begin{cases} 1 & mm(x, y, t) \rightarrow motion \\ 0 & mm(x, y, t) \rightarrow static \end{cases}$$

The proposed method of detecting increased activities is again based on outlier detection (Chapter 2) but this time of the motion amount over time. System needs to learn the distribution of motion amount over time when the recorded video activity was considered usual or nominal. Then time intervals with increased activity are detected as outliers of the learned distribution. The proposed approach works under the assumption that there exists an upper bound on the size of moving objects whose motion we want to detect (measured in the number of moving blocks), and that the genuine moving objects do not appear rapidly in the frame. These assumptions hold for most surveillance

videos. The first video example, called Temple 1, satisfies the assumptions. Typical moving objects, people and cars, cannot get arbitrarily large as the distance to the roof mounted camera is fixed, and the motion regions are limited in size. As stated before, the upper bound on the size of moving objects the algorithm learns automatically. Similarly, the number of humans and vehicles cannot rapidly increase, since the regions of entry into the camera view field are limited in size.

The graph of function ma for Temple 1 video and correctly detected alarm situations are shown in Figure 5.1. For example, a significant increase in the number of motion blocks around frame 310 indicates an alarm situation. This is a correct prediction, since a street fight is recorded on the video around frame 310, see Figure 5.2. Also, Figure 5.2 shows a frame with nominal detected motion. Only when nominal motion is exceeded around frame 310, the system marks these frames as 'ACTIVITY'.

Temple 2 video exhibits periodic noise as described in Chapter 4. Once noisy frames are identified and removed, a new filtered motion amount is used to detect increased activity. The dynamic thresholding and outlier detection is used on the filtered motion amount values to detect increased activities. The thresholding constants are set to $C_1 = 5$, $C_2 = 2$, and $u = 0.95$. These values are different from the motion detection constants used on the motion measure.

Figure 5.3 shows the filtered motion amount and detected increased activity as dotted boundaries around frame 1700 of Temple 2 video sequence. The corresponding video frame shows a street disturbance around this timeframe in Figure 5.4. Individual motion objects are marked with rectangles to highlight their position in the video frame.

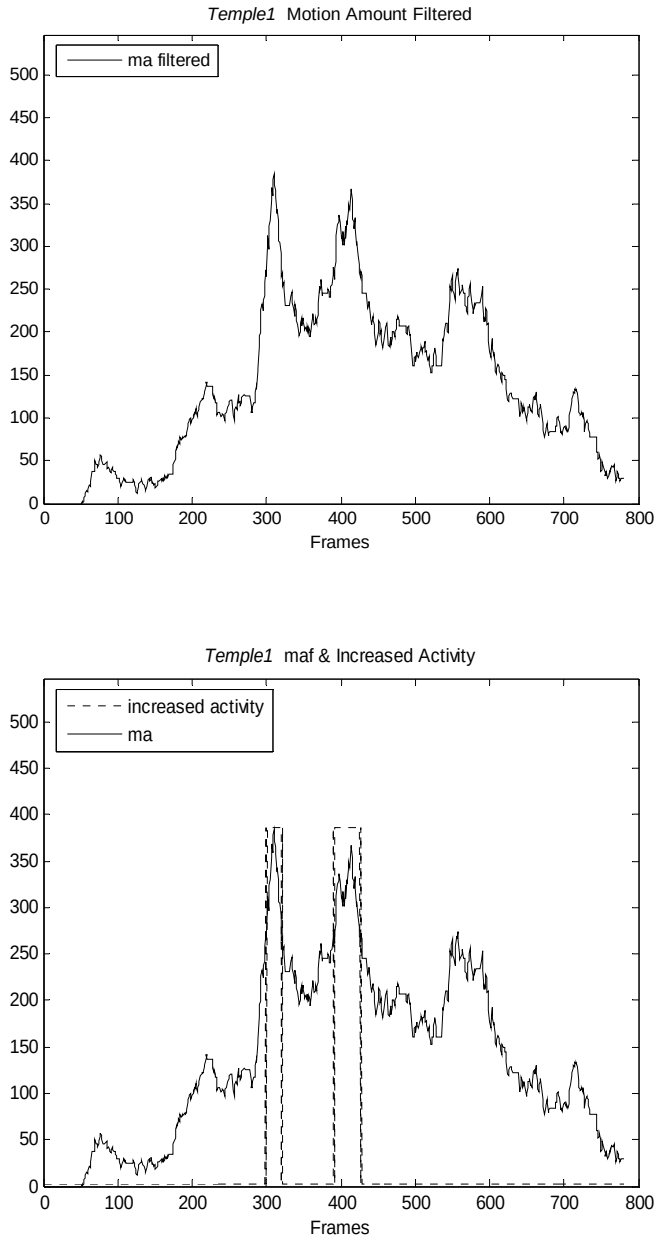

Figure 5.1: Temple 1 increased activity. Motion amount of Temple 1 video; and increased activity frames marked with dotted boundaries.

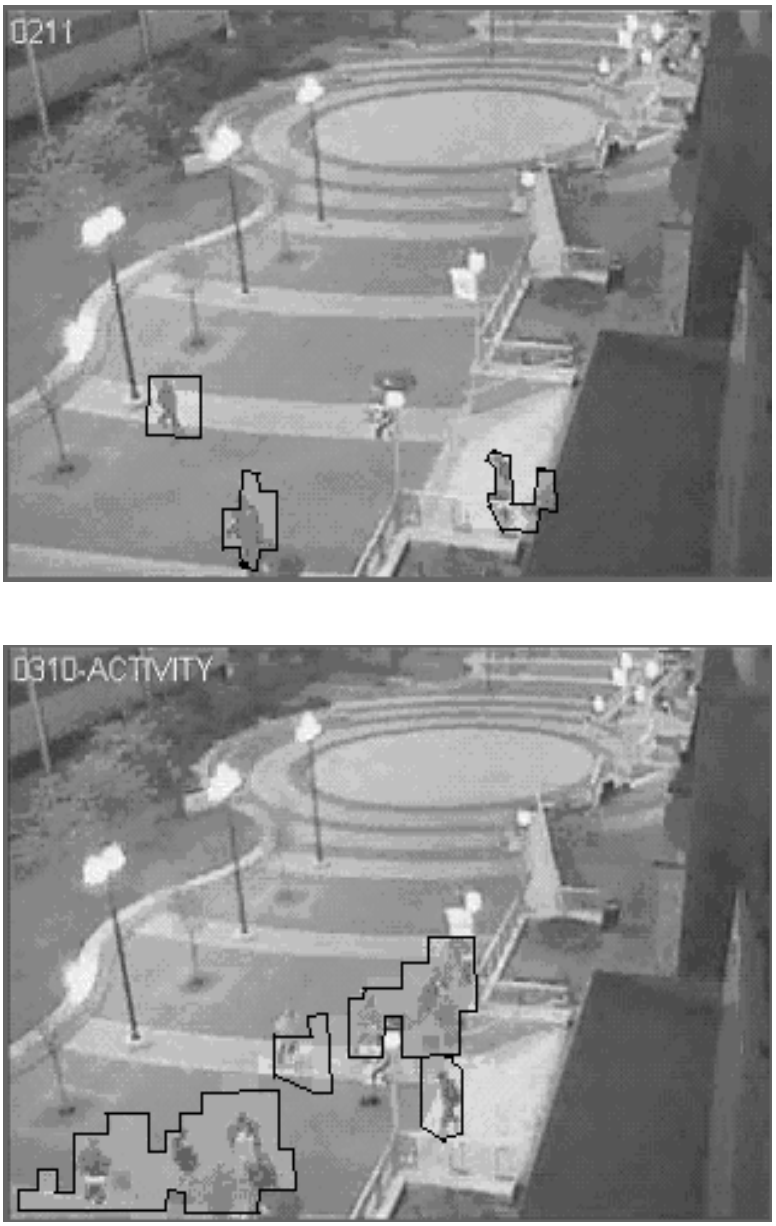

Figure 5.2: Temple 1 detected activity video frames. Frames showing no activity and showing increased activity due to street fight (ACTIVITY label is shown next to the frame number).

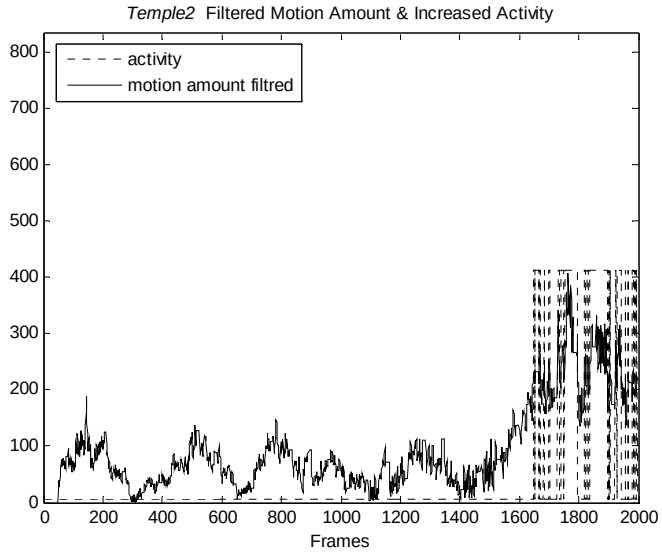

Figure 5.3: Temple 2 increased activity detection. Increased activity frames marked with dotted boundaries.

Figure 5.4: Temple 2 increased activity video frame. Increased activity video frame 1771 showing a street disturbance.

CHAPTER 6

OBJECT TRACKING

This chapter presents several tracking methods that are the basis for the proposed selective hypothesis tracking algorithm. Only the swarm template matching from Section 6.1 was not fully utilized in the proposed tracking method; however it is presented here as it was the first tracking method that employed the spatiotemporal motion regions. In Section 6.2 an initial minim cost tracking algorithm is presented, and in Section 6.3 the classic Lucas-Kanade image registration is evaluated. The proposed selective hypothesis tracking algorithm is presented in Section 6.4 utilizing modified image registration and minimum cost computation based on motion detection method presented in Chapter 2. Evaluation of several test videos is performed showing different tracking scenarios in color and infrared images.

6. 1 Swarm Template Matching

The particle swarm optimization (PSO) method, is an optimization method based on artificial life with its roots in bird flocking and swarming theory. Each potential solution is assigned a randomized velocity vector, and the potential solutions

called particles then converge through the search space seeking the function optima [26].

Random vector introduces a source of noise, search then becomes more directed after a few iterations as the swarm starts to concentrate around the best region. Particles change their direction based on the combination of their own experience and the best experience of the group. Better solutions attract the particles to the region with the optimum solution. Overall the total number of function evaluations is very small compared to the size of the solution space, such as pixel to pixel match.

Particle swarm optimization general formula defines $x^i(t)$ and $v^i(t)$ as the position and velocity vectors at time t of the i^{th} particle. The velocity vector and new position are defined as

$$v^i(t) = V_{old}^i + P_{best}^i + G_{best}^i$$

$$x^i(t) = x^i(t-1) + v^i(t)$$

where

$$V_{old}^i = wv^i(t-1)$$

$$P_{best}^i = c_1 * rand() * \left(p_{best}^i - x^i(t-1) \right)$$

$$G_{best}^i = c_2 * rand() * \left(g_{best} - x^i(t-1) \right)$$

and

c_1 and c_2 are weights that influence the velocity vector,

w is a convergence decay constant,

$rand()$ function generates a random number between 0 and 1 with a uniform distribution,

p_{best}^{i} is the best solution observed so far of the i^{th} particle,

g_{best} is the global best parameter among all particles.

In experiments we have used: $c_1 = 0.01$, $c_2 = 0.5$, and $w = 0.01$. The convergence condition is based on the global maximum of the target function. It is reached if a certain percentage of particles are in a given neighborhood $U(g_{best})$ of the best particle. We used 80% and 99% of particles, and $U(g_{best})$ was a 5×5 pixel neighborhood.

A modification to PSO formula is based on *k*-Groups of swarms. Each group evaluates a different function template, that is, each group searches for the maximum of a different function. The relations between maxima are approximately known and given by

$$offset(i, j) = -offset(j, i)$$

The *k*-Groups swarm track algorithm has the following steps:

- Find global best solution g_{best},
- Evaluate for 50% convergence in $U\left(g_{best}^{j}\right)$, for each group $j = 1 \ldots k$,
- For each group update velocity and position vectors,
- If less than 50% of particles in *k*-Group are within best solution use g_{best} and $offset_j$. If more than 50%, of particles in *k*-Group are within best solution use g_{best}^{j} in the PSO formula.

The update formula for position of particles in group with g_{best} is

$$g\left(x_i^{(k)}\right) = g(gbest_i^{(k)})$$

$$\Delta x_i^{(k+1)} = \alpha \Delta x_i^{(k)} + c_1 rand()\left(pbest_i^{(k)} - x_i^{(k)}\right) + c_2 rand()\left(gbest^{(k)} - x_i^{(k)}\right)$$

The update formula for position of particles in groups without g_{best} is

$$g\left(x_i^{(k)}\right) \neq g(gbest_i^{(k)})$$

$$\Delta x_i^{(k+1)} = \alpha \Delta x_i^{(k)} + c_1 rand()\left(pbest_i^{(k)} - x_i^{(k)}\right) + c_2 rand()\left(gbest^{(k)} + offest - x_i^{(k)}\right)$$

For particles in a group without the g_{best}, the position update is based on the g_{best} and the known $offset$. This allows groups without g_{best} to take advantage of the other group's best value and its own offset. The convergence based on this concept,

allows dividing the search space where groups initially work alone and then combine their best experiences.

6. 2 Minimum Cost Estimate

Robust detection of motion regions in videos introduced in Chapter 2 is the basis for this initial tracking algorithm. A modified and simplified version of the minimum cost tracking algorithm introduced by [22] is proposed. Each new detected motion region i in frame t has a known bounding box B_i, centroid location X_i and zero initial velocity V_i. Known motion region L in a frame $t-1$ has centroid X_L and velocity V_L and a predicted centroid X_{LP} in frame t. Minimum cost C_{Li} between X_L and X_i is computed based on the predicted location of known track labeled regions and new detected motion regions.

Predicted centroid location of known motion region in frame t is

$$X_{LP}^{t} = X_{L}^{t-1} + V_{L}^{t-1}$$

The cost between know motion region L and new region i is defined as

$$C_{Li} = \left\| X_{LP}^{t} - X_{i}^{t} \right\|$$

The association of motion region to predicted X_L regions is:

$$M_i = \sum_j^L m_{ji}$$

where

$$m_{Li} = \begin{cases} 1 & X_{LP}^t \in B_i^t \\ 0 & otherwise \end{cases}$$

If $M_i = 0$ then there is no known region association with any labeled region L. If the best C_{Li} is less than the minimum cost threshold T_C, then L is selected as the best match. Otherwise new tracking motion region is created with initial velocity set to 0. If $M_i = 1$ then there is exactly one tracking L region association (Figure 6.1). If however $M_i > 1$, then there is more than one tracking centroid within B_i (merge or crossover of motion regions). In this case X_i is updated using only the predicted location and the velocity remains constant.

Each labeled object L has a maximum time to live tracking factor associated with it, T_L. For each selected associated pair (L,i), the T_L is set to the maximum allowed time to track value $T_{L\max}$. All labeled objects L not associated with any current detected motion regions i have its T_L decremented by 1. Once T_L reaches 0 the labeled object L is no longer used in computing the minimum cost association between pairs (L,i).

The minimum cost computation as proposed by [22] is also based on the size of the bounding box and predicted size computation. In this preliminary method the size component of the minimum cost computation is negligible and therefore not used. In the proposed selective hypothesis tracking method (Section 6.4), the minimum cost computation does include size as a cost measurement.

An example of minimum cost computation tracking is shown in Figure 6.1. Object labeled ‘5’ is walking along the fence. It then turns the corner and disappears from the filed of view. This causes the motion not to be detected and consequently tracking to be suspended. Once object ‘5’ reappears on the other side of the fence, the minimum cost computation has no assigned motion rectangle with any tracking object. However, the closest object to this new motion region is ‘5’ and it is selected without creating a new object L. This is only possible as the distance between a motion region and last know labeled object to be less than T_C. If however, this distance was greater than T_C, then object ‘5’ that reappeared would not be labeled as ‘5’ but rather a new object would be created in its place.

The selective hypothesis tracking method does utilize the minimum cost estimation, but it also uses image registration to confirm an object’s general characteristics. Image registration confirms if predicted velocity vector matches motion region objects. Additionally, size is also a factor in the proposed tracking method along with a direction cost. The direction cost is based on the direction of the labeled object and, and the direction from the object to the motion region.

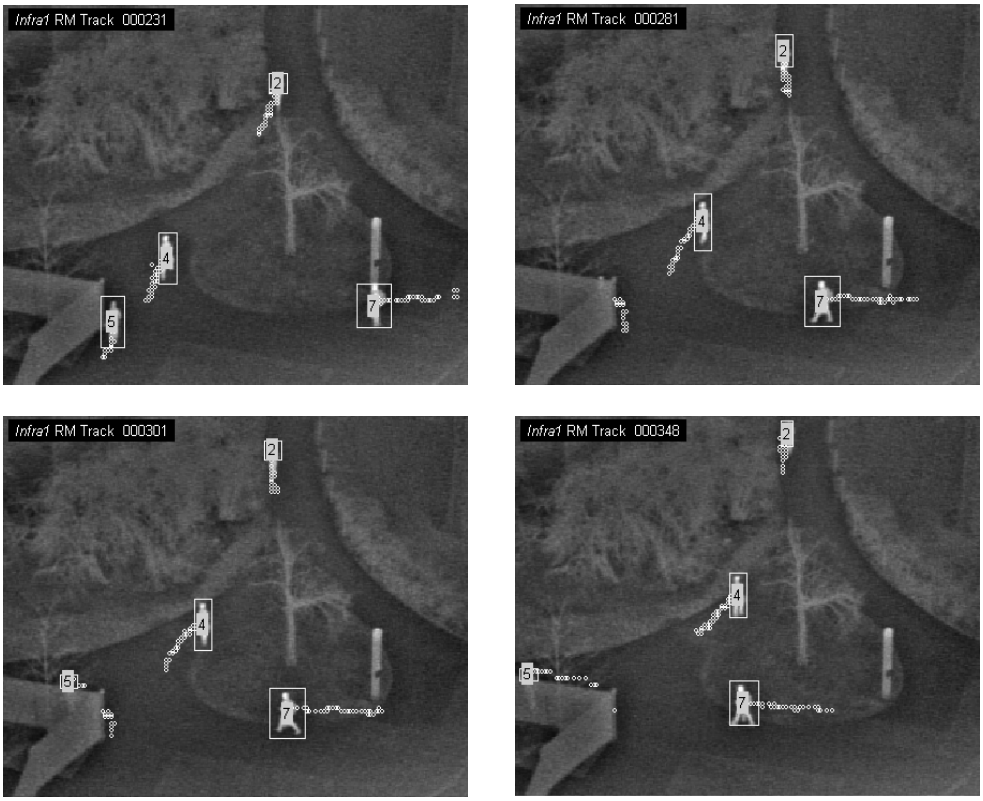

Figure 6.1: Infra 1 object tracking. Video frame sequence 231-348 showing object ID, bounding box and tracking trail. Object 5 walking along the fence, turning corner and is hidden behind the fence; and finally head of object 5 reapers behind the fence, tracking continues.

6.3 Image Alignment

One of the most widely used image alignment techniques is the Lucas-Kanade algorithm. It has become not only the standard for image alignment but also a standard for optical flow measurement. The basis to image alignment is the gradient descent computation, which is the de facto standard method.

The original image alignment algorithm was described by Lucas and Kanade in their 1981 paper [27]. The aim of Lucas-Kanade algorithm is to align a template T from image at time $t(i)$, to an input image G at time $t(i+1)$. Both template T and image G contain vector of the pixel values at coordinates (x, y). Template T is the extracted sub-region from image at time $t(i)$.

The goal of the Lucas-Kanade algorithm is to minimize the sum of squared error between the template T and image G. Image G is warped back onto the coordinate frame where template T resides. This warping requires interpolating the image G at sub-pixel coordinates relative to the coordinates of the template's frame.

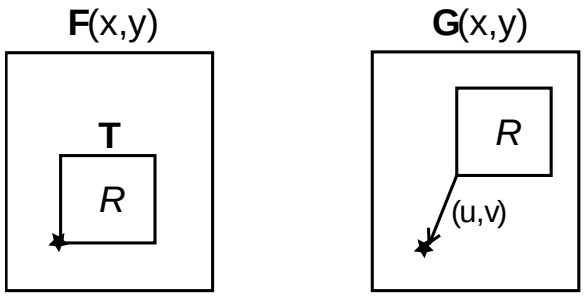

Figure 6.2: Lucas-Kanade registration. Register region R in frame G and find the offset vector (u, v)

The following error estimation is used to terminate the iterative process

$$E(u,v) = \sum_{x,y \in R} (F(x+u, y+v) - G(x))^2$$

it may be approximated by

$$E(u,v) \approx \sum_{x,y \in R} (F_x(x,y)u + F_y(x,y)v - F_t(x,y))^2$$

where spatial gradients are F_x and F_y and the temporal gradient is F_t. To minimize $E(u,v)$, both gradients are initially set to 0

$$\frac{\partial E}{\partial u} = 0$$

$$\frac{\partial E}{\partial v} = 0$$

and vectors u and v are computed based on the spatial and temporal gradients

$$\begin{bmatrix} \sum F_x^2 & \sum F_x F_y \\ \sum F_x F_y & \sum F_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum F_t F_x \\ \sum F_t F_y \end{bmatrix}$$

$$\mathrm{C}\begin{bmatrix} u \\ v \end{bmatrix} = \mathrm{D}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \mathrm{C}^{-1}\mathrm{D}$$

Parameters u and v are then updated

$$u \leftarrow u + \Delta u \ , \ v \leftarrow v + \Delta v \, ,$$

and $E(u,v)$ is checked against the previous values, and the error threshold

$$\left| E_{old} - E \right| < \varepsilon$$

Optimal (u,v) satisfies the Lucas-Kanade equation, however it should be noted that solving for (u,v) implies that $\mathrm{C}^{-1}\mathrm{D}$ should be well-conditioned and should be invertible. Additionally, the eigenvalue ratio λ_1 / λ_2 should not be too large.

The algorithm iterates until measured error is small enough and convergence is achieved. Value of $\|F_t\|^2$ is used to test for convergence. An example of error value convergence is shown in Figure 6.3.

To compute gradients F_x, F_y, and F_t, the image is blurred and convoluted with spatial derivative filter. The F_t is computed using weighted averaging kernel where emphasis is on the center. Both F_x and F_y are computed using standard deviation where emphasis is on the spread.

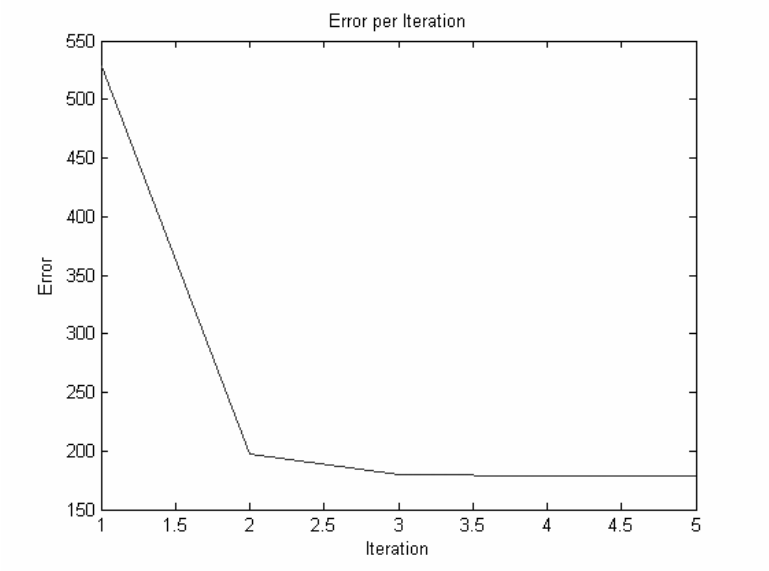

Figure 6.3: Error value per iteration. Convergence to the minimum error occurs around iteration number 4.

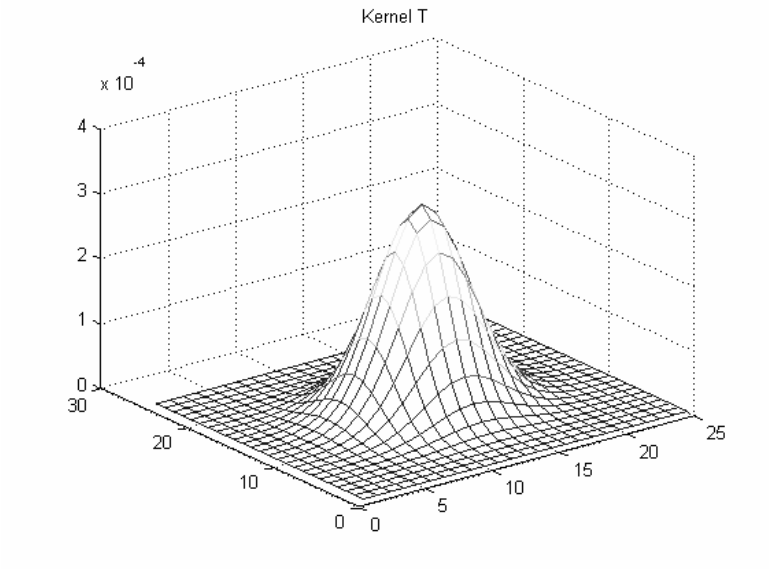

Figure 6.4: Image alignment temporal kernel. Temporal kernel used in Lucas-Kanade image registration.

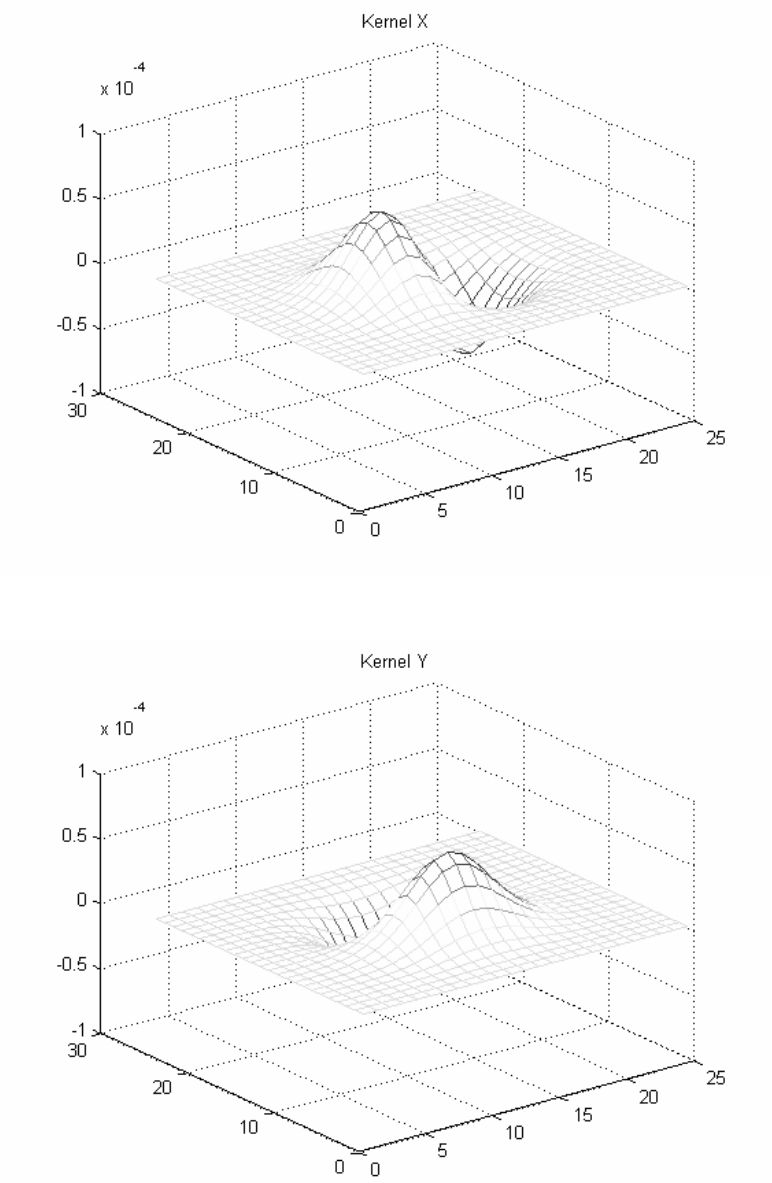

Figure 6.5: Image alignment spatial kernels. Spatial kernels used in Lucas-Kanade image registration showing Kernel X and Kernel Y.

The selection of Gaussian kernel derivatives to compute the gradients is based on mathematical convenience and efficiency. They also provide localization of the derivative and low signal to noise ratio.

Sub-pixel image interpolation is needed for the warping part of the registration algorithm, as image pixels will not be located on an integral grid. Simple bilinear interpolation is used to warp frame G, since the assumption is that the image is locally bilinear.

The warping function is defined as

$$G(x, y) = ax + by + cxy + d = 0$$

and it is interpolating within given region

$$\begin{bmatrix} G(x, y) & G(x+1, y) \\ G(x, y+1) & G(x+1, y+1) \end{bmatrix}$$

when u is within distance x and x+1

$$u \in \langle x, \ldots, x+1 \rangle$$

and v is within distance y and y+1

$$v \in \langle y, \ldots, y+1 \rangle .$$

The warping is computed using the above equation for $G(x, y)$ to obtain

$$G_{warp} = h\left(G, \begin{bmatrix} u \\ v \end{bmatrix}\right)$$

function h warps frame G by vectors (u, v) using bilinear interpolation.

An extension to the above two-dimensional registration is accomplished by applying the affine motion model, which is defined as

$$\begin{bmatrix} x+u \\ y+v \end{bmatrix} = \begin{bmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \alpha_5 \\ \alpha_6 \end{bmatrix}$$

this model provides for 2D translation, 2D rotation, scale in X and Y, and shear. The error estimation function for the affine motion model is defined as

$$E(\alpha_1, , \alpha_6) = \sum_{x,y \in R} (F(\alpha_1 x + \alpha_2 y + \alpha_3, \alpha_4 x + \alpha_5 y + \alpha_6) - G(x, y))^2$$

where $\langle \alpha_1, \ldots, \alpha_6 \rangle$ are solved using

$$\begin{bmatrix} \alpha_1 \\ \ldots \\ \alpha_6 \end{bmatrix} = \mathrm{C}^{-1}\mathrm{D}$$

6.4 Selective Hypothesis Tracking

The proposed tracking method utilizes spatiotemporal motion regions, modified image registration technique, and improved minimum cost estimation based on distance, motion vectors, direction, and persistence. Selection is made as when to use image registration offsets or estimated velocity vectors. Also, the minimum cost analysis selects the appropriate offsets. There is only one hard limit used in minimum cost estimation, the distance between motion region and any know template. This hard limit is based on the video's frames per second and a fixed threshold, providing a standard time and distance based limit.

Known templates are compared to the motion regions in order to classify the motion regions into either new templates or known templates. Each active template is registered in the current frame using the image alignment technique described in Section 6.3. Each template is then associated with a motion region based on image alignment (if template is not merged) or predicted motion (if template is merged). This template association selection is the core of the selective hypothesis tracking method. Finally, minimum cost estimation is computed for each motion region that does not have two or more templates associated with it. If there is exactly one association of template to motion region, it is treated as if motion region had no associated template. This decision allows picking the best template that matches the motion region, as the associated template may not be the best. This applies to the merge-split and disappear-reapers situations.

Each template has the following information associated with it:

N	Number of tracked objects
T^i	i^{th} tracked objects
T_f^i	i^{th} tracked object's last registered frame number
T_t^i	i^{th} tracked object's time to live value, where $t \in \{0,..,TTL_{MAX}\}$
T_m^i	i^{th} tracked object's merged flag, where $m \in \{0,1\}$
T_a^i	i^{th} tracked object's assigned flag, where $a \in \{0,1\}$
T_C^i	i^{th} tracked object's centroid location (row, column)
T_R^i	i^{th} tracked object's bounding rectangle (left, top, right, bottom)
T_v^i	i^{th} tracked object's registration offset vector
T_s^i	i^{th} tracked object's motion speed vector
T_M^i	i^{th} tracked object's associated motion bounding rectangle (left, top, right, bottom)

Each motion region has the following information associated with it:

K	Number of motion regions in frame f
M^j	j^{th} motion region in a frame f
M_C^j	j^{th} motion region's centroid location (row, column)

M_R^j j^{th} motion region's rectangle (left, top, right, bottom)

$\vec{M}_T^j$ j^{th} motion region's vector of associated templates

Constants used in the selective hypothesis tracking include:

TTL_{MAX} Maximum time to live beyond last know registration, based on frames per second $\alpha f(_{fps})$

w_p Cost weight factor for position and distance offset

w_d Cost weight factor for direction difference

w_s Cost weight factor for size difference, where $w = w_p + w_d + w_s = 1$

P_{MAX} Maximum distance as a function of block size, $\beta f(_{BLOCK})$

S Speed update decay factor, where $S \in \{0.0..1.0\}$

L_{MAX} Maximum number of iterations if convergence is not achieved

Selective hypothesis tracking algorithm processes each frame, aligning and computing the minimum cost between known templates and current motion regions. If

there are no templates created and there is motion in the current frame, then templates are created and new next frame is processed.

If $N = 0$ and $K(f) > 0$

Create initial templates T^i

$N = K(f)$

Go to next frame

For every active template perform image alignment of the templates region of interest to the current frame. Compute the offset vector for all templates that are not merged.

For each template repeat, $i = 1 : N$

If $T_t^i = 0$ or $T_m^i = 1$ go to next i

$T_a^i = 0$

$F = T_f^i$

Compute spatial gradients based on T_R^i region

$F_x^2(T_R^i)$

$F_y^2(T_R^i)$

$F_x F_y(T_R^i)$

$C = L_{\max}$

$G = f$

Repeat

$$G_{warp} = h\left(G, \begin{bmatrix} u \\ v \end{bmatrix}\right)$$

$F_t = (G_{warp} - F)$

Compute temporal gradients

$F_t F_x(T_R^i)$

$F_t F_y(T_R^i)$

Compute offset

$$\begin{bmatrix} u_e \\ v_e \end{bmatrix} = \begin{bmatrix} \sum F_x^2(T_R^i) & \sum F_x F_y(T_R^i) \\ \sum F_x F_y(T_R^i) & \sum F_y^2(T_R^i) \end{bmatrix}^{-1} \begin{bmatrix} \sum F_t F_x(T_R^i) \\ \sum F_t F_y(T_R^i) \end{bmatrix}$$

$u = u - u_e$

$v = v - v_e$

$T_v^i = \langle u, v \rangle$

Update the error estimation

$E(u,v) = \sum (F_x(T_R^i)u + F_y(T_R^i)v - F_t(T_R^i))^2$

$C = C - 1$

Until $E(u,v) < \varepsilon$ or $C = 0$

TM = FALSE

For $j = 1:K$

If $T_m^i \neq 1$ then

$P \leftarrow (T_C^i + T_v^i) \in M_R^j$

Else

$P \leftarrow (T_C^i + T_s^i) \in M_R^j$

If $T_t^i = TTL_{\max}$ and P = TRUE

$\vec{M}_T^j = \vec{M}_T^j + T^i$

TM = TRUE

If TM is FALSE

Check if there is a motion blob in the vicinity of this template

If any $\left| M_C^j - T_C^i \right| < P_{MAX}$ then

$T_t^i = T_t^i - 1$

Else

$T_t^i = T_t^i - \delta$, where $\delta > 1$

Process motion blobs that have more than one template (merged condition),

For $j = 1:K$ and $\vec{M}_T^j > 1$

For $i = 1:\vec{M}_T^j$

$T_f^i = f$

$T_t^i = TTL_{MAX}$

$T_m^i = 1$

$T_a^i = 1$

$T_C^i = T_C^i + T_s^i$

$T_R^i = T_R^i + T_s^i$, note: T_C^i and T_R^i are bound by M_R^j

$T_M^i = M_R^j$

Process motion blobs that have one or no registered templates based on minimum cost estimation,

For $j = 1:K$ and $\vec{M}_T^j \le 1$

$C_{best} = \infty$

$T_{best} = 0$

For $i = 1:N$ and $T_t^i > 0$ and $T_a^i \neq 1$

$\Delta p = \left| M_C^j - T_C^i \right|$

$\Delta s = \left| M_R^j - T_R^i \right| / (M_R^j + T_R^i)$

$\Delta d = \left| \arctan(T_s^i) - \arctan(\left| M_C^j - T_C^i \right|) \right|$

$\Delta t = (TTL_{MAX} - T_t^i) / TTL_{MAX}$

Compute the cost

$C = w_p \Delta p + w_d \Delta d + w_s \Delta s + \Delta t$

If $C < C_{best}$ and $\Delta p < P_{MAX}$

$C_{best} = C$

$T_{best} = T^i$

If $T_{best} = 0$

Create new template based on M^j

Else

$$T^i = T_{best}$$
$$T_f^i = f$$
$$T_t^i = TTL_{MAX}$$
$$T_m^i = 0$$
$$T_a^i = 1$$
$$T_s^i = S \bullet T_s^i + (1 - S) \bullet (M_C^j - T_C^i)$$
$$T_C^i = M_C^j$$
$$T_R^i = M_R^j$$
$$T_M^i = M_R^j$$

Each new frame containing motion regions is evaluated against know templates. New template is only created if a motion region has no associated template based on image alignment, predicted position, or minimum cost computation. New template's characteristics are only based on an unassociated motion region. Once the association is established, template's predicted velocity is still computed despite the image alignment calculation, as it may be necessary to use predicated position when templates merge.

Templates that are not within any motion region have their time to live T_t^i decreased. Once this value reaches zero, the template will no longer be used during the association with the motion region. This may provide false results when an object disappears and reappears beyond the time to live timeframe. A higher level template matching method may be used to solve this problem, such as the swarm template matching presented in Section 6.1.

6.5 Results

Two methods are used to evaluate selective hypothesis tracking results: the visual inspection of identified tracking objects and comparison of tracking centroids to independent ground truth data. The simplest visual evaluation involves tracking a single object that appears and disappears in the field of view, without any obstruction or merging with other objects. More challenging scenarios involve several objects merging and splitting where two or more objects cross paths as observed by the camera and some objects become hidden during the merging. The tracking algorithm must predict the possible location of each individual object despite the fact that the motion detection only provides a single motion rectangle. In this scenario, the image registration technique will not work as possibly only one of the merged objects is in the foreground. The known velocity of each object before the merge occurred is used to update the predicted position of each object. The predicted position is then bound by the observed motion rectangle, limiting the objects position to within the motion rectangle.

Another challenging tracking sequence involves an object that appears and disappears within the field of view. This may occur when object disappears behind a tree, building, or a parked car and reappears within a few seconds later. This scenario has one major difference from the objects that merge: there is no motion rectangle presents. When object that is tracked disappears, the corresponding motion rectangle is not present. The algorithm keeps tracking templates for some period of time in case this template reappears later.

There is one limitation to this scheme. An object that disappears and reappears much later than is allowed by the algorithm, a new template is created instead of matching to templates already seen by the system. The splitting of single template into multiple objects is a difficult case of label assignment. The single label of the object before the split must now be assigned to one of the motion regions after the split.

An example of two objects crossing the same path in the field of view is shown in Figure 6.6. In frame 863, object 3 (van) and object 4 (group of people) approach each other. In frame 898 the merge occurs with object 3 is in the foreground. Image registration of object 4 is impossible as it is partially visible. The predicted position based on last know motion velocity along with bounding motion rectangle provide a sufficient location of individual objects (frame 923). In frame 963 both objects split and continue in their respective directions while maintaining the correct labels.

Single object split is shown in Figure 6.7. Single object 4 (group of people) is approaching parked cars in frame 1043. In frame 1099 a person left one of the parked cars while object 4 passed next to it. In this situation there is a single motion blob corresponding to tracking object 4. While this motion region is expanding due to the fact that a single person is walking in the opposite direction to the object 4, there is still single object being tracked. The split occurs in frame 1141, where the single person is assigned new tracking label 6 while object 4 continues along its course maintaining it own label, as seen in frame 1214.

Example of an object that disappears and later reappears is shown in Figure 6.8. Infra 3 is a thermal infrared video sequence showing single person walking behind two

trees. In frame 233 object 1 approaches the first tree and becomes invisible to the camera in frame 252. It reappears in frame 263 as the tracking algorithm keeps the same object label and does not create a new template. Object 1 is tracked continuously until frame 431 when it disappears again. Later it reappears again as object 1 in frame 461 before leaving the field of view.

The selective hypothesis tracking algorithm may also be applied to tracking microscopic objects, such as individual cells. Figure 6.9 shows tracking of a single cell while it is transforming its shape from a star-like figure (frame 49) to an elongated cigar-like figure (frame 97). Frames 67 and 86 showed the position change via the trailing template centroids. This type of visual verification of the tracking algorithm is supplemented by the comparison of tracking centroid to the independent ground truth data.

Independent ground truth data was used not only to test the proposed motion detection method but also to verify the selective hypothesis tracking algorithm. Split 1 video along with its ground truth data is used to evaluate the proposed tracking method. Once each tracking object is identified, their centroids were compared to the ground truth data. Figure 6.10 displays the projection of all ground truth data onto a single frame, this includes individual object projection and group projection. A close up of the projection were two objects merged and split is also shown. On average, the tracking centroid distance from ground truth data was 5.2 pixels with standard deviation of 2.6 pixels for Split 1 video, while the motion block size was $4x4$.

Figure 6.6: Campus 1 cross-tracking. Example of two objects crossing paths, when one becomes obscured by the foreground objects. Objects 3 and 4 approach each other. Object 4 is less visible as object 3 is in the foreground, and finally objects 3 and 4 continue in opposite directions. Campus 1 video sequence 863-963.

Figure 6.7: Campus 1 single split tracking. Example of single object splitting into two distinct objects, when the single object retains most of its previous shape and direction. Object 4 approaches parked cars when a person appears inside object 4 and walks in the opposite direction. Object 4 splits from and continues while new object 6 is created. Finally object 4 and 6 continue in the opposite directions. Campus 1 video sequence 1043-1214.

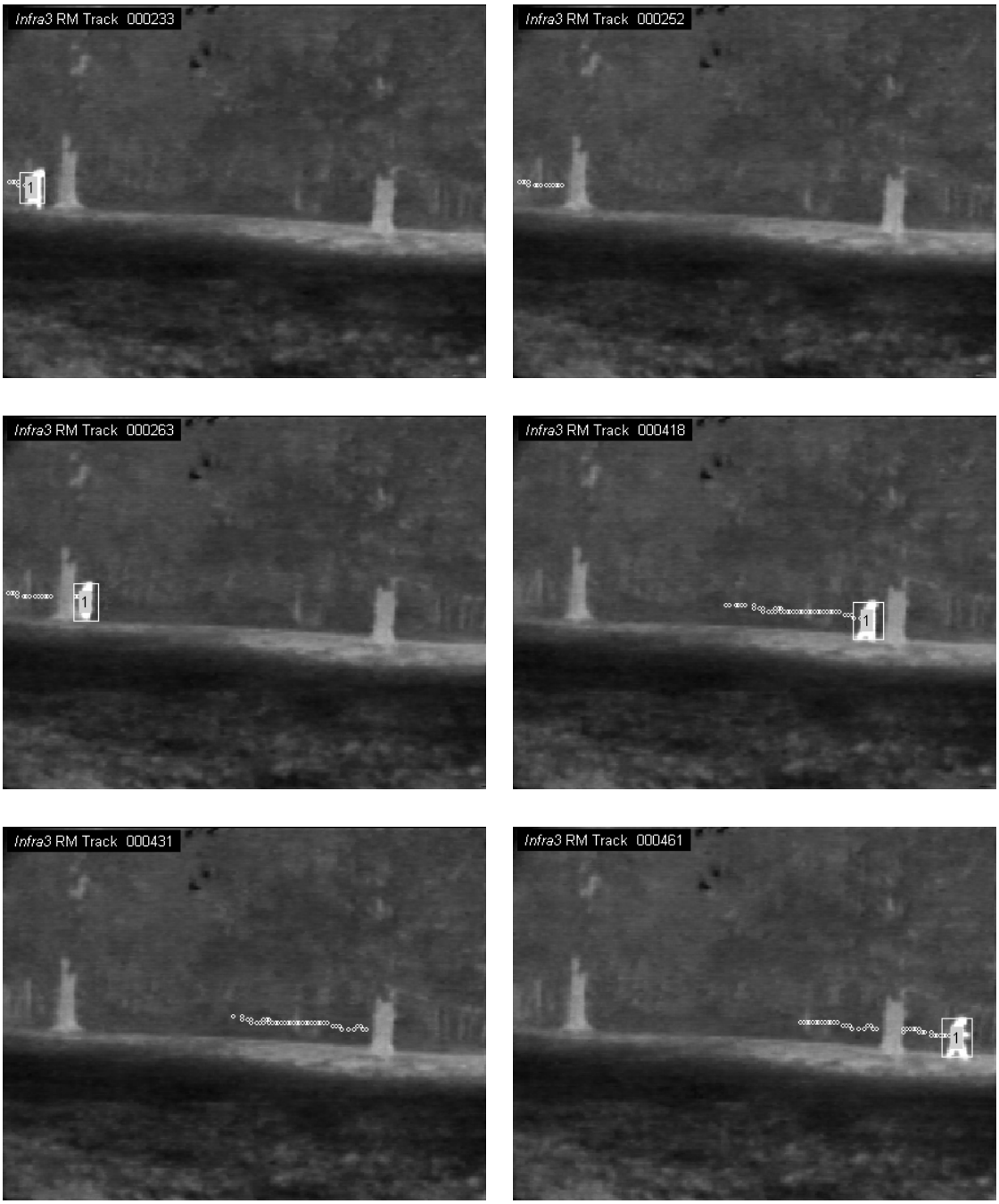

Figure 6.8: Infra 3 disappear-reappear tracking. Example of tracking an object that disappears from the field of view. Person appears before and after each tree while label is maintained. Person is hidden behind a tree, as no active object is being tracked. Infra 3 video sequence 233-461.

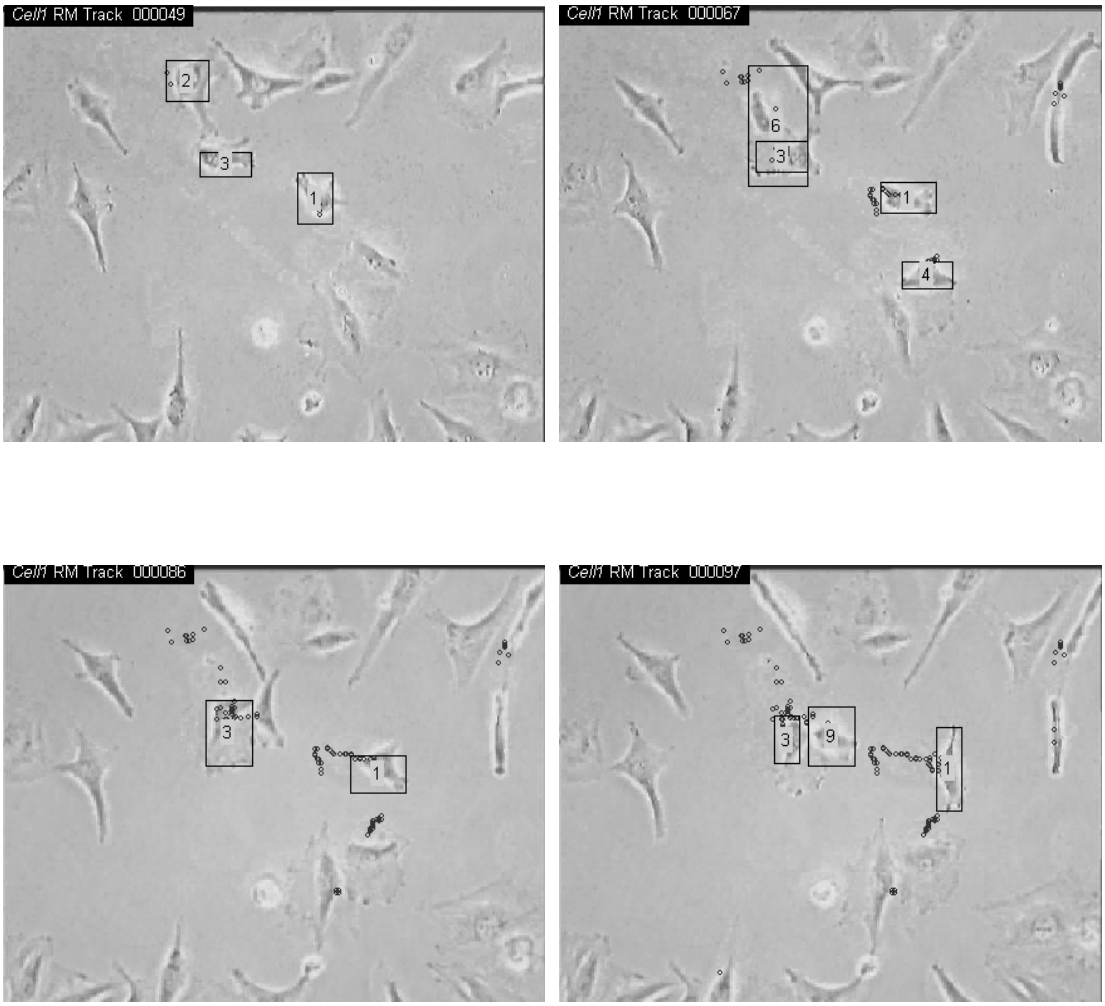

Figure 6.9: Cell 1 change size tracking. Example of tracking an object that changes size, direction, and position. Cell labeled ‘1’ starts as a compact star-like shape and changes into a vertically elongated cell. Cell 1 video sequence 49-97.

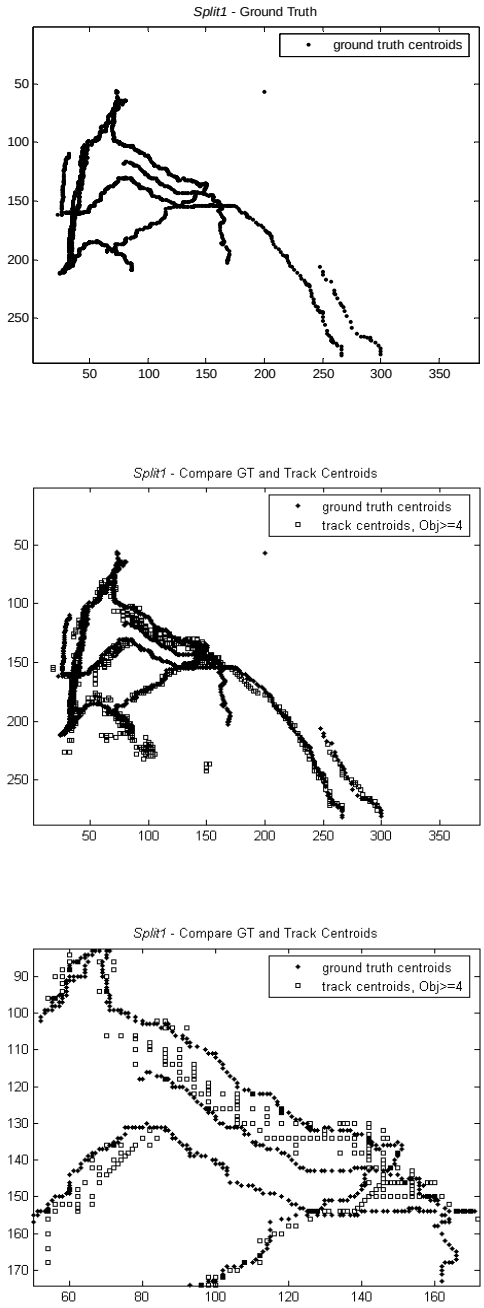

Figure 6.10: Split 1 tracking ground truth comparison. Comparing ground truth data and tracking centroids of Split 1 video. Showing original ground truth data, Ground truth data with superimposed tracking centroids, and close-up of the location when two objects meet and walk together.

6.6 Future Work

A map of active regions may be constructed from tracking centroids when they are projected onto camera's field of view. The projection will dynamically identify most active regions which it turn may be assigned a higher detection priority. An inverse of this may also apply, the least active regions of the field of view may be assigned higher detection priority, including an alarm generation when motion is detected. Figure 6.11 shows all the tracking centroids of Campus 1 video sequence projected onto a single frame. The alignment of tracking centroids to the active part of the field of view identifies the road as the most active motion region.

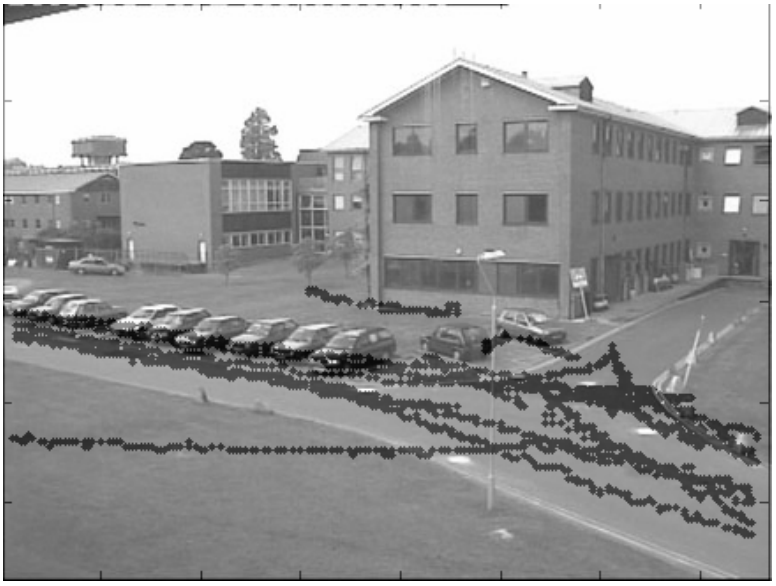

Figure 6.11: Campus 1 track centroid projection. Superimposing tracking centroids on a field of view allows building a map of active regions of Campus 1 video.

Information from the tracking data may be compiled to create a timeline of motion events. This timeline identifies the object's unique number along with a picture of the object. Analysis may reveal what objects occupied the same timeframe, and if any object disappeared and reappeared later. Further analysis may include the identification of the object's type, such as a human or a vehicle. Figure 6.12 shows the timeline of Campus 1 video sequence with each object's unique label, picture and timeframe.

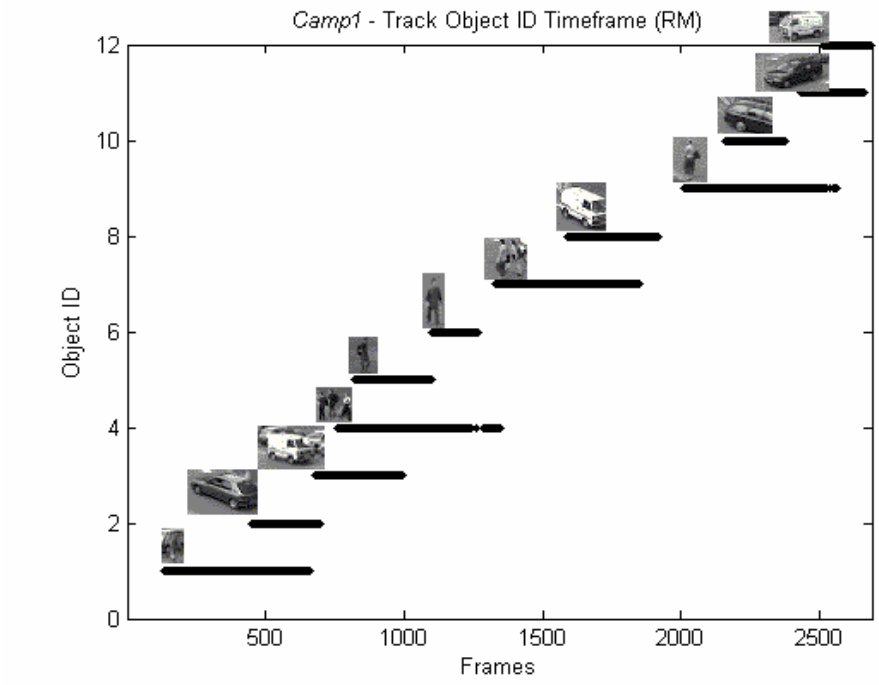

Figure 6.12: Campus 1 tracking timeline. Timeline of activity in Campus 1 video shows type of object at each time frame and total duration of object tracking.

CHAPTER 7

REAL-TIME MOTION DETECTION SOFTWARE

The ViVi Lab Motion Detector software was written in C++ for a Microsoft® Windows® platform. A faithful implementation of the motion detection, dynamic distribution learning and outlier detection was achieved. The Principal Components Analysis classical method for dimensionality reduction and exploratory data analysis was based on the Karhunen-Loeve expansion.

There are several USB-based security video cameras manufactured around the world. The uniformity with which cameras are available to the software developer under the Windows platform was the main reason for selecting the USB-based system. High quality USB video cameras also provide a direct access to the frame buffer using a proprietary application programming interface (API). This API may be used to manipulate the video frames instead of the generic Windows function calls. Regardless of the method used, any USB camera and corresponding software provide a viable solution for a commercial-based security product.

The incoming video frame size was set to 320x240 pixels in 24-bit RGB color resolution. The system is capable of showing live video stream at 30 frames per second from USB-based color camera, and 14 frames per second from an infrared camera

(camera limitation). While detecting motion, a frame rate of about 25 frames per second was achieved for the color camera.

Figure 7.1 shows the main screens of the motion detection system. Video resolution and camera control is accessible from the File menu. Settings for the initial alarm threshold is available form the Video menu. Live video stream is displayed in the main view, along with estimated motion frames per second (fps) and a current timestamp. Once video input size is selected, the main view will automatically resize to accommodate the full width and height of the video frame.

The initialization stages required to configure the ViVi Lab Motion Detector software is shown in Figure 7.1. The first step involves the creation of the projection matrix for each block in the frame. If frame size is set to 320x240 pixels and block size is 8 pixels, the software must create 1200 projection matrices. Next initializing step involves computing the average and standard deviation of each block's motion measure, used by the dynamic distribution learning thresholding algorithm. Motion and activity detection is processed at 25 frames per second (color camera) and shows a number of motion block enclosed by square brackets (Figure 7.1).

The real-time motion detection and increased activity detection was tested indoors where captured video frames showed an intruder in the filed of view. Changes in lighting conditions did not trigger any false alarms, as the dynamic distribution learning algorithm constantly monitored the video stream. Figure7.2 shows one RGB color video frame with detected alarm situation and alarm timestamp. The estimated motion detection and activity detection processing speed is between 25 to 30 frames per

second for the full-color 320x240 RGB frame size. Person entering the field of view triggered a sequence of video frames to be saved in the remote image depository server. Figure 7.3 shows detected motion of packages traveling on an industrial package-sorting conveyer belt. The surrounding area was flooded with fluorescent light, however motion was detected correctly.

Software system parameters that are configurable include motion detection dynamic distribution learning thresholding parameters C_1^m, C_2^m, and u^m, and corresponding activity detection flag D^a, and dynamic distribution learning thresholding parameters C_1^a, C_2^a, and u^a. These parameters may be changed while the system is detecting motion blocks, that is, when it is actively processing frames. This allows fine tuning the parameters for a given situation, such as type of camera used (color or infrared), or a specific location (indoor or outdoor).

Additional motion detection tests are performed indoors under fluorescent lights. Figure 7.2 shows a typical office setting with detected motion, and Figure 7.3 shows an industrial setting where boxes are moving on a conveyer belt.

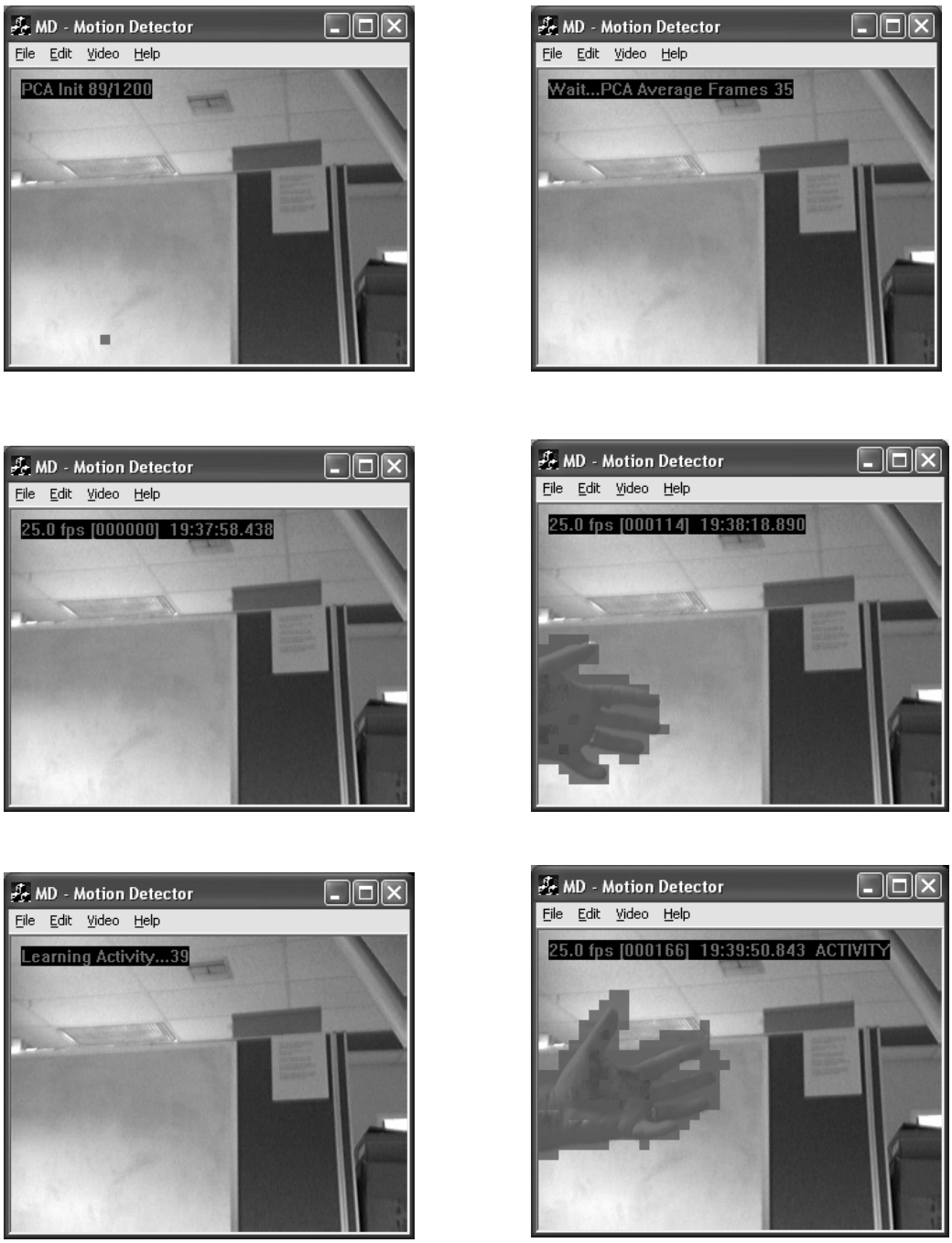

Figure 7.1: Stages of Motion Detector software. Create PCA projections; Average frames; Running, no motion detected; Running, motion detected, number of blocks is 114; Learning activity; Increased activity detected, number of blocks is 166.

Figure 7.2: Indoor motion detection. Showing no motion detected where strong fluorescent light is present. Motion detected, number of motion blocks is 126.

Figure 7.3: Conveyer belt motion detection. Showing no motion detected where multiple fluorescent lights are present. Motion detected, number of motion blocks is 42.

CHAPTER 8
CONCLUSION

The proposed spatiotemporal texture motion detection algorithm is a much simpler but also a more adequate model for motion detection in color, infrared and thermal surveillance videos. It can significantly reduce the processing time in comparison to the Gaussian mixture model, due to smaller complexity of the local variation computation. Moreover, the local variation based algorithm remains stable with higher dimensions of input data, which is not necessarily the case for an Expectation-Maximization type algorithm, used for Gaussian model estimation. The selective hypothesis tracking based on spatiotemporal motion regions is the foundation for more sophisticated object classification algorithm. It provides for the robust tracking of objects that merge and split, and objects that appear and disappear. The tracking method combines the best image alignment algorithm with improved minimum cost estimation algorithm.

8.1 Future Work

Future work may include the expansion of tracking algorithm to classify tracking objects as human, cars or planes; and building a map of objects that occupy the

same timeframe. The most interesting work may include combining one color camera with one thermal infrared camera to create motion detection and tracking system that takes advantage of both sensors and combines the results. Problem of an object that disappears from the filed of view and reappears much later is also not solved completely. A more precise image registration must be used to see if the returning object is the same as seen previously or it is in fact a new object.

REFERENCES

[1] D. Buttler, S. Sridharan, and V.M. Bove, “Real-time adaptive background segmentation”. *In Proc. IEEE Int. Conf. on Multimedia and Expo (ICME)*, Baltimore 2003.

[2] R.T. Collins, A.J. Lipton, and T. Kanade, “Introduction to the Special Section on Video Surveillance”, *IEEE PAMI* 22(8), pp. 745–746, 2000.

[3] Devore, J. L., *Probability and Statistics for Engineering and the Sciences,* 5th ed., Int. Thomson Publishing Company, Belmont, 2000.

[4] Duda, R., P. Hart, and D. Stork, *Pattern Classification,* 2nd ed., John Wiley & Sons, 2001.

[5] Flury, B. *A First Course in Multivariate Statistics*, Springer Verlag, 1997.

[6] I. Haritaoglu, D. Harwood, and L. Davis, “W4: Real-Time Surveillance of People and Their Activities”, *IEEE PAMI* 22(8), pp. 809–830, 2000.

[7] R. Jain, D. Militzer, and H. Nagel, “Separating nonstationary from stationary scene components in a sequence of real world TV images”. *In Proc. IJCAI*, 612–618, Cambridge, MA, 1977.

[8] Jolliffe, I. T, *Principal Component Analysis*, 2nd edn., Springer Verlag, 2002.

[9] O. Javed, K. Shafique, and M.A. Shah,. “Hierarchical approach to robust background subtraction using color and gradient information”. *In Proc. IEEE Workshop MOTION*, 22-27, Orlando, 2002.

[10] N. M. Oliver, B. Rosario, and A. P. Pentland, “A Bayesian Computer Vision System for Modeling Human Interactions”, *IEEE PAMI* 22(8), pp. 831–843, 2000.

[11] L.J. Latecki, R. Miezianko, and D. Pokrajac. “Motion Detection Based on Local Variation of Spatiotemporal Texture”. *CVPR Workshop on OTCBVS*, Washington, July 2004.

[12] Temple University ViVi Lab video results and data URL: http://knight.cis.temple.edu/~video/VA/.

[13] Remagnino, P., G. A. Jones, N. Paragios, and C. S. Regazzoni, eds., *Video-Based Surveillance Systems*, Kluwer Academic Publishers, 2002.

[14] C. Stauffer, and W. E. L. Grimson, “Learning patterns of activity using real-time tracking”, *IEEE PAMI* 22(8), pp. 747–757, 2000.

[15] Westwater, R., and B. Furht, *Real-Time Video Compression*: Techniques and Algorithms, Kluwer Academic Publishers, 1997.

[16] C. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland, “Pfinder: Real-time Tracking of the Human Body”, *IEEE PAMI* 19(7), pp. 780–785, 1997.

[17] S. Glisic, Z. Nikolic, D. Pokrajac, and P. Leppanen, “Performance Enhancement of DS Spread Spectrum systems: Two Dimensional Interference Suppression,” *IEEE Trans. Communication*, Vol. 47, No. 10, pp.1549-1560, 1999.

[18] W. Niu, J. Long, D. Han, and Y.-F. Wang. Human “Activity Detection and Recognition for Video Surveillance”. *In Proc. IEEE ICME*, 2004.

[19] J. Davis, and M. Keck, “A two-stage approach to person detection in thermal imagery” *In Proc. Workshop on Applications of Computer Vision*, January 2005.

[20] Haralick, R.M., and L.G. Shapiro, *Computer and Robot Vision*, Volume I, Addison-Wesley, 1992.

[21] D. Pokrajac, and L.J. Latecki: “Spatiotemporal Blocks-Based Moving Objects Identification and Tracking”, *IEEE Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS),* October 2003.

[22] J. Omar, and M. Shah, "Tracking and Object Classification for Automated Surveillance", *The seventh European Conference on Computer Vision,* Copenhagen, May 2002.

[23] L.J. Latecki, R. Miezianko, and D. Pokrajac. “Activity and Motion Detection Based on Measuring Texture Change”. *International Conference on Machine Learning and Data Mining MLDM´2005*, Leipzig, June 2005.

[24] EC Funded CAVIAR project IST 2001 37540, found at URL: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.

[25] Performance Evaluation of Tracking and Surveillance (PETS) repository videos Campus 1 and 3: ftp://pets.rdg.ac.uk/PETS2002//DATASET1/TESTING/***/.

[26] R. C. Eberhart, and J. Kennedy, “A new optimizer using particle swarm theory” In Sixth International Symposium on Micro Machine and Human Science, Agoya, Japan, 1995.

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