Video Motion Detection and Tracking for Surveillance Applications

by

Joey Quevillon
B.Eng, Lakehead University, 2006

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
MASTER OF APPLIED SCIENCE
in the Department of Electrical and Computer Engineering

© Joey Quevillon, 2012
University of Victoria
All rights reserved. This thesis may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.

Supervisory Committee

Video Motion Detection and Tracking for Surveillance Applications

by

Joey Quevillon

B.Eng, Lakehead University, 2006

Supervisory Committee

Dr. A. Branzan-Albu, Department of Electrical and Computer Engineering
Supervisor
Dr. Mihai Sima, Department of Electrical and Computer Engineering
Departmental Member
Dr. George Tzanetakis, Department of Computer Science
Departmental Member

Abstract

Supervisory Committee
Dr. A. Branzan-Albu, Department of Electrical and Computer Engineering
Supervisor
Dr. Mihai Sima, Department of Electrical and Computer Engineering
Departmental Member
Dr. George Tzanetakis, Department of Computer Science
Departmental Member

These days, video surveillance is an important security asset to control theft, trespassing or traffic monitoring for any physical systems, whether personal or commercial. Implementing a surveillance system can allow people to get an idea of what is going on without the physical need to be there. In the classical video surveillance installations, there is a need for a human operator to consistently watch the video feed to see if there is any interesting activity. An intelligent, computer vision-based motion detector eliminates the need for constant surveillance by an operator by notifying an interested party that there is relevant motion in the area being monitored.

Pan tilt zoom (PTZ) cameras can be used to track an object of interest moving throughout a scene. However, in classic systems, this again would require the operator to manually move the PTZ camera to get the subject in scope. The goal of this work is to eliminate the operator from controlling the system. With the proposed automated tracking approach, an object found in a scene can be tracked automatically and followed until the target is out of camera scope.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank Dr. Alexandra Branzan-Albu, for her continuous encouragement, support and guidance throughout my research process.

1. Introduction

Pedestrian detection surveillance systems deal with the monitoring of stationary and moving objects through a specific scene in real time. These systems aim to provide an autonomous way to track and understand the motions of objects observed by surveillance cameras. Intelligent visual surveillance is broken into the following steps: video motion detection, object recognition, and object tracking. Increasing surveillance and security requirements of public, military, and commercial companies have created the need to create and implement intelligent video surveillance systems. For example, public transportation systems can help monitor and respond to situations that are of interest to the public, whether it is an individual who is a treat or a whole crowd.

This field of intelligent video surveillance systems is continuously increasing in popularity and is the topic of many publications in IEEE conferences on visual surveillance [51-61] and in peer-reviewed journals on visual surveillance [57-59] or on human motion analysis [60].

1.1. Evolution of intelligent surveillance systems

The field of video surveillance began with the introduction of CCTV cameras and systems. CCTV (Closed-circuit television) systems began as a closed loop system that would display analog video on a various fixed number of monitors with no digital processing done to the video feed. Different CCTV systems could be integrated to be used with each other [61] but analog techniques were still used for the distribution and storage of the video. Conventional CCTV cameras would generally use a CCD (Charged Coupled Device) to capture the images that would cause some image degradation. Advances in technology allowed for improvements to the systems that are now known as second generation surveillance systems which can take advantage of the initial digital format of the captured images. The initial digital format of those captured images is

where the majority of the algorithms in the field operate to produce automatic real-time surveillance.

As the technology behind CCTV cameras evolved, so too did the visual information processing techniques. Most of the research focused on automating the visual surveillance process, thus on eliminating the need for human operators. Automated video surveillance involves motion detection, object detection and visual servoing. Video tracking with the use of Pan-Tilt-Zoom cameras allows for tracking an object through the camera's whole range of motion automatically.

1.2. Motivation

A primary constraint to pedestrian detection is the need to have an operator monitoring the screen for activity. Moreover, the performance and safety requirements cannot be left to risk and must answer to real world problems and situations. The task of designing a system to replace an operator in order to track a human target is challenging, particularly with respect to speed and accuracy. Using motion detection and feature points we can expand upon a system of stationary cameras by adding PTZ servoing. This system would serve as a useful tool to automatically track pedestrians throughout a surveillance zone and meet requirements for real world issues.

One primary use for intelligent surveillance systems is for counter terrorism. As an example, Lower Manhattan spent over 150 million dollars on a surveillance system that incorporates over 3000 fixed cameras [62]. The use of PTZ cameras instead of fixed ones could have reduced the number of cameras and lowered the project cost of the intelligent surveillance system.

Intelligent surveillance systems offer various advantages over traditional CCTV systems. They provide an interactive and automated way to monitor and detect activity over a large area without the need for continuous monitoring by a human operator. Intelligent surveillance systems can notify an operator of important events in their system such as trespassers in a secure lot or traffic congestion in an ITS (Intelligent Transportation System).

The few intelligent surveillance systems that do include PTZ cameras still require an operator to track a target through its field of view after notification from the intelligent system[64]. These issues were the main motivation for the research described in this thesis. More specifically, this thesis contributes to the design of a PTZ human pedestrian tracking system solution that utilises novel PTZ servoing methods that integrate various Computer Vision and Machine Learning techniques.

1.3. Thesis Goals and Scope

The primary goal of this thesis is to propose a novel approach for visual surveillance of pedestrians based on PTZ servoing. This is accomplished through the development of a method capable of acquiring a moving human target through a video scene, finding useful features to track on the target, and then tracking the target throughout the camera's possible range of motion. In order for the system to properly locate the moving target in the scene, appropriate background subtraction methods are used and combined with feature tracking methods to provide motion vectors of the target's path. Consequently, a mechanism that interprets the motion vectors and provides an estimation of the target's next position controls the PTZ camera's pan or tilt movement.

1.4. Organization of Thesis

This chapter has provided an introduction to intelligent video surveillance systems through the method of human motion detection and tracking, discussed its implications, and defined the broad scope of the thesis.

Chapter 2 provides an overview of related work and focuses on methods developed to solve motion analysis and object tracking problems. Tracking algorithms with special focus on human targets are explored.

Chapter 3 describes the proposed approach. This chapter is divided into six sections that detail each module of the proposed approach, namely decompression of the video

input, background subtraction, binary image analysis, object detection, object tracking, and PTZ servoing.

Chapter 4 presents the efforts that were necessary for the systemic integration of the proposed approach. This section explains the process it takes to implement the proposed academic algorithm into a real world solution.

Chapter 5 describes the experimental results and performs a statistical analysis of the measured accuracy of the tracking algorithms implemented in this thesis.

Chapter 6 draws conclusions and describes potential future work directions.

2. Related Work

This chapter surveys related work in pedestrian tracking with computer vision techniques and discusses different approaches to solving specific problems in this area. Background subtraction, object detection, motion tracking, and PTZ servoing algorithms are surveyed. Finally, the open research questions that need to be further explored are examined. Trying to solve some of the open research questions constitutes the main focus for the remainder of this thesis.

2.1. Background Subtraction

Background subtraction is widely used to detect moving objects in a scene acquired with a stationary camera. Many methods have been proposed, ranging from simple to very involved. The computationally simple approaches aim at providing maximum speed while the more complex ones aim at providing the highest accuracy and robustness with respect to a high degree of background clutter.

Wren et al. [1] proposed a running Gaussian average method. This method fits a Gaussian probability density function on each of the pixel values based on a certain square, movable window. The algorithm works with grey scale images. A running average is used to update the mean value, which means that the algorithm does not need a learning phase. This method is fast, computationally efficient, and has very low memory requirements. The approach in [1] is limited however by its assumption that the background scene is significantly less dynamic than the human subject in the foreground. For instance, if the lighting in the scene changes drastically, the algorithm in [1] is very likely to produce false positives.

Versions of the method in [1] have been proposed by other research groups. Lo and Velastin [2] used a temporal average of the last *n* frames instead of a running average. Cucchiara et al.[3] showed that similar results can be obtained in the background model if one computes the median of the *n* frames even when they are subsampled by a factor

of 10. A disadvantage the [2] and [3] versus [1] is that they must keep a running buffer of values, which needs additional memory.

Stauffer and Grimson [4] decided to implement a mixture of Gaussians because they considered the fact that over time different background objects, such as stationary vehicles that occasionally leave and enter the scene, could appear at the same pixel location. There can be two reasons for this. The first involves a permanent change such as a background object being first present in the field of view and disappearing afterwards. The second reason is a quickly changing element such as tree leaves or other moving objects that shouldn't be considered as foreground objects but would not fit a static background model. The study in [4] implements a multi-distribution background model that is able to handle multiple objects in the background. The algorithm is based on an adaptive background subtraction method that models each pixel as a mixture of Gaussians and uses an online approximation to update the model. Power and Schoonees' [5] describe a practical implementation of the Stauffer and Grimson [4] method as they felt that the original paper [4] lacked the values for all model parameters for implementation.

Elgammal et al. [6] present a new non-parametric background model, which can handle cluttered background scenes and ignore small motions such as the ones of tree branches and leaves. Their Kernel Density Estimation technique adapts quickly to changes in the scene and guarantees a smooth continuous histogram. Their background probability density function is a sum of Gaussian kernels which is centered in the most recent background values.

Comaniciu and Meer [7] propose a mean-shift vector approach. They use a nonparametric estimator of density gradient, the mean shift, which is employed in the joint, spatial-range domain of gray level and color images for discontinuity preserving filtering and image segmentation. This approach is able to detect the main modes of the true probability density function directly from sample data. The downside of their approach (and of the similar one in [8]) is that it is very computationally expensive and not suitable for real-time systems. Piccardi and Jan [9] apply some optimizations that improve the performance of the Comaniciu and Meer's [7] algorithm. Their main contribution is the description of each pixel location by a distribution (or a mixture of

distributions) to reflect the changing nature of the scene's background. Han et al. [10] use the same method as Piccardi and Jan [9], however they only use the mean-shift vector for model initialization when it is off-line. This Sequential Kernel Density Approximation proposed in [10] allows the system to be more computationally effective online since the model has already been initialized offline. The approach updates the real-time model by simple heuristics when online. Their method [10] uses an efficient method to sequentially propagate the density modes over time.

Oliver et al. [11] base their method on eigenvalue decomposition and a statistical Bayesian approach to modeling that includes both prior knowledge and evidence from data, assuming that the Bayesian approach provides the best framework for coping with small data sets and novel behaviors. They perform segmentation by eigenbackground subtraction. The method is computationally expensive and requires a learning phase to perform the subtraction.

To summarize, the current state-of-the-art in background subtraction is focused on statistical modelling of the background. The necessity of a learning phase indicates that such approaches would perform poorly under quick changing conditions. This also indicates that statistical methods would be impractical for a PTZ camera system as they would need to re-learn the environment at each pan or tilt.

2.2. Feature Detection

Feature detection is a required process for feature-based object tracking. To track an object through the entire field of view of the camera, feature points located on the object need to be determined. Feature points belonging to adjacent frames are compared via correlation in order to establish inter-frame feature correspondence. Connor and Limb [12] use correlation functions and power density spectra to show that the frame-difference signal arising from temporally uncorrelated noise is fundamentally different in areas where there has been foreground movement compared to areas with no movement at all. Cafforio and Rocca [13] describe other various techniques to measure small displacements of television images. They use the differences of two successive frames to

approximate a linear combination of the components and thus to measure the displacement of the object. Ryan et al [14] use a prediction of correlation in errors to determine the movement of the object; however, Wood [15] provides evidence that the previous method falls victim to automatic correlation issues. Autocorrelation of the errors, which themselves are unobserved, can generally be detected because it produces autocorrelation in the observable residuals. Tian and Huhns [16] present four new approaches to obtain sub-pixel accuracy in the process of finding correlation between points in images. These methods are correlation interpolation, intensity interpolation, differential method, and phase correlation.

Burt et al.[17] present a novel idea of using the local sum-of-squared-difference (SSD) to track features. The matching cost of a given pixel is the squared difference of intensity values at a given disparity. Aggregation is then done by summing the matching cost over square windows with a constant disparity, and then the disparities are computed by selecting the minimal aggregated value at each pixel. The minimal aggregated value is then selected and used for tracking. Anandan [18] describes a method based on SSD that uses a hierarchical computational framework to determine the dense displacement fields from two successive images. Anandan [18] also optimized the tracking when the image displacement between successive frames was small by using a matching criterion over all small translations.

Shi and Tomasi [19] argue that there still remained an important issue that needs to be solved. They believed that it is important to concentrate on feature selection prior to feature matching. According to them, it is important to focus on features that have a physical meaning. Their proposed method implements a feature selection criterion that extends Newton-Raphson search styles to work under affine image transformations. Their approach uses what they call textures. A large and a small eigenvalue correspond to a unidirectional texture pattern. Two large eigenvalues can represent corners, salt-and-pepper textures, or any other pattern that can be reliably tracked. Because their search method locates features with these good textures, they state that their algorithm can detect occlusions, disocclusions and features that don’t correspond to points in the real world.

To summarize, feature detection for object tracking is a useful way to find the points that will need to be tracked throughout the video stream.

2.3. Object Tracking

The most straightforward way to follow an object using feature-based tracking is to exhaustively search every frame in the video sequence for the features to be matched. This technique is very time consuming and is not a practical solution to real time systems. One technique discussed by Barnea and Silverman [20] is the sequential similarity detection algorithm (SSDA). SSDA reduces the redundancy of exhaustively searching the image for each feature by performing a sequential search, which may be terminated before all the window pairs for a particular reference point are tested. However, this method only estimates the error for each displacement vector but doesn't specify a strategy for examination.

Other approaches such as Moravec [22] use a coarse-to-fine search strategy (in their case to guide the visual mapping of a robot rover). They concluded that they could more quickly correlate values in a very coarse version of an image by successively refining the position to find smaller and smaller areas in finer and finer representations. Their findings contributed to the pyramidal image search, which works with various image resolutions. Lucas and Kanade [23] present an algorithm that specifies the search order for the exploration of the entire possible space for the displaced object. Their algorithm uses a spatial intensity gradient at each point in the frame to predict the position of the object using a modified Newton-Raphson iteration. This method was originally proposed to produce dense results. However, because the algorithm can be applied to a smaller set of points (or blob) in an image it has become more commonly used for the tracking of sparse points.

Object tracking methods have proved useful to follow a set of features throughout a stationary video frame. It is unclear how well these methods would function when tracking objects with PTZ cameras, where ego motion is present as well.

2.4. PTZ Servoing

The aspect of a moving field of view (FOV) introduces new issues related to active vision. The paradigm discussed in [24] describes the dynamic visual interaction between the subject and the observer, in this case the camera. The idea of a moving FOV brings forth the issue of ego motion. Ego motion as referred by [25] is defined as the camera's motion relative to a rigid scene. The estimation of ego-motion was the first issue that needed to be solved for PTZ motion tracking. Ego motion estimation poses problems for object tracking as the object is moving through the scene and out of the field of view. This ego motion violates the rigid world assumption held by most fixed camera tracking algorithms because of the projective transformation that generates the image. In ego motion, the camera must pivot or rotate to track the object and not slide along horizontally or vertically with the object.

Murray and Basu [26] attempted solving the issue of tracking ego motion with a PTZ camera by using image map transformations. The image mapping is used to align images of different viewpoints so that static camera motion detection can be applied. Irani and Anandan's [27] approach is based on a stratification of the moving object detection problem into scenarios that gradually increase in complexity. The method estimates the dominant parameters of an affine transformation. Araki et al. [28] proposes a new method for detection and tracking of moving objects from a moving camera image sequence using robust statistics and active contour models. Their statistical method estimates background motion by using an iterative process of least squares to solve the linear system. Guo et al. [29] show that the previous system [28] can use Ullman and Basri's [30] linear combination representation for outlier detection to filter out mismatched points in their tracking. Ullman and Basri's [30] method represents 3D objects by linear combinations of 2D images. They show that for 3D objects, the set of possible images of a given object is embedded in a linear space spanned by a smaller number of views. Foresti and Micheloni [31] developed a feature-clustering technique adapted from the Lucas and Kanade model. Their clustering technique was further improved in Micheloni and Foresti [32] which added a feature rejection rule to eliminate outliers in the data features.

Tordoff and Murray[33] describe reactive visual methods to fixate on the object itself rather than computing and estimating the ego motion. Their method tracks the affine projection of the object's center of mass by using the concept of clustering feature points. The cluster of feature points is updated by two proposed methods. In the first method the cluster of points is transferred via the observed planar structure. In the second method, the cluster is transferred through the affine structure recovered by factorization.

Another PTZ tracking method consists in using information from a combination of cameras to track an object. The configuration of the cameras allows for the exploration of the wide FOV of the omnidirectional camera and the wide zoom range of the PTZ camera. Fleuret et al. [34] effectively combine a generative model with dynamic programming to accurately follow a group of up to six individuals across thousands of frames using a derived metric trajectory for each individual, then combining them to form the overall trajectory. Their approach estimates the probabilities of occupancy of the ground plane given the binary images obtained from the input images through background subtraction. Their method then combines those probabilities with a color and motion model and uses a Viterbi [21] algorithm to accurately follow individuals across frames. The downside to their approach is that the cameras mentioned in the paper need to all share the majority of their FOVs which isn't always possible in large scale implementations. Hampapur et al. [35] describe new issues of situational awareness that arise with multiple camera systems with different resolutions. One issue described is that, at the present time, the component technologies are evolving in isolation; for example, face recognition technology addresses the identity tracking challenge while constraining the subject to be in front of the camera. However, intelligent video surveillance technologies provide activity detection capabilities on video streams while ignoring the identity tracking challenge.

Scotti et al. [36] use a 360 degree catadioptric sensor in conjunction with a PTZ camera which leads to relative image portions from the sensor to rectify the pan, tilt and zoom parameters for the PTZ camera. These image portions are automatically adjusted to the PTZ dimensions and parameters by the system in order to track detected objects. This system is still left with some ambiguity and occlusion issues that can be solved using

Chen et al.'s coefficients [37]. Chen et al. [37] propose two methods (geometry and homography calibration) to overcome the need for a priori knowledge of the omnidirectional camera's projection model to solve the nonlinear spatial correspondences between the two cameras.

Olfati-Saber et al. [38] propose a theoretical framework in which a consensus could be reached for multi-agent network systems. This framework could be extended, as proposed by Soto et al. [39] to their application of a distributed estimation algorithm known as the Kalman-Consensus filter. This filter creates a process through which each camera comes to a consensus with its neighbouring cameras about the actual state of the target. This allows the system to reach a distributed estimation of the current position of the tracked objects. However, the system is still dependant on the ground-plane estimation that is done independently by each camera. Independent estimates in each camera can cause problems in a system that uses cooperative camera networks since the results might vary.

To address the issue of independent estimation one can use 3D stereo camera configurations. In this configuration, the system's processing is usually done in one dedicated camera node. Park et al. [40] propose a system which uses a look-up table to rank the cameras according to which camera can view the desired location. Their approach involves locating the two closest cameras first. Next, it implements differencing techniques and transformations to track the object. Lowe's [41] SIFT features or Meltzer and Soatto's [42] edge descriptors can then be used on the points obtained from the camera in the look-up table to solve for the matching points.

Another solution to independent depth estimation is through calibrated camera methods. Wan and Zhou's [43] method allows for solving the depth estimation issue analytically to determine the calibration matrices using a spherical rectification technique. Hart et al. [44]'s method uses epipolar geometry to control two humanoid robotic heads. This method captures the projective relationship between the cameras in a stereo vision system, assisting in the reconstruction of 3D information. Their approach uses an epipolar kinematic model that is based on optical properties of a stereo vision system. Uncalibrated methods also exist such as Kumar et al. [45], which employs SIFT

matching. Their method requires more offline initialization then calibrated camera do, and their system is more sensitive to sensors with different resolutions.

The servoing methods above give a good framework for building a PTZ method that can track an object throughout the camera's range. The methods however mention offline initialization which is not a viable option in a dynamic surveillance system as stationary objects such as parked vehicles can be removed or added to the scene during runtime which would produce false positives.

2.5. Open Questions

There has been a lot of work done in background subtraction and motion detection however not much has been related to PTZ surveillance. The majority of the work in the field of video tracking has been done using raw video. Uncompressed video is an unrealistic expectation when it comes to larger distributed PTZ surveillance systems as system bandwidth is limited to the network architecture. Therefore, issues have to be addressed before developing a system that uses these methods in a compressed PTZ surveillance system that is dynamic and does not need offline initialization or learning phases. Video compression leaves the original video with many compression artifacts that would cause false positives with algorithms designed around raw video. And of course, the higher the compression ratio, the more artifacts are present.

Current algorithms are limited to theoretical systems that operate on raw video in ideal network conditions. These algorithms fail to address the issues that arise in real time commercial systems. Commercial systems are often distributed and operate over less than ideal network conditions. For example, the San Franscisco Bay area video surveillance system that monitors traffic and roads (CALTRANS) still use aged ISDN technology [65]. A proposed solution to use an ideal situation algorithm would fail unless a complete system upgrade to a newer networking technology is implemented, as video compression ratio needs to be significantly increased to not exceed bandwidth limitations. This is an issue for many existing systems that currently have dated architectures in place. The cost of a complete system upgrade would far outweigh the

benefit of an autonomous tracking system. This thesis aims to find a solution for a robust real-time algorithm that can be deployed on existing large scale system architecture and provide benefit through reliable object detection and PTZ surveillance.

3. Proposed Algorithm

The goal of this research is to develop a suitable algorithm to track a target with a pan-tilt-zoom camera without user intervention. The input of the algorithm is compressed surveillance video. Compression is performed using either MPEG or MPEG4 standards (See Appendix A). The output of the algorithm is a rendered video with the tracking information overlaid to display the scene activity. The main assumptions are as follows:

1) The background is static.

2) One single target is present in the field of view.

These assumptions present well constrained problem for autonomous PTZ tracking. If multiple targets are in the field of view, then individual targets cannot be tracked by one PTZ camera unless they are moving consistently in the same direction.. In such a case, the PTZ camera can either track the first target detected, or allow an operator to input which target to track depending on configuration. The system diagram presented in Figure 3.1 shows how each module interacts with the others in order to achieve the tracking objective.

The diagram below describes the proposed approach. The process begins with the acquisition of a new frame, which is first decompressed. If the algorithm is currently not tracking an object, it performs the background subtraction and morphological cleaning in order to being the search for a moving object. On a clean difference frame it searches for a contour that could potentially represent the object of interest in the scene. If such a contour is detected, the proposed approach finds features on the contour that are robust to track. If no object contours are found, tracking is disabled and renders the original input frame.

If the algorithm is currently tracking an object, it inspects the scene window and searches for the features that were detected in the previous frame. If these features are found, the object's absolute position is computed and a command is sent to the PTZ camera to center the object. The algorithm then saves the window and features for the

next frame and overlays the object's information on the frame in order to send it to the renderer. If no features are found, it disables tracking and renders the incoming frame.

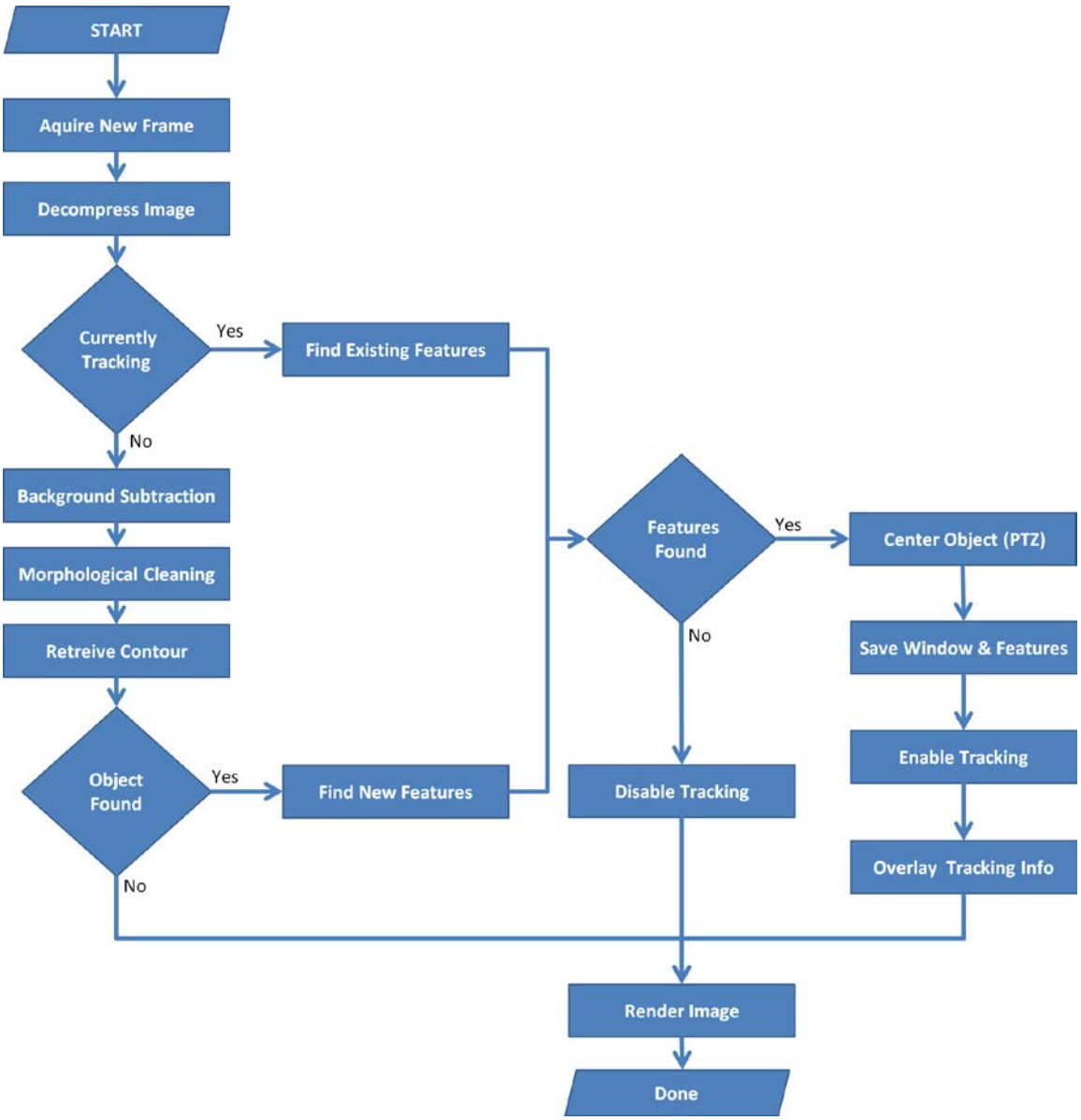

Figure 3.1 – Flow Chart of the proposed approach

This chapter discusses in detail each module of the proposed approach. Section 3.1 deals with the decompression of the input video. Section 3.2 describes background subtraction, while section 3.3 is dedicated to binary image analysis. Section 3.4 describes

the object detection. The remaining sections 3.5 and 3.6 discuss the tracking of the object throughout the scene and the interfacing module to the PTZ camera.

3.1. Decompression of Input Video

Video surveillance systems typically have multiple cameras distributed throughout various locations; some systems even span over an entire city or country. In order to design and implement a system that can be distributed and operated in real time, video compression must be used. Raw video would easily exceed bandwidth limitations of a system that requires digital media transmission. The proposed method receives the video from the incoming encoder and decompresses the stream. The two prevalent compressions algorithms in encoders used by surveillance systems are MJPEG and MPEG4.

Whether MJPEG or MPEG4, the stream needs to get decompressed in order to retrieve the image that corresponds to the frame. Decompressing a video compressed with MJPEG it is a simple process because MJPEG has no temporal compression and is simply a series of JPEG images.

MPEG4 performs temporal compression and uses three different frame types. An I-frame represents the complete compressed image. A, B, and P frames are partially compressed frames that express the differences between the data in the current frame and a subsequent I-frame. Therefore, an I-frame must arrive before beginning the decompression. The compression techniques are further discussed in Appendix A.

3.2. Background Subtraction

A complex algorithm is not necessary because background subtraction is only needed in the initial phase of the algorithm to isolate the foreground object. Background subtraction is not involved in tracking. Many background subtraction algorithms need a learning phase to discover which objects are foreground. Because of the spatial

variability of the considered environment, an involved learning process was not possible. For a combination of robustness and speed, background subtraction based on an averaging method is used. Figure 3.2 shows examples of typical results of this algorithm.

Figure 3.2 – Examples of results obtained with averaging method

3.2.1. Background Subtraction based on Averaging Method

This background subtraction method is adaptive and learns about the environment by learning the average and standard deviation of each pixel in the frame. This method uses all frames within a temporal window of given length to calculate the background. The first step is to calculate the running mean and the running variance as follows

Running Mean

$$\boldsymbol{avg_i(x,y) = (1-\alpha)\cdot avg_{i-1}(x,y) + \alpha \cdot I_i(x,y)} \quad (3.1)$$

Running Variance

$$\boldsymbol{\sigma^2 = \left(\frac{1}{N}\sum_{i=0}^{N-1} x_i^2\right) - \left(\frac{1}{N}\sum_{i=0}^{N-1} x_i\right)^2} \quad (3.2)$$

where *(x,y)* denotes the pixel location of current image I_i and α is a constant value set to 0.5 which allows us to weight the new input image to 50% against the existing running average avg_{i-1}. In eq. 3.3, *N* is defined as the number of samples and x_i is the current sample being processed. These values get calculated with each new frame processed within a given window. When the maximum number of variances is reached, the window shifts and earlier variance values are "forgotten". Using a threshold value *t*, the pixel's intensity *I* and the previous pixel's mean value $\bar{x}$, we can determine whether a pixel is foreground or background.

Background pixel	$\bar{x} - (\bar{x}t) < I(x,y) < \bar{x} + (\bar{x}t)$	(3.3)
Foreground pixel	*Otherwise*	

The threshold *t* used can be set up beforehand to fit the environment. If a value of a pixel falls between the thresholds it is considered background; otherwise it is classified as foreground. The same binary foreground mask image mentioned earlier is used to construct a foreground mask for every frame.

Figure 3.3 - Foreground of Image Extracted Showing Greyscale Data

3.3. Binary Image Analysis

Pixel-based morphological operations must be performed on the binary mask to find connected components and identify the foreground object. The initial foreground mask image is very noisy and all erroneous foreground pixels from the mask need to be removed.

In order to extract the object, all foreground connected pixels must be found. The idea is to group together foreground pixels that are separated by some background pixels due to noise, intensity fluctuations etc. This process will also allow for ignoring noisy foreground pixels generated in the decompression process.

Before trying to find the largest connect component (which is assumed to correspond to the object of interest), image clean-up is necessary. This is achieved by a morphological filter which involves *opening* followed by *closing*. A brief description of morphological operations, starting with the fundamental operators (dilation and erosion) follows below. Before discussing opening or closing, dilation and erosion must be discussed.

3.3.1. Binary Morphological Filtering

Dilation is a union neighbourhood operator. A binary structuring element S is scanned over a binary input image.

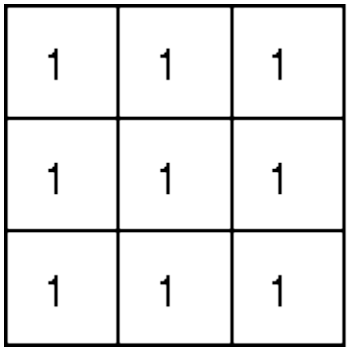

Figure 3.4 – Structuring Element S

Each time the origin of the structuring element S touches a binary 1-pixel, the entire translated structuring element shape is ORed with the input image, which has been initialized to all zeros. The definition is given below of the binary image B.

Dilation

$$B \oplus S = \bigcup_{b \in B} S_b \tag{3.4}$$

Erosion is dual to dilation. At each position where every 1-pixel of the structuring element covers a 1-pixel of the binary image, the binary image pixel corresponding to the origin of the structure element is ANDed with the input image. The erosion of a binary image B by the structuring element S is given below.

Erosion

$$B \ominus S = \{b | b + s \in B \forall s \in S\} \tag{3.5}$$

The process of *opening* occurs when erosion is performed before dilation. This process allows small clusters of foreground pixels to turn into background pixels. The reverse process, called *closing*, occurs when the dilation process is performed before erosion. This will result in a video frame where small clusters of background pixels

become foreground. Opening opens any dark intensity spaces in the image and closing helps to close the remaining gaps in the image.

The *opening* operation causes the small areas of noisy pixels to shrink to 0 and the *closing* operation rebuilds the areas of the image that remain.

Figure 3.5 - Morphology Example.
(Left) Original Image. (Center) Opening. (Right) Closing

3.4. Object Detection

3.4.1. Contour Extraction

Considering a cleaned up binary mask, the contours of the remaining regions are ready to be extracted. A contour can be defined as a list of connected pixels that represent a closed curve in the image. To find the contours in the binary mask, the border following algorithm proposed by Suzuki and Abe [51] is used because of its simplicity and high speed of detection. This algorithm determines the borders by evaluating the relationships of a binary pixel with its surrounding pixels using 1-pixel connected components. The output of the algorithm is a topological structural tree shown in figure 3.5 below.

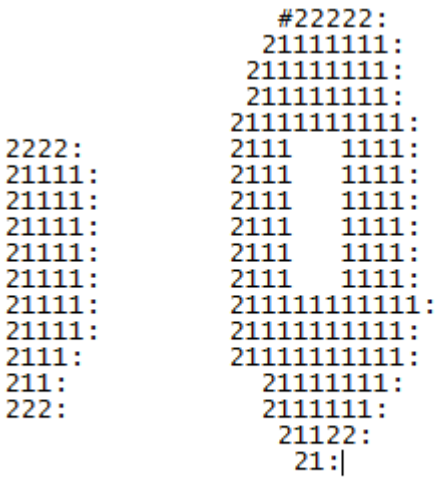

Figure 3.6 - The result of the contour detection algorithm.

Depending on the configuration specified by the user, the algorithm can determine the contour of the largest object, or of any object larger than a given size. The proposed approach locates the contour with the largest perimeter, which is tagged for further use. This contour is maintained in the binary image mask while all the other contours are set to 0.

3.4.2. Object Representation

Once the largest contour has been extracted, the binary mask corresponding to the region enclosed by this contour is applied to the original frame. Each frame is thus represented by a vector structure (R, G, B, A) where R is the red component, G is the green component, B is the blue component, and A is the foreground alpha mask which is the binary mask of the region enclosed by the contour.

3.5. Object Tracking

Object tracking is the process of following the moving target represented by the foreground object by controlling the PTZ camera via pan and tilt commands. Because of

occlusion and other moving objects in the input video, feature-based tracking was selected. Feature extraction allows for selecting multiple points on the target object to be tracked. Some points might get occluded during tracking, and the fact that multiple points are tracked confers robustness to the proposed approach.

Once the segmented object is extracted from the frame, the tracking process begins. This section discusses first how features are extracted from a given frame. Next, the optical flow feature-based computation is presented.

3.5.1. Feature Extraction

There are many different types of features that can be tracked. Obviously, a point inside a large uniform region is not a good choice since it would be nearly impossible to find the same point in a subsequent frame. If many points have identical or even very similar local neighborhoods, tracking those points in the next frame of the video would be nearly impossible. Thus, it is important to define what a 'good' feature is.

A 'good' feature is an image point that has strong spatial intensity changes in its neighbourhood, for example strong derivatives. But if that point is located on the edge of a homogeneous region, all points along that edge will look the same and therefore the frame to frame correspondence will not be uniquely determined. Thus, a 'good' feature is characterized by strong derivatives along two orthogonal directions. Points that have this property are called corners.

The most widely used mathematical definition of a corner is provided by Harris [63]. Corners are points in the image where the autocorrelation matrix of the second derivatives has two large eigenvalues. Second derivatives are useful because they don't respond to uniform gradient and they are also rotational invariant. This is important for tracking non-linear movement because features may shift or rotate throughout different frames.

Harris [63] showed that 'good' corners were found as long as the smaller of both eigenvalues was greater than a predefined minimum threshold.

For appropriate feature detection, the following steps are performed:

i) The image is blurred with a Gaussian filter. The size of this filter can be varied to detect features at different scales.

ii) Gradients are computed. The gradient is computed for each pixel for both x and y axes.

iii) A 2x2 gradient matrix for each pixel is calculated. The 2x2 gradient matrix is computed for each pixel by looking at an adjustable window around the pixel. The matrix, also known as the Hessian matrix, is a measure of the curvature of the intensity surface at the pixel. A larger window means that the curvature is measured over a larger region. The 2x2 matrix is as follows:

$$\begin{bmatrix} \frac{\partial^2 \mathrm{f}}{\partial \mathrm{x}^2} & \frac{\partial^2 \mathrm{f}}{\partial \mathrm{x}\,\partial \mathrm{y}} \\ \frac{\partial^2 \mathrm{f}}{\partial \mathrm{x}\,\partial \mathrm{y}} & \frac{\partial^2 \mathrm{f}}{\partial \mathrm{y}^2} \end{bmatrix} \tag{3.6}$$

iv) Eigenvalues for each 2x2 gradient matrix are found. As mentioned earlier, the 2x2 gradient matrix is a measure of curvature. The eigenvectors of this matrix represent the directions of minimum and maximum curvature. The eigenvalues represent the amount of curvature in those directions.

v) Only pixels that have the largest minimum eigenvalue are maintained as features. The minimum eigenvalue is the amount of curvature in the direction of minimum curvature. This is the value which is used to decide which features are "good." The features are ranked in decreasing order of their eigenvalues and the tracking process selects pixels based on a standard deviation of the points. If points fall out of a 20% range from the standard deviation, points are rejected.

Shi and Tomasi [19] provide a better way to find useful feature points within a video sequence compared to Harris [63]. Harris' original definition was the following:

$$R = Det(A) - kTr(A) \tag{3.7}$$

where R is the comparison value, $Det(A)$ is the determinant of the eigenvalue matrix A, $Tr(A)$ is the trace of the eigenvalue matrix with a weighting coefficient k. In

their implementation, *R* is positive for corners, negative for edges and small for flat regions.

Shi and Tomasi show that good corners resulted as long as the smaller of the two eigenvalues was greater than a single minimum threshold. This method proves to be computationally more effective than Harris' corner definition and is the method used in the proposed algorithm. Using the previously computed input mask of the largest foreground object, all corners in the foreground objects can be found. The image shown below has these corners represented with green dots. The red dots shown on the figure represent corners found that were below the absolute threshold.

Figure 3.7 - Example of result obtained with the Shi Tomasi Algorithm [19]: ***The green dots represent good features found, the red dots represent unusable features.***

3.5.2. Optical Flow Method

The tracking method used in the proposed approach is a sparse optical flow method, based on Lucas-Kanade [23]. The Lucas-Kanade method was originally designed to produce dense optical flow results. Yet, since the method can be applied to a subset of the points in the input image, it's been adapted to work on a subset of points to produce sparse results.

The Lucas-Kanade method can be considered as a pyramidal algorithm. The version allows for searching for a feature in a frame in a small window. If the feature is not found in the small window, the algorithm increases the window size to allow tracking of larger motion, where the corresponding displacements fall outside the initial window. A pyramidal approach solves this by tracking starting from a small search window in the image and working its way to a larger window in order to save search time if the feature has not moved far from its previous location. The optical flow method searches within its given window until a predefined minimal mean absolute difference is found between the point in a previous frame and a point in the current frame. If the mean absolute difference between the previous and current point is lower than our threshold, we track the point. If the difference falls above our threshold, or the maximum number of iterations have occurred, the point is considered lost.

Combining this method with Shi and Tomasi's corners allows for tracking 'good' features on an object in motion and for calculating the object's average motion vector.

The optical flow algorithm is described by the following:

$$\mathrm{I_x(p_n)V_x + I_y(p_n)V_y = -I_t(p_n)} \tag{3.8}$$

Where p_n is the feature inside the window and I_x, I_y, I_t are the partial derivatives of the image I with respect to *x*,*y* and time t evaluated at the given feature point p. Using this

formula, we can determine the x and y components, V_x and V_x respectively, for the motion vector V.

The equations described in 3.8 can be formed into a matrix $Av = b$ where

$$A = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \dots & \dots \\ I_x(p_n) & I_y(p_n) \end{bmatrix},$$

$$\text{v} = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \tag{3.9}$$

$$\text{b} = \begin{bmatrix} -I_t(p_1) \\ \dots \\ -I_t(p_n) \end{bmatrix}$$

To solve for the system, the algorithm sets up a least-squared minimization of the equations whereby $\|Av - b\|^2$ is solved in standard form as:

$$(A^T A)v = A^T b$$

The V_x and V_y motion components of motion vector are obtained using:

$$\begin{bmatrix} \sum \text{I}_\text{x}\text{I}_\text{x} & \sum \text{I}_\text{x}\text{I}_\text{y} \\ \sum \text{I}_\text{x}\text{I}_\text{y} & \sum \text{I}_\text{y}\text{I}_\text{y} \end{bmatrix} \begin{bmatrix} \text{V}_\text{x} \\ \text{V}_\text{y} \end{bmatrix} = - \begin{bmatrix} \sum \text{I}_\text{x}\text{I}_\text{t} \\ \sum \text{I}_\text{y}\text{I}_\text{t} \end{bmatrix} \tag{3.10}$$

Then the solution becomes

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = (A^T A)^{-1} A^T b \tag{3.11}$$

which can be solved when $(A^T A)$ is invertible, which occurs when it has two large eigenvectors. Two large eigenvectors indicate that the image region include texture running in at least two directions, which is previously defined as a Harris [63] 'good' feature.

3.5.3. Feature Tracking

In order to track the points from the optical flow, the camera's pan and tilt values must be calculated in order to center the object. The method applied is a general

technique for data analysis [7] that can be applied to the cluster of points to track an object in a video [50].

The mean-shift algorithm [7] described by Comaniciu is a method used to find local extreme points in a density distribution. Although this process is usually straightforward when the data is continuous, the process becomes complex when dealing with discrete data like in this case.

The algorithm is a procedure for locating the maxima of a density function given the sampled discrete data. The algorithm is iterative and begins with an initial estimate. The algorithm works by ignoring outliers in the points found in the previous step during the optical flow process, by processing only the points within a local window of the image data.

In order to begin the mean-shift tracking, it is necessary to assign a probability value *P* to every image pixel *I(x,y)*. With this probability assigned, a target model of the desired tracked object is created in the form a 1D histogram. The histogram is quantized into discrete bins. The probability values are determined by its grayscale intensity values which lie between 0 and 1. This allows the 1D histogram to be interpreted as a sort of look up table.

Once the probability distribution P(x,y) is computed, the maximum is searched. The location of the maximum is the peak and represents the position of the desired tracked object in the frame. A search window must then be determined. To determine the maximum search window, the 0^{th} and 1st order statistical moments are used.

M_{00} is the zero-order defined by:

$$M_{00} = \sum_{x} * \sum_{y} I(x, y) \tag{3.13}$$

the first -order moments are defined by:

$$\mathrm{M}_{10} = \sum_{\mathrm{x}} * \sum_{\mathrm{y}} \mathrm{xI(x, y)} \tag{3.14}$$

$$\mathrm{M}_{01} = \sum_{\mathrm{x}} * \sum_{\mathrm{y}} \mathrm{yI(x, y)}$$

The position of the target object C(x_c, y_c) is calculated as:

$$x_c = \frac{M_{10}}{M_{00}} \qquad (3.15)$$
$$y_c = \frac{M_{01}}{M_{00}}$$

The vector found will describe the object's movement and allow us to center the newly found center of mass. This process is repeated until the mean-shift vector converges to 0. At this point there is no more window centering movement possible and our peak point to the center of the object has been found. Once we have our motion vector, we can calculate the object's trajectory and move the camera to center the object.

3.6. PTZ Servoing

The PTZ servoing is the final step in the proposed approach. This step uses the center of mass of the desired object found by the mean shift tracking algorithm and controls the camera motion in order to align the optical center of the camera to the center of mass. The camera will continue panning or tilting until the object leaves the camera's field of motion at which point the camera stops in its current position and waits until a new foreground object enters the scene.

3.6.1. Camera PTZ Control

The proposed system interfaces with the PTZ camera through an IP interface so that Pan and Tilt commands can be sent. Absolute camera location can be determined from the desired object's center of mass and the motion vector calculated from the mean-shift

tracking algorithm. In order for the camera to smoothly track the object, we must estimate the next location of the object. The motion vector from the optical flow algorithm will help determine what location and at what speed to move the camera. Using the motion vector the target object's speed can be determined. With the speed determined we can estimate a movement point to center the object.

$$(x_c, y_c) = \frac{(x_c, y_c)_{\mathrm{n}-1} + (x_c, y_c)_{\mathrm{n}-2} + \cdots + (x_c, y_c)_{\mathrm{n}-\mathrm{fps}}}{\mathrm{fps}} \tag{3.16}$$

Where (x_c, y_c) is the current estimated object centroid. Using a moving average of the object's speed and direction, we can estimate how many pixels to pan or tilt ahead of the target so that tracking the target remains smooth and fluid while keeping it in view. The moving average is calculated over a 1 second window, if a video stream has 30 fps, it would average over the last 30 values. Once the camera has completed it pan or tilting the process of object tracking continues until the object leaves our camera's field of motion. At this point the tracking approach is reinitialized with the background subtraction algorithm. If we a camera's pan or tilt limitation, we continue to visually mean-shift track the target and render a bounding box around it until we can no longer see it.

4. System Integration

In order for this algorithm to be functional, implementation into a full C++ video decoding project was needed. The system uses a mechanism to pull both MJPEG and MPEG4 stream from various standard field encoders. The MJPEG video is pulled from an HTTP URI while the MPEG4 video uses RTSP to request the stream from the encoder. Once the stream is received, the packets are processed and passed onto a decoder to decode the stream. The decoded stream is then sent to the video player which handles the proposed algorithm and then passes the output video to the renderer to be displayed. All the acronyms and network related terms are explained in the glossary.

4.1. Getting the Stream

The first step in the system is requesting the stream from the unit. Once the stream is requested, the stream is received from the encoder to the processing system. The process of retrieving the stream is done differently depending on the stream type used.

4.1.1. MJPEG

MJPEG is the easiest stream type as far as session negotiation goes. Starting a stream from an MJPEG encoder is as simple as sending a HTTP_GET (as defined by RFC 2616). The stream will then begin to flow over a TCP connection through the HTTP port 80. The system listens over the socket connection and stores the incoming packets into a buffer ready to be processed.

4.1.2. MPEG4

Receiving an MPEG4 stream is more elaborate than MJPEG because it uses the RTSP client / server architecture to pull the video. RTSP (as defined by RFC 2326) is a client server protocol that provides a means to control the delivery of data with real-time properties. RTSP is not a connection that the client (video player) establishes and keeps with the server (video encoder). Instead RTSP is a way to maintain a session while receiving information about the encoder.

The first step of the RTSP process is the OPTIONS command. The client sends out this request to which the server will reply with all the RTSP commands that it supports. The next call requested by the client is the DESCRIBE command. The DESCRIBE command essentially asks the server to provide information about the stream available by the encoder at the requested URI. The server responds to this command with an SDP (as defined by RFC 3016). The SDP contains the encoder's media attributes which will specify which stream type to use and other information useful for the video decoding process. The next step is the SETUP command. This step is necessary so the client and server can negotiate a port range, transport type and transmission method over which the video packets will be sent. Finally the client will send a PLAY command to let the server know that it has agreed to the negotiation and is ready to receive the stream.

After the RTSP communication is over, RTP packets will begin to flow over the negotiated port where our system will begin to buffer the packets.

4.2. Processing the Incoming Stream

Once the stream is flowing in both cases, a packet queue is filled. A secondary thread is started after successful negotiation which loops through the packet queue and sends each packet to a frame assembler. The frame assembler reassembles the RTP or TCP packets into the full compressed video frame. The video frames are then added to a frame queue which then waits to be processed the video decoder.

The video decoding process is a complex operation and was out of the scope of this system to develop. A third party video decoder from Intel was used instead. Intel's IPP libraries offer video decoding for both MJPEG and MPEG4. A video decoding thread loops through the frame queue and passes the assembled frames through the IPP decoder, which returns a full uncompressed video frame. The uncompressed video frame is then added to a processing queue. Once a queue of uncompressed video is obtained, the analytical processing implemented by the proposed algorithm begins.

A video rendering thread runs and passes the uncompressed video frames through the proposed algorithm. In order to try and keep this system as close to real-time as possible (since it not possible to guarantee that a processing a frame through the proposed algorithm will finish before a new frame comes in) frames that are not ready to be processed are discarded. If more than one frame is available in the uncompressed video frame queue on the next loop, the oldest frame is discarded. The algorithm simply processes the most recent video frame available as input. Once a frame is processed with the proposed algorithm, it is placed in an outgoing video queue that awaits the renderer.

4.3. Rendering the Video

The video rendering is performed by DirectShow. DirectShow offers a Video Renderer module that can take uncompressed RGB32 and render them to the screen. The video renderer operates in a threaded model the same way as many other processes in the proposed system. A thread will take each new processed video from the outgoing video queue and renders them to the screen so the users can visualize and incorporate the results obtained by the proposed algorithm.

4.4. Camera Control

The final piece of the project needing integration is the motion control over our PTZ camera. In the current integration, an Axis Q6034E camera is used. Axis provides a C++ based API that can be integrated into the proposed algorithm to pan and tilt the

camera. The Axis API is IP based and all commands are sent over the network so there is no need for locality and the proposed algorithm can work on a field encoder with just a network connection.

Figure 4.8 - Axis Q6034E – PTZ camera used by the system

5. Experimental Results

5.1. Experimental Design

The proposed method has been implemented on an Intel Core i7 M640 in C++ using a custom library with OpenCV image processing functions. The custom library is responsible for the processing of the frames and PTZ servoing. Two separate Axis IP cameras were used for obtaining the experimental results. The first camera, Axis M1011W, is a stationary IP camera and was responsible for establishing the ground truth by viewing the entire field of view. The second camera, Axis Q6034E, is the PTZ camera that was used for the video acquisition and PTZ servoing. Table 1 shows the capabilities and specifications of both cameras. Experimental validation involved testing the complete computer vision system in online experiments. No public data set is available for testing the complete tracking system, because of its dynamic nature. The tracking algorithm has been tested over events such as the moving target entering or leaving the FOV of the PTZ camera and getting occluded by stationary objects in the scene. The stationary Axis camera (M1011W) is used to manually acquire the ground-truth for performance evaluation. The ground truth in this application is used to establish where the target is located with respect to the PTZ camera's FOV. Although the incoming live network feed is 30 fps, post processing time might require dropping intermediate frames if the frame arrives before post processing is complete. Therefore the actual processing video frame rate will be lower than the incoming video frame rate. The actual frame rate when the algorithm searches for motion is limited to roughly 25 fps, and when the algorithm tracks an object is lowered to an average of 15 fps.

Table 5.1

Camera	Max Resolution	Max Framerate	Pan Range	Max Pan Speed	Tilt Range	Max Tilt Speed
M1011W	640x480	30	N/A	N/A	N/A	N/A
Q6034E	1280x720	30	360° endless	450°/s	220°	450°/s

The testing scenarios involve a subject walking into the field of view of the PTZ camera and the camera tracking the subject until she reaches the other side of the room. The subject has no predetermined starting position and can start walking from any direction. The subject can change directions or change speed to test the robustness of the servoing algorithm. Direction can vary between constant or having 1 change in direction. Speed can vary between constant or having 2 changes in speed; speeding up and then slowing down. Lighting can be either optimal or dim.

In each experiment one single parameter is varied, while all other are kept constant. 2 different lighting modes, 3 different types of occlusion, 2 variations of speed and direction completed by each of the 3 test subjects renders a total of 72 tests.

5.2. Experimental Evaluation

To evaluate the algorithm, five metrics are used, as follows:

1. Precision (P) calculates the target localization accuracy. It is defined as:

$$P = \frac{TP}{TP + FP} \tag{5.1}$$

where TP and FP are true positive and false positives, respectively. TP is the number of frames in which the target is correctly localized by the camera. FP is the number of frames in which the algorithm incorrectly tracks the target so that it follows an incorrect path. Correctly localized is defined by a having a target with the mean of the features found being located on the target.

2. Detection precision (DP) calculates the target detection accuracy. It is defined as:

$$DP = \frac{CDF}{nF} \tag{5.2}$$

where CDF (Correctly Detected Frames) is the number of frames in which the target is detected correctly and nF is total number of frames that contain the target. We do not consider the false positive results as they are considered a failed detection, and count only the number of frames in which the system can detect the target correctly. DP is used for this purpose.

3. The Normalized Euclidean Distance of the target (d_{gc}) evaluates the dynamic performance of the tracking system. It is defined as:

$$d_{gc} = \frac{\sqrt{(x_c - {x_g}^2) + (y_c - {y_g}^2)}}{a} \tag{5.3}$$

where (x_g, y_g) is the ground-truth target coordinate, (x_c, y_c) is the center of the image and a is the radius of the circle that circumscribes the image (the maximum distance). It is the spatial latency of the tracking system, as, ideally, the target should be at the image center. This is calculated on each frame that contains the target.

4. Normalized Euclidean distance of the error (d_{gp}) shows the error of the tracking algorithm. It is the target position error, and is defined as:

$$d_{gp} = \frac{\sqrt{(x_p - {x_g}^2) + (y_p - {y_g}^2)}}{2a} \tag{5.4}$$

where (x_p, x_p) is the tracked object coordinate. Ideally, d_{gp} should be zero.

5. Track fragmentation (TF) indicates the lack of continuity of the tracking system for a single target track [47].

$$TF = \frac{T_{OUT}}{N_F} \tag{5.5}$$

T_{OUT} is the number of frames in which the target is outside the FOV and N_F is the total number of frames.

5.3. Varying Parameters

The tests are done with 3 different human subjects; A, B, and C in an indoor setting. The tests also manipulate different variables to access a potential point of weakness of the algorithm. The variables include lighting, occlusion, speed, direction and subject. By varying these 5 variables we can achieve 72 different tests to establish the robustness and points of future work of the algorithm. Table 2 shows the variable tests that were performed.

Table 5.2

Test	*Lighting*	*Occlusion*	*Speed*	*Direction*	*Subject*
1	Optimal	None	Fixed	Fixed	A
2	Optimal	None	Fixed	Fixed	B
3	Optimal	None	Fixed	Fixed	C
4	Dim	None	Fixed	Fixed	A
5	Dim	None	Fixed	Fixed	B
6	Dim	None	Fixed	Fixed	C
7	Optimal	Partial	Fixed	Fixed	A
8	Optimal	Partial	Fixed	Fixed	B
9	Optimal	Partial	Fixed	Fixed	C
10	Dim	Partial	Fixed	Fixed	A
11	Dim	Partial	Fixed	Fixed	B
12	Dim	Partial	Fixed	Fixed	C
13	Optimal	Full	Fixed	Fixed	A
14	Optimal	Full	Fixed	Fixed	B
15	Optimal	Full	Fixed	Fixed	C
16	Dim	Full	Fixed	Fixed	A
17	Dim	Full	Fixed	Fixed	B
18	Dim	Full	Fixed	Fixed	C

19	Optimal	None	Variable	Fixed	A
20	Optimal	None	Variable	Fixed	B
21	Optimal	None	Variable	Fixed	C
22	Dim	None	Variable	Fixed	A
23	Dim	None	Variable	Fixed	B
24	Dim	None	Variable	Fixed	C
25	Optimal	Partial	Variable	Fixed	A
26	Optimal	Partial	Variable	Fixed	B
27	Optimal	Partial	Variable	Fixed	C
28	Dim	Partial	Variable	Fixed	A
29	Dim	Partial	Variable	Fixed	B
30	Dim	Partial	Variable	Fixed	C
31	Optimal	Full	Variable	Fixed	A
32	Optimal	Full	Variable	Fixed	B
33	Optimal	Full	Variable	Fixed	C
34	Dim	Full	Variable	Fixed	A
35	Dim	Full	Variable	Fixed	B
36	Dim	Full	Variable	Fixed	C
37	Optimal	None	Fixed	Variable	A
38	Optimal	None	Fixed	Variable	B
39	Optimal	None	Fixed	Variable	C
40	Dim	None	Fixed	Variable	A
41	Dim	None	Fixed	Variable	B
42	Dim	None	Fixed	Variable	C
43	Optimal	Partial	Fixed	Variable	A
44	Optimal	Partial	Fixed	Variable	B
45	Optimal	Partial	Fixed	Variable	C
46	Dim	Partial	Fixed	Variable	A
47	Dim	Partial	Fixed	Variable	B
48	Dim	Partial	Fixed	Variable	C

49	Optimal	Full	Fixed	Variable	A
50	Optimal	Full	Fixed	Variable	B
51	Optimal	Full	Fixed	Variable	C
52	Dim	Full	Fixed	Variable	A
53	Dim	Full	Fixed	Variable	B
54	Dim	Full	Fixed	Variable	C
55	Optimal	None	Variable	Variable	A
56	Optimal	None	Variable	Variable	B
57	Optimal	None	Variable	Variable	C
58	Dim	None	Variable	Variable	A
59	Dim	None	Variable	Variable	B
60	Dim	None	Variable	Variable	C
61	Optimal	Partial	Variable	Variable	A
62	Optimal	Partial	Variable	Variable	B
63	Optimal	Partial	Variable	Variable	C
64	Dim	Partial	Variable	Variable	A
65	Dim	Partial	Variable	Variable	B
66	Dim	Partial	Variable	Variable	C
67	Optimal	Full	Variable	Variable	A
68	Optimal	Full	Variable	Variable	B
69	Optimal	Full	Variable	Variable	C
70	Dim	Full	Variable	Variable	A
71	Dim	Full	Variable	Variable	B
72	Dim	Full	Variable	Variable	C

The optimal lighting is artificial indoor lighting with no shadows. Dim lighting consists of 50% lighting strength. Fixed speed is normal walking pace throughout the FOV. Variable speed includes increasing and decreasing walking speed (The subject will increase their walking speed once and decrease their speed once during the crossing of

the field of view). Fixed direction is walking into the scene from left to right; or from right to left. In the variable direction case, the subject turns around and changes direction halfway through the FOV. In the case of partial occlusion, the subject's lower body is occluded while walking throughout the scene. In the case of full occlusion, the subject is completely occluded for 1 second and continues to walk out from behind the occlusion.

Table 2 (below) show the seven metric values that were calculated during the experiments. Unsurprisingly, the algorithm is shown to be performing best under ideal conditions. When occlusion is present, a lower target tracking precision (P) and more track fragmentation (TF) is achieved. We also observe a decline in localization of the target (μd_{gc} and μd_{gp}) when a partial or full occlusion is present. This indicates that the target is farther from the ground-truth after it passes through an occlusion.

Table 5.3

Test	P (%)	DP (%)	TF (%)	μd_{gc}	$\sigma^2 d_{gc}$	μd_{gp}	$\sigma^2 d_{gp}$
1	92.81	94.24	0.46	0.10328	0.003133	0.062754	0.001389
2	94.13	95.17	1.12	0.10121	0.002911	0.051677	0.001511
3	93.71	95.33	1.04	0.10517	0.003551	0.058112	0.001478
4	89.55	90.67	1.53	0.10732	0.007853	0.063178	0.004045
5	88.10	89.26	2.31	0.10702	0.007831	0.063002	0.004034
6	87.09	88.75	1.36	0.10643	0.007788	0.062654	0.004012
7	80.00	81.12	3.10	0.140777	0.012023	0.077083	0.005912
8	80.44	82.11	3.20	0.168626	0.012339	0.099268	0.006356
9	81.44	83.60	2.94	0.160005	0.011708	0.094193	0.006031
10	79.69	80.96	3.97	0.175092	0.012812	0.103075	0.0066
11	80.10	81.57	3.22	0.171557	0.012553	0.100994	0.006467
12	79.41	80.90	4.01	0.177505	0.012989	0.104496	0.006691
13	69.10	70.87	7.21	0.266387	0.019493	0.156819	0.010041
14	68.79	70.73	8.99	0.26906	0.019688	0.158393	0.010142
15	68.99	69.72	8.64	0.267336	0.019562	0.157378	0.010077
16	69.11	70.39	9.64	0.266301	0.019486	0.156769	0.010038

17	67.89	69.04	8.97	0.276819	0.020256	0.16296	0.010434
18	68.14	69.75	7.96	0.274664	0.020098	0.161692	0.010353
19	76.00	77.19	6.79	0.214203	0.015674	0.126099	0.008074
20	77.10	79.29	6.11	0.19742	0.014446	0.116219	0.007441
21	76.44	77.82	6.70	0.20311	0.014862	0.119568	0.007656
22	76.01	77.38	5.99	0.206817	0.015134	0.121751	0.007796
23	75.94	77.01	5.81	0.20742	0.015178	0.122106	0.007818
24	75.44	76.34	5.14	0.211731	0.015493	0.124644	0.007981
25	74.11	75.94	7.31	0.223196	0.016332	0.131393	0.008413
26	73.96	76.03	7.22	0.22449	0.016427	0.132155	0.008462
27	73.09	74.10	6.91	0.23199	0.016976	0.13657	0.008744
28	73.46	74.38	6.97	0.2288	0.016742	0.134692	0.008624
29	72.99	74.19	7.41	0.232852	0.017039	0.137077	0.008777
30	73.00	75.19	7.91	0.232766	0.017032	0.137027	0.008774
31	69.14	69.80	8.44	0.266043	0.019467	0.156616	0.010028
32	69.22	71.09	8.32	0.265353	0.019417	0.15621	0.010002
33	70.20	71.48	9.24	0.256904	0.018799	0.151237	0.009684
34	67.49	68.09	9.89	0.280267	0.020508	0.16499	0.010564
35	68.32	70.27	8.34	0.273112	0.019985	0.160778	0.010294
36	68.82	70.32	7.98	0.268801	0.019669	0.15824	0.010132
37	82.00	83.85	3.98	0.124784	0.017504	0.082385	0.004425
38	81.44	82.07	4.21	0.160005	0.011708	0.094193	0.006031
39	83.95	85.65	4.04	0.138366	0.010125	0.081455	0.005215
40	80.66	82.95	4.00	0.166729	0.0122	0.098152	0.006285
41	81.27	82.98	3.77	0.16147	0.011815	0.095056	0.006086
42	81.00	82.57	3.98	0.163798	0.011986	0.096426	0.006174
43	76.54	77.88	4.78	0.202248	0.014799	0.119061	0.007623
44	75.46	76.00	5.01	0.211558	0.01548	0.124542	0.007974
45	76.12	78.33	5.98	0.205868	0.015064	0.121192	0.00776
46	75.23	76.23	7.65	0.213541	0.015626	0.125709	0.008049

47	75.10	76.89	6.98	0.214662	0.015708	0.126369	0.008091
48	74.97	76.36	7.04	0.215782	0.01579	0.127029	0.008134
49	69.43	71.20	9.71	0.263543	0.019284	0.155145	0.009934
50	68.23	70.11	8.94	0.273888	0.020041	0.161235	0.010324
51	70.18	72.43	9.49	0.257077	0.018811	0.151338	0.00969
52	67.97	69.95	10.67	0.276129	0.020205	0.162554	0.010408
53	68.54	70.79	9.11	0.271215	0.019846	0.159661	0.010223
54	68.00	70.02	9.67	0.275871	0.020186	0.162402	0.010398
55	76.94	78.45	7.80	0.198799	0.014547	0.117031	0.007493
56	76.99	78.73	7.66	0.198368	0.014515	0.116777	0.007477
57	75.11	76.78	8.79	0.214576	0.015701	0.126318	0.008088
58	73.46	74.06	11.61	0.2288	0.016742	0.134692	0.008624
59	74.00	74.79	8.67	0.224145	0.016401	0.131952	0.008449
60	75.11	77.30	8.99	0.214576	0.015701	0.126318	0.008088
61	73.11	73.74	12.69	0.231817	0.016963	0.136468	0.008738
62	69.94	70.64	12.01	0.259146	0.018963	0.152556	0.009768
63	72.33	73.30	11.00	0.238542	0.017455	0.140427	0.008991
64	72.49	74.76	11.48	0.237162	0.017354	0.139615	0.008939
65	73.00	74.11	11.98	0.232766	0.017032	0.137027	0.008774
66	72.41	74.30	12.00	0.237852	0.017404	0.140021	0.008965
67	69.00	71.00	14.78	0.296944	0.021728	0.174808	0.011193
68	68.44	70.91	12.00	0.302308	0.022121	0.177966	0.011395
69	68.00	70.00	11.13	0.3862	0.02826	0.227352	0.014557
70	67.00	69.02	16.00	0.316102	0.02313	0.186086	0.011915
71	66.10	67.94	16.00	0.324723	0.023761	0.191161	0.01224
72	65.00	67.00	17.00	0.3745	0.027404	0.220464	0.014116

Table 5.3. Test Number, P precision, μd_{gc} mean of d_{gc}, μd_{gp} mean of d_{gp}, $\sigma^2 d_{gc}$ variance of d_{gc}, $\sigma^2 d_{gp}$ variance of d_{gp} for our tracking algorithm.

The graphs in Fig. 5.8-5.11 show the d_{gc} and d_{gp} values for 4 separate experiments. These distances allow for verifying whether or not the target is near the ground-truth and whether or not the tracking has lost the target. If d_{gc} is greater than 1, then the target is lost (it's completely outside of the FOV), distances shorter than 0.6 are considered to be near the field of view but still a distance from the centroid.

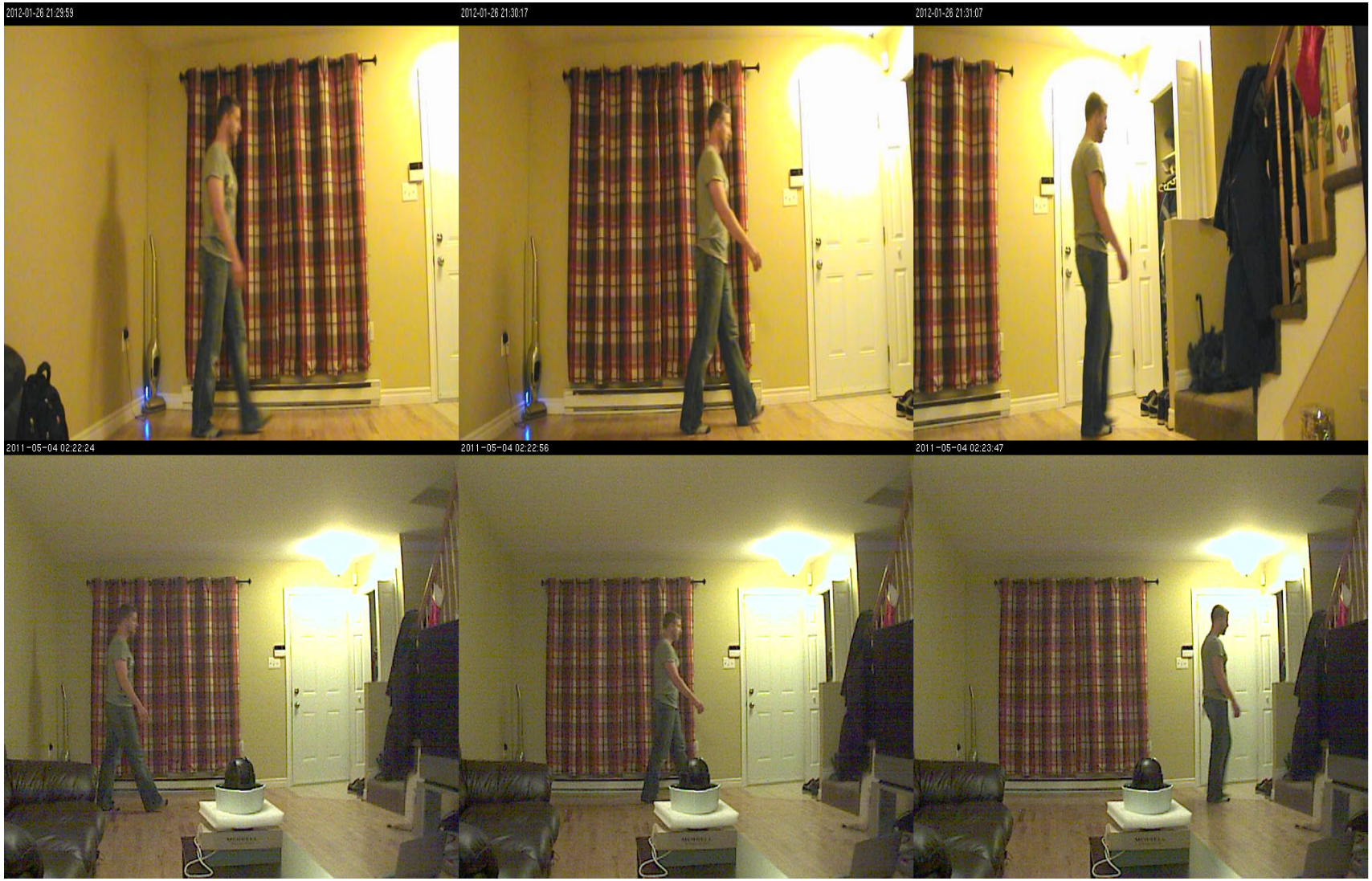

Figure 5.9 - Target being tracked by PTZ Camera

Test 1 shows the normalized distances under ideal tracking conditions. One may see here that the tracking centroid is always very near the object centroid and only reaches a maximum error distance of 0.25 when the camera is panning. The 'Test 19' shows the normalized distances under speed variation. This test shows two increases in the distance from the target. This happens the first time when the target increases their speed and a second time when they slow down. The tracker simply adjusts its panning speed in these cases and can continue to track the target reliably. Test 37 shows the normalized distances under direction variation. Again, in this case one may see a spike in distance from the target. This happens when the target changes direction and the camera tracking algorithm must adjust its pan speed in a negative direction to follow the target. And the final graph, Test 7 shows the normalized distances when the object is partially occluded. Here, a very large spike is present when the object is partially occluded. This happens because a certain percentage of the features that were being tracked will be misdetected on the object that is causing the occlusion. As the target continues to walk through past the occlusion the feature of points cloud grows and the object's centroid is disturbed. At this point the detection probability of the features points that were left behind from the misdetection will begin to decrease. Once the probability of these feature points reach the threshold for non-matching points, they are dropped and the tracking continues properly. This is why one sees such a fast drop after the error distance ramps upwards.

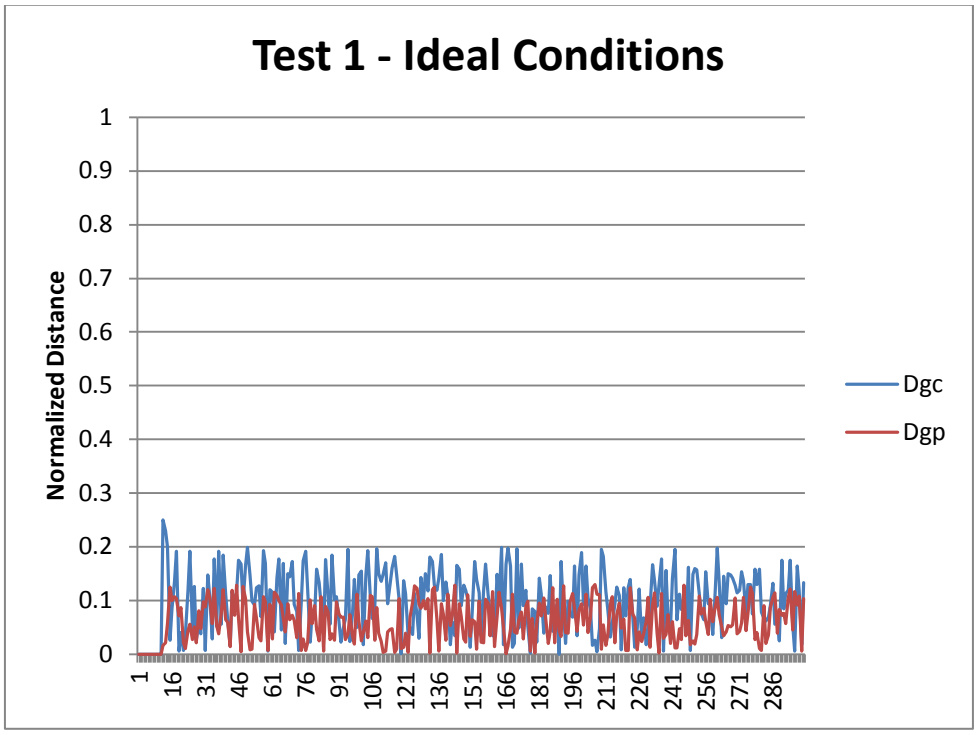

Figure 5.10 - Test 1 - Ideal Conditions

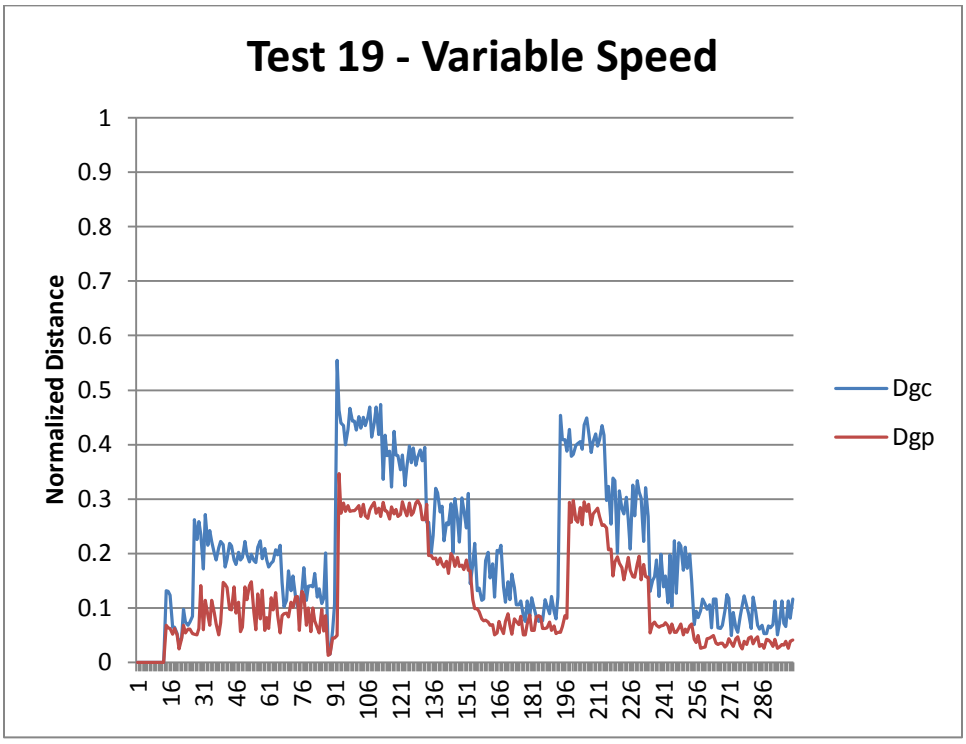

Figure 5.11 - Test 19 - Variable Speed

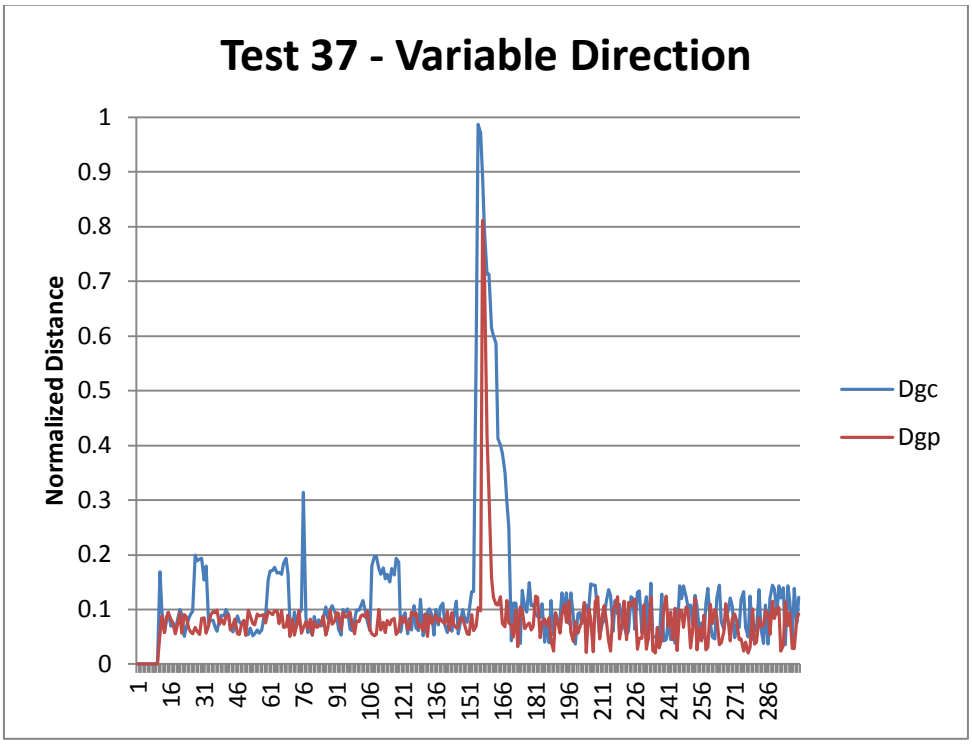

Figure 5.12 - Test 37 - Variable Direction

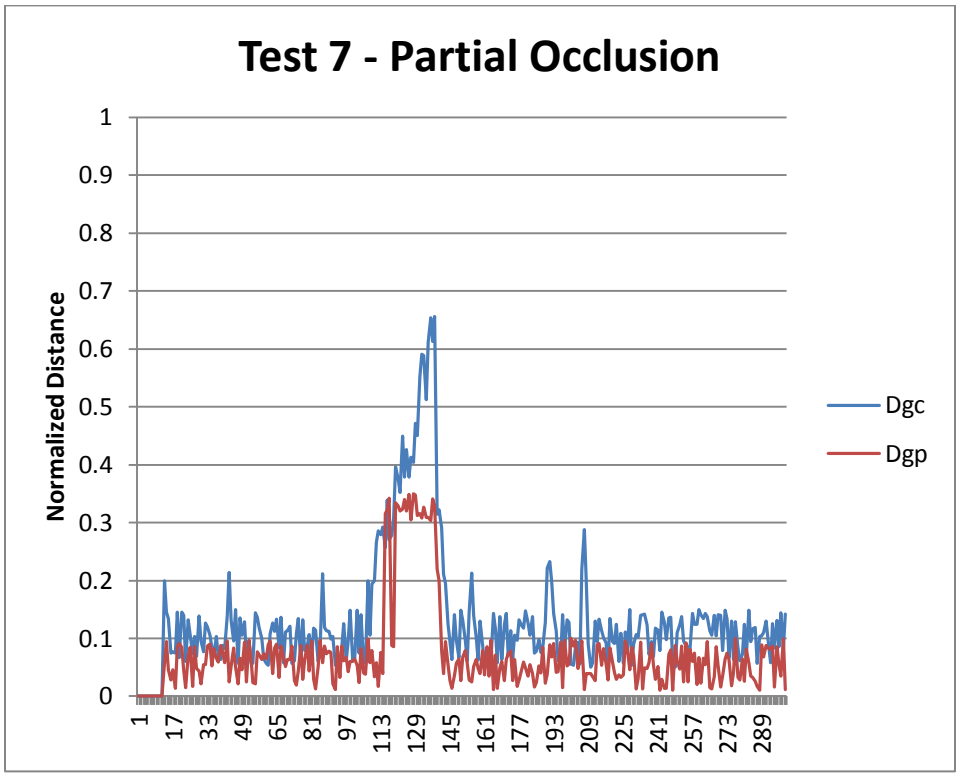

Figure 5.13 - Test 7 - Partial Occlusion

Results show that the algorithm can handle and overcome variations such as direction, speed, lighting and occlusion because of the feature detection and tracking mechanisms in place. It only loses a target if the target is fully occluded by an object; in this case, the feature tracking algorithm detects the high error distance between the points and stops the tracking. When this occurs, the background subtraction algorithm restarts and tracking can restart on the same target. The tracking algorithm will also have a lag on the target if the target suddenly changes direction or speed but the algorithm will recover by increasing or decreasing tracking speed. By using the feature detection and variable speed tracking the algorithm can handle random motion between captured frames, as long as the target's appearance does not change significantly.

5.4. Performance Results

Codec	**Resolution**	**Frame Rate Input**	**CPU Usage**	**Memory (MB) Usage**	**FPS Output (Motion Detection)**	**FPS Output (Tracking)**
MJPEG	640x480	30	3.2%	59.1	26	17
MJPEG	800x600	30	4.3%	61.2	25	16
MJPEG	1280x720	30	5.4%	65.1	23	15
MPEG 4	640x480	30	3.1%	58.1	27	17
MPEG 4	800x600	30	3.9%	60.1	24	16
MPEG 4	1280x720	30	4.9%	64.4	22	14

The table above depicts the performance results of the algorithm. The tests were executed using 2 different encoding types at 3 different resolutions. The performance tests were conducted on a Dell Latitude E6410 laptop. The system has an Intel Core i7 M640 (2.8 GHz) processor, 4096 MB of RAM running Windows 7 64-bit.

6. Conclusion

6.1. Summary

In this study, we have proposed an integrated method for network-based PTZ tracking of a target throughout an indoor scene. Building upon previous research in the PTZ servoing field, we apply mean shift tracking to network-based IP Cameras. This model is therefore useful for real-time tracking in surveillance application with remote network cameras.

The proposed motion tracking algorithm detects motion using a background subtraction algorithm and contour detection. The target is followed by mean-shift tracking a set of corner features detected on the object contour. The corners are then grouped into a target and tracked through the scene using a variable speed servoing.

The proposed network-based method has several advantages for surveillance applications. This lightweight algorithm allows for real-time tracking of a target in a client-server surveillance system. The algorithm can be run from any client with the ability to PTZ the IP Camera. The proposed approach combines detection and matching and estimates the PTZ speed and motion position. Results show that our algorithm can handle and overcome a large change in motion between pairs of consecutive frames. A target loss can occur if the target is fully occluded or suddenly changes its direction of motion by walking fast in the opposite direction of the predicted PTZ direction. However, the proposed algorithm enables tracking to resume if the target re-enters the stalled field of view of the PTZ camera.

The algorithm was also found to be computationally efficient. The tracking algorithm reduces the fps rate of an incoming stream to roughly half that of the original. The results indicate a good performance and would not hinder the video speed unusable. This shows that this system would be acceptable for live surveillance system usage.

6.2. Future Work

There are many options to improve on this algorithm in future work. One direction will be about enhancing the robustness of the motion prediction to ensure that the target doesn't get lost before exiting the field of view. Expanding the definition of targets to also include non-human moving objects would also be useful for tracking vehicles in outdoor surveillance systems.

Further expansion would be human target detection that is capable of identifying a certain person. The system would create a feature based model of an object and match any new object entering into the field of view to a model. This feature would be useful for surveillance scenarios where only authorized personnel are allowed. This method would allow for the system to recognize when a target has been occluded and continue to track the same object model.

Another future goal would be to relax the constraints of the system. Our current single target, static indoor background model could be expanded to track multiple targets in an outdoor dynamic environment. The addition of large FOV static cameras to trigger more specific PTZ camera would also be beneficial to large scale event surveillance. The static camera could identify a single person walking through a crowded area and alert a nearby PTZ camera to closely track the target.

Bibliography

[1] Wren, C.R.; Azarbayejani, A.; Darrell, T.; Pentland, A.P.; "Pfinder: real-time tracking of the human body," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.19, no.7, pp.780-785, Jul 1997

[2] Lo, B.; Velastin, S.A.; Vicencio-Silva, M.A.; Jie Sun; , "An intelligent distriblited surveillance system for public transport," Intelligence Distributed Surveillance Systems, IEE Symposium on (Ref. No. 2003/10062) , vol., no., pp. 10/1- 10/5, 26 Feb. 2003

[3] Cucchiara, R.; Grana, C.; Prati, A.; Vezzani, R.; , "Probabilistic posture classification for Human-behavior analysis," Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on , vol.35, no.1, pp. 42- 54, Jan. 2005

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

[5] P. W. Power, J. A. Schoonees. "Understanding background mixture models for foreground." In Proceedings Image and Vision Computing New Zealand (2002)

[6] Elgammal, A.; Duraiswami, R.; Davis, L.S.; , "Probabilistic tracking in joint feature-spatial spaces," Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on , vol.1, no., pp. I-781- I-788 vol.1, 18-20 June 2003

[7] Comaniciu, D.; Meer, P.; , "Mean shift analysis and applications," Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on , vol.2, no., pp.1197-1203 vol.2, 1999

[8] Comaniciu, D.; , "Robust information fusion using variable-bandwidth density estimation," Information Fusion, 2003. Proceedings of the Sixth International Conference of , vol.2, no., pp. 1303- 1309, 2003

[9] Piccardi, M.; Jan, T.; , "Mean-shift background image modelling," Image Processing, 2004. ICIP '04. 2004 International Conference on , vol.5, no., pp. 3399- 3402 Vol. 5, 24-27 Oct. 2004

[10] B. Han, D. Comaniciu, Y. Zhu, and L. S. Davis, "Sequential kernel density approximation and its application to real-time visual tracking," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 7, pp. 1186–1197, 2008.

[11] Oliver, N.M.; Rosario, B.; Pentland, A.P.; , "A Bayesian computer vision system for modeling human interactions ," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.22, no.8, pp.831-843, Aug 2000

[12] Connor, D.; Limb, J.; , "Properties of Frame-Difference Signals Generated by Moving Images," Communications, IEEE Transactions on , vol.22, no.10, pp. 1564- 1575, Oct 1974

[13] Cafforio, C.; Rocca, F.; , "Methods for measuring small displacements of television images," Information Theory, IEEE Transactions on , vol.22, no.5, pp. 573- 579, Sep 1976

[14] T. W. Ryan, R. T. Gray, and B. R. Hunt. "Prediction of correlation errors in stereo-pair images". Optical Engineering, 19(3):312–322, May/June 1980.

[15] G. A. Wood. "Realities of automatic correlation problems." Photogrammetric Engineering and Remote Sensing, 49(4):537–538, April 1983

[16] Q. Tian and M. N. Huhns. "Algorithms for subpixel registration." Computer Vision, Graphics, and Image Processing, 35:220–233, 1986.

[17] P. J. Burt, C. Yen, and X. Xu. "Local correlation measures for motion analysis: a comparative study." In IEEE Conference on Pattern Recognition and Image Processing (PRIP'82), pages 269–274, IEEE Computer Society Press, 1982.

[18] P. Anandan. "A computational framework and an algorithm for the measurement of visual motion." International Journal of Computer Vision, 2(3):283–310, January 1989.

[19] Jianbo Shi; Tomasi, C.; , "Good features to track," Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on , vol., no., pp.593-600, 21-23 Jun 1994

[20] D. I. Barnea and H. F. Silverman "A class of algorithm for fast digital image registration", IEEE Trans. Comput., vol. C-21, pp.179 1972

[21] J. Berclaz, F. Fleuret, and P. Fua. Robust People Tracking with Global Trajectory Optimization. In Conference on Computer Vision and Pattern Recognition, New York, 2006. 3

[22] H.P. Moravec, "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," PhD thesis, Stanford Univ., Sept. 1980. *(published as Robot Rover Visual Navigation. Ann Arbor, MI: UMI Research Press, 1981.)*

[23] B.D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," Proceedings of the 1981 DARPA Imaging Understanding Workshop (pp. 121-130), 1981.

[24] Y. Aloimonos. Active Perception. Hillsdale, NJ: Lawrence Erlbaum, Associates, 1993.

[25] Burger, W.; Bhanu, B.; , "Estimating 3D egomotion from perspective image sequence," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.12, no.11, pp.1040-1058, Nov 1990

[26] Murray, D.; Basu, A.; , "Motion tracking with an active camera," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.16, no.5, pp.449-459, May 1994

[27] Irani, M.; Anandan, P.; , "A unified approach to moving object detection in 2D and 3D scenes ," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.20, no.6, pp.577-589, Jun 1998

[28] Araki, S.; Matsuoka, T.; Takemura, H.; Yokoya, N.; , "Real-time tracking of multiple moving objects in moving camera image sequences using robust statistics," Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on , vol.2, no., pp.1433-1435 vol.2, 16-20 Aug 1998

[29] Guo, G.; Dyer, C.R.; Zhang, Z.; , "Linear combination representation for outlier detection in motion tracking," Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on , vol.2, no., pp. 274- 281 vol. 2, 20-25 June 2005

[30] S. Ullman and R. Basri, "Recognition by linear combination of models," IEEE Trans. Pattern Analysis and Machine Intell. 13(10), 992-1006, 1991.

[31] G. L. Foresti and C. Micheloni, “A robust feature tracker for active surveillance of outdoor scenes,” Electron. Lett. Comput. Vision Image Anal., vol. 1, no. 1, pp. 21–36, 2003.

[32] C. Micheloni and G. L. Foresti, “Real time image processing for active monitoring of wide areas,” J. Vision Commun. Image Represent., vol. 17, no. 3, pp. 589–604, 2006

[33] Tordoff, B.; Murray, D.; , "Reactive control of zoom while fixating using perspective and affine cameras," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.26, no.1, pp.98-112, Jan. 2004

[34] Fleuret, F.; Berclaz, J.; Lengagne, R.; Fua, P.; , "Multicamera People Tracking with a Probabilistic Occupancy Map," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.30, no.2, pp.267-282, Feb. 2008

[35] Hampapur, A.; Brown, L.; Connell, J.; Ekin, A.; Haas, N.; Lu, M.; Merkl, H.; Pankanti, S.; , "Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking," Signal Processing Magazine, IEEE , vol.22, no.2, pp. 38- 51, March 2005

[36] Scotti, G.; Marcenaro, L.; Coelho, C.; Selvaggi, F.; Regazzoni, C.S.; , "Dual camera intelligent sensor for high definition 360 degrees surveillance," Vision, Image and Signal Processing, IEE Proceedings - , vol.152, no.2, pp. 250- 257, 8 April 2005

[37] Chung-Hao Chen; Yi Yao; Page, D.; Abidi, B.; Koschan, A.; Abidi, M.; , "Heterogeneous Fusion of Omnidirectional and PTZ Cameras for Multiple Object Tracking," Circuits and Systems for Video Technology, IEEE Transactions on , vol.18, no.8, pp.1052-1063, Aug. 2008

[38] Olfati-Saber, R.; Fax, J.A.; Murray, R.M.; , "Consensus and Cooperation in Networked Multi-Agent Systems," Proceedings of the IEEE , vol.95, no.1, pp.215-233, Jan. 2007

[39] Soto, C.; Bi Song; Roy-Chowdhury, A.K.; , "Distributed multi-target tracking in a self-configuring camera network," Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on , vol., no., pp.1486-1493, 20-25 June 2009

[40] J. Park, P. C. Bhat, and A. C. Kak, "A Look-up Table Based Approach for Solving the Camera Selection Problem in Large Camera Networks," IEEE Workshop on DSC, 2006.

[41]D. G. Lowe; , “Distinctive Image Features from Scale-Invariant Keypoints,” Internation Journal of Computer Vision, vol. 60, no.2, pp. 99-100, 2004

[42] Meltzer, J.; Soatto, S.; , "Edge descriptors for robust wide-baseline correspondence," Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on , vol., no., pp.1-8, 23-28 June 2008

[43] Dingrui Wan; Jie Zhou; , "Multiresolution and Wide-Scope Depth Estimation Using a Dual-PTZ-Camera System," Image Processing, IEEE Transactions on , vol.18, no.3, pp.677-682, March 2009

[44] J.W. Hart, B. Scassellati, and S.W. Zucker, "Epipolar Geometry for Humanoid Robotic Heads", in Proc. ICVW, 2008, pp.24-36.

[45] S. Kumar, C. Micheloni, and G.L. Foresti,; “Stereo Vision in Cooperative Cameras Network,” in Smart Cameras, N.Belbachir, Ed., 2009

[46] R. Spence, Information Visualization: Design for Interaction, Perason/Prentice Hall, USA, 2007

[47] K. Cook, J. Thomas, “Illuminating the Path: Creating the R&D Agenda for Visual Analytics”, National Visualization and Analytics Center, USA, 2005, pp. 3-33.

[48]D. Rueckert, A. Frangi, and J. Schnbel, “Automatic Construction of 3D Statistical Deformation Models using Nonrigid Registration”, in Proceedings of Medical Image Computing and Computer-Assisted Intervention (MICCAI’01), Vol. 2208, Utrecht, The Netherlands, Oct. 2001, pp. 77-94.

[49] A. Bovic, Handbook of Image and Video Processing, Elsevier Academic Press, Burlington, MA, USA, 2005.

[50] Bradski, G.R.; , "Real time face and object tracking as a component of a perceptual user interface," Applications of Computer Vision, 1998. WACV '98. Proceedings., Fourth IEEE Workshop on , vol., no., pp.214-219, 19-21 Oct 1998

[51] Suzuki, S, and K Be. “Topological structural analysis of digitized binary images by border following.” Computer Vision Graphics and Image Processing 30.1 (1985) : 32-46.

[52] "Proceedings 1998 IEEE Workshop on Visual Surveillance," Visual Surveillance, 1998. Proceedings., 1998 IEEE Workshop on , vol., no., pp.vii+94, 2 Jan 1998

[53] "Proceedings Second IEEE Workshop on Visual Surveillance (VS'99) (Cat. No.98-89223)," Visual Surveillance, 1999. Second IEEE Workshop on, (VS'99) , vol., no., pp.vi+91, Jul 1999

[54] "Proceedings Third IEEE International Workshop on Visual Surveillance," Visual Surveillance, 2000. Proceedings. Third IEEE International Workshop on , vol., no., pp.vi+85, 2000

[55] "First IEE Intelligent Distributed Surveillance System," Intelligent Distributed Surveillance Systems, IEE , vol., no., pp. , 2003

[56] "Second IEE Intelligent Distributed Surveillance System," Intelligent Distributed Surveillance Systems, IEE , vol., no., pp. 2004

[57] Special issue on visual surveillance, International Journal of Computer Vision, 2000

[58] Special issue on visual surveillance, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000

[59] Special issue on third generation surveillance systems, Proc. IEEE, 2001

[60] Special issue on human motion analysis, Computer Vision Image Understanding ,2001

[61] Nwagboso, C.: 'User focused surveillance systems integration for intelligent transport systems', in Regazzoni, C.S., Fabri, G., and Vernazza, G. (Eds.): 'Advanced Video-based Surveillance Systems' (Kluwer Academic Publishers, Boston, 1998), Chapter 1.1, pp. 8–12

[62] CBSNewsOnline. "Fighting Terrorism in New York City." YouTube. 25 Sept. 2011. Web. 25 Sep. 2011. <http://www.youtube.com/watch?v=Nf_PzCfpPug>.

[63] C. Harris and M. Stephens, "A combined corner and edge detector," Proceedings of the 4th Alvey Vision Conference (pp. 147-151), 1988

[64] Räty, T.D.; , "Survey on Contemporary Remote Surveillance Systems for Public Safety," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on , vol.40, no.5, pp.493-515, Sept. 2010

[65] D. Galarus, "TMC-TMS Communications: Overview and Demonstration", NRITS Conference, Anchorage, Alaska, September 2008

Appendix

Appendix A - Video Input

MJPEG

Unlike other video compression, MJPEG is not defined over a time domain; simply a spatial domain. MJPEG is simply a temporally ordered sequence of JPEG images, there is no temporal compression achieved through this method. JPEG is a lossy compression algorithm and uses Discrete Cosine Transformation to achieve compression. Although it is a lossy compression; it is able to lose information and still allow the image to appear unaltered. This is due to the fact that human vision is not very cable of viewing high-spatial frequency components; therefore the frequency can be lowered without the human perception being changed.

In order to perform the DC transformation on the image, each image is divided into 8 x 8 blocks. A two dimensional DCT is then applied to each block. Once our DCT coefficients from the transformation have been found the image can be quantized. The goal of quantization is to reduce the total number of bits needed to represent the image. This step is accomplished by dividing each entry of the frequency space block calculated by the DCT by an integer from the quantization matrix.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(a)

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

(b)

Figure A.14 - JPEG Quantization Matrix

The above table shows the default JPEG 8 x 8 quantization matrix ((a) is luminance component and (b) is chrominance). Since the dividing numbers in the quantization matrix are quite large, this is the step that causes the most loss in JPEG compression.

The final stage of JPEG compression is entropy coding. This last step is broken up into two parts; both of which are lossless. The two steps consist of encoding both AC and DC coefficients from the DCT. The AC coefficients are encoded with Run-Length Coding (RLC). A zigzag scan is used to encode the long run of zeros in the 8x8 matrix into vectors.

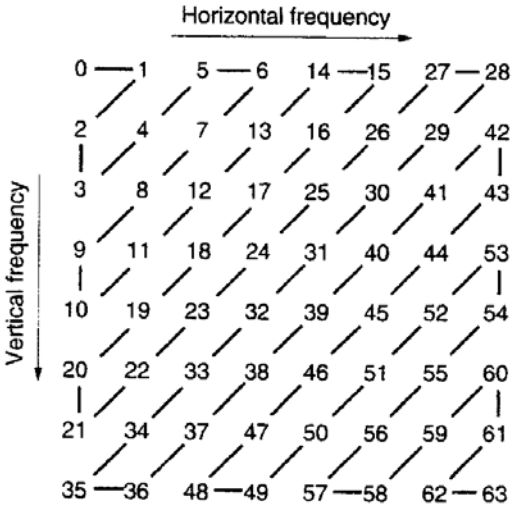

Figure A.15 - Zigzag Scan

The DC coefficients are encoded using Differential Pulse Code Modulation (DPCM). Once the both types of coefficients go through that encoding process, a final encoding process is performed over both. The method used to encode both of these is a variant of Huffman coding.

MPEG4

The biggest advantage of MPEG4 is its temporal compression. This allows for high quality video to be sent over a network at a much smaller bitrate then MJPEG. MPEG-4 has a new method of coding; an object based method. In MPEG-4 the breakdown of video object is very object-oriented. Each video-object sessions (VS) have one or more video objects (VOs) which have one or more video object layers (VOLs). This object oriented methodology can be broken down into five levels.

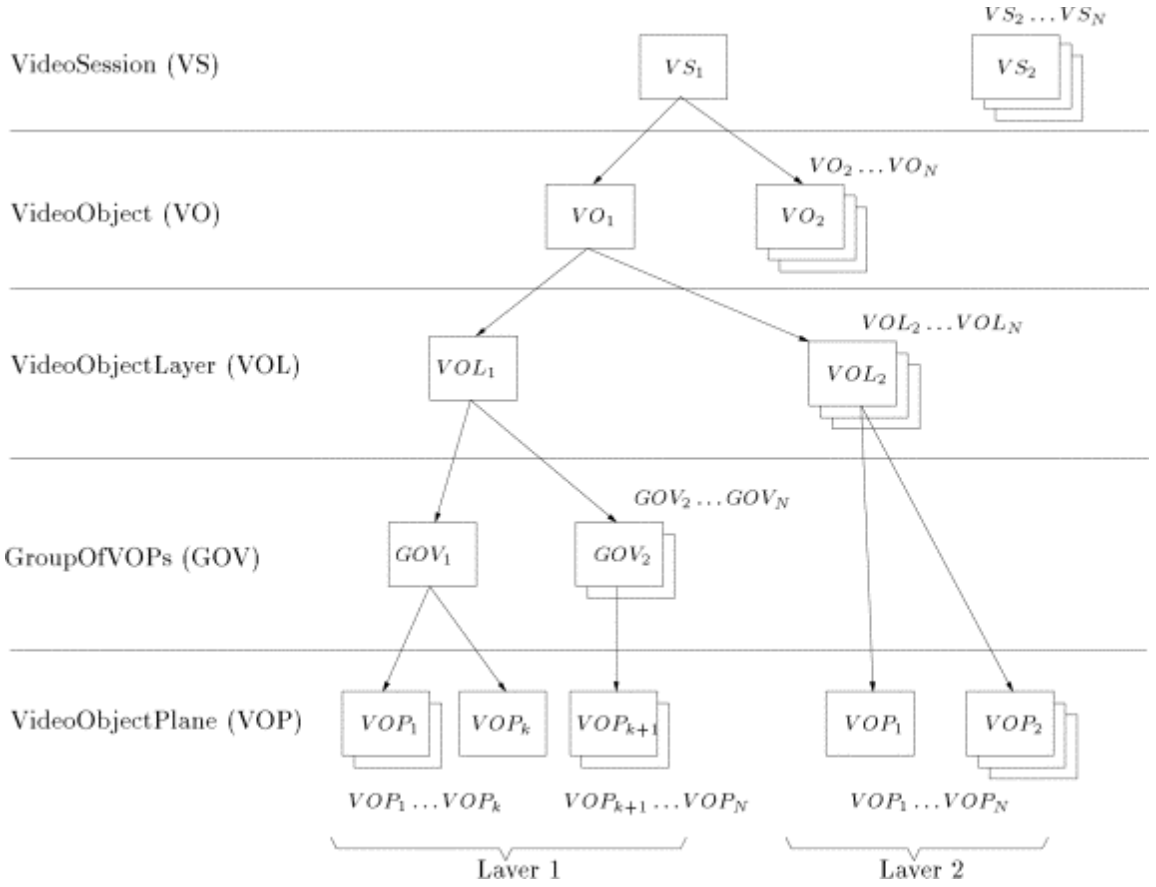

Figure A.16 - MPEG4 Video Object Layers

The top part is the Video Session (or Video Sequence). This part encapsulates the entire MPEG-4 visual scene. The next part is the Video Object which is a single object in the scene. This object can be a random shape or can be the background. The next part down is the Video Object Layer. This part enabled scalable coding. The forth part is the Group of Video Object Planes which is an optional level that groups together different Video Object Planes. The final part is the Video Object Plane. This is a snapshot of a Video Object at a particular instance in time. This part holds information such as the object's shape, texture and motion parameters at that instance in time.

Object Oriented Coding

The encoding process of this standard encodes each VOP separately. Block matching algorithms of previous standards only provide best match possibilities but might still find an incorrect match. If that is the case, then motion of the object would be wrongly predicted. This new coding method will ideally obtain a unique motion vector

that is always consistent with the VO's motion. This new coding technique however still relies on previous motion compensation techniques. An I-frame coded VOP is called an I-VOP, a P-frame P-VOPs or B-frame to B-VOPs. A new component now however must be considered. As before the shape was always a rectangle of given block size; now the shape to be coded is arbitrary. This means that shape information must now be incorporated into our coding process; this is a step that MPEG-4 refers to as texture coding.

Shape Coding

Because the VOP's shape is not necessarily the exact shape of a macroblock, its shape information needs to be encoded. There are two types of shape information that MPEG-4 encodes; binary and grayscale. Binary shape information is a binary map that is the same size as the VOP's bounding box. It has a value of 1 if it has the object at that pixel, or 0 if it doesn't. Grayscale shape information actually displays the object's intensity values. In the grayscale map, values range from 0 – 255 to show the intensity of the object.

Motion Compensation

VOP based motion compensation involves three steps similar to motion compensation of previous compression standards: motion estimation, motion compensation based prediction, and coding of the prediction errors. Each VOP is surrounded by a bounding box which is then broken up in 16 x 16 macroblocks for luminance and 8 x 8 for chrominance (as is for previous standards). Both luminance and chrominance boxes must be multiples of 16, therefore the bounding box is usually larger than the actual VOP.

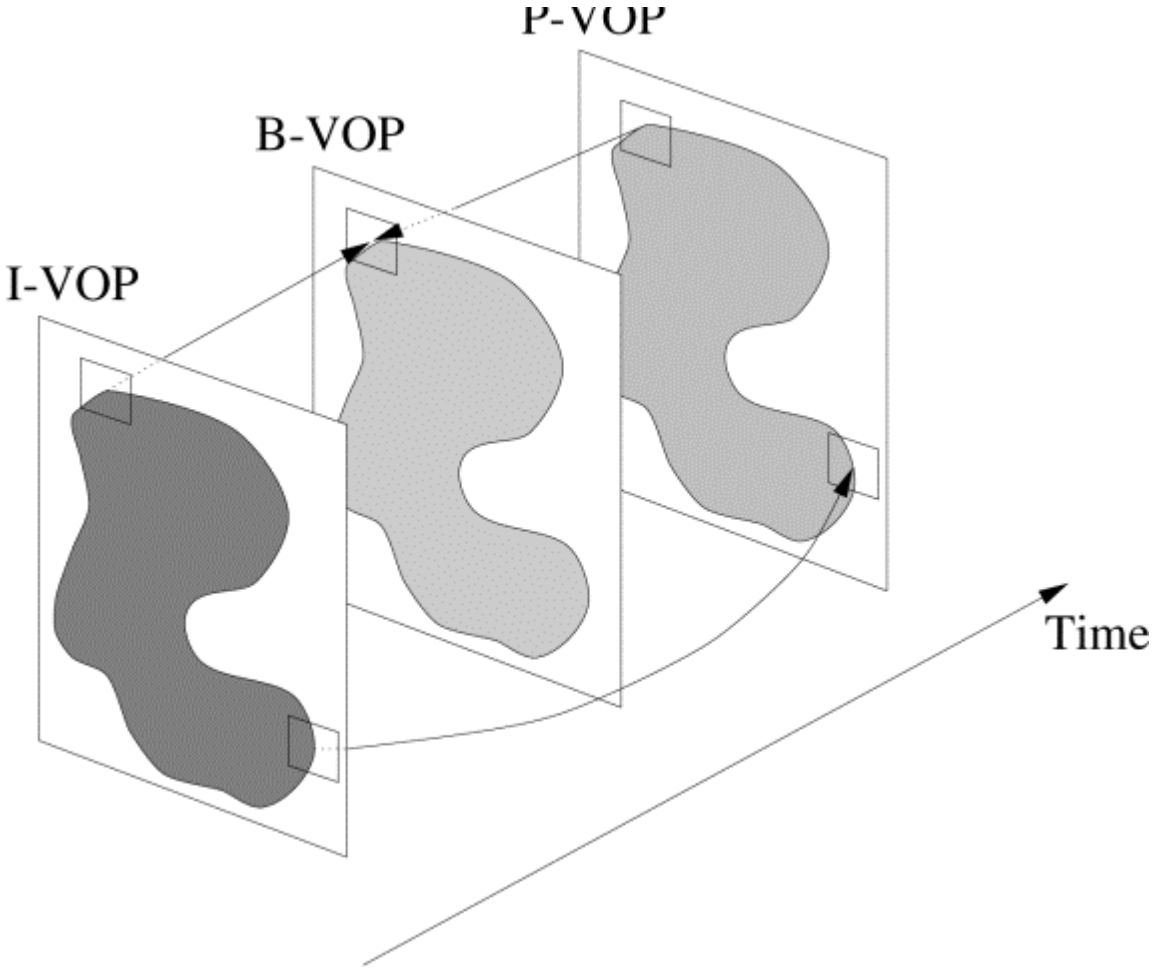

Figure A.17 - MPEG4 Frames

The macroblocks that contain only the VOP (not border blocks) are called interior blocks and the ones that lie on the borders are called boundary blocks. Motion compensation for interior blocks is identical to that of previous standards. The compensation for boundary blocks however is handled differently. In order to try and match this boundary in another block, some padding must be added.

Texture Coding

Texture coding in MPEG-4 is performed on the DCT coefficients. For an I-VOP, the values of the pixel in each block of the VOP are coded using DCT followed by a variable length coding (similar to JPEG). However, for a P or B-VOP, it uses motion compensation, therefore its errors are sent to the DCT and then through VLC.

The coding of any interior block is the same process as that of the previous compression standards. For boundary macroblocks however, the process is different due to the added padding information. It first calculates the motion compensation, and then the texture prediction errors within the target VOP are found. For the portions of the block that that fall outside the VOP, zeros are padded to the block and then sent off to the DCT.

Glossary

HTTP	Hypertext Transfer Protocol
Packet	Formatted unit of video data carried over the network
Packet Queue	An ordered collection of packets
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
SDP	Session Description Protocol
Socket	An endpoint of an inter-process communication flow across a computer network
TCP	Transmission Control Protocol
Thread	Smallest unit of processing that can be scheduled by an operating system
URI	Uniform Resource Identifier