ENG4112 – DISSERTATION

VIDEO PROCESSING ROAD SAFETY SYSTEM

RUSSELL JOHNSEN BIRKETT – U1046481

SUPERVISOR: TOBIAS LOW

University of Southern Queensland

Faculty of Health, Engineering and Sciences

ENG4111 & ENG4112 Research Project

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering and Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.
Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering and Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitles “Research Project” is to contribute to the overall education within the student’s chosen degree program. This document, the associated hardware, software, drawings, and any other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Certification

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Russell Johnsen Birkett

Student Number: 0061046481

Contents

Abstract

The purpose of this report was to investigate the possible applications of computer vision processing in roadway safety. With the huge expansion of computer vision software as well as affordable processing hardware opportunity to apply established methods to new areas has arisen. A literature review was undertaken investigating and comparing algorithms in computer vision and their possible application for roadway safety. Through the literature review it was decided the most effective application for pedestrian tracking would be a combination of Gaussian Mixture Modeling for background subtraction and a Kalman filter for tracking. This in conjunction with a range of morphological filters was the final design decision. This design was cemented as a Simulink model and was run against a series of scenarios. The most important results were testing against a dynamic roadway showing extremely promising outcomes. The model was able to detect and track pedestrians over a range of designated areas. This in addition to its ability to differentiate between cars and pedestrians and the location of those pedestrians in relation to the road made the system a success. With simple counting and regions of interest changes the model can be applied to range of different environments and scenarios. However problems did emerge in the implementation of the Kalman filter causing issues in its application and track association. This report was dedicated to investigating and applying current methods and techniques in computer vision that could benefit pedestrian safety, in this regard it has successfully represented the possible uses and application of such a model.

Aims

The focus of this project will be directed at making robust video processing software that can be run through road footage to detect entities and obstructions on the roads and surrounding areas. This in turn can be used to warn drivers and pedestrians to avert accidents. Safety is the most important aspect in designing a road network for both pedestrians and motorists. Therefore any improvements or added systems that can reduce the risk and increase the safety must be considered and evaluated to ensure that any safety aspects are not overlooked. With the sudden increase in small affordable computing devices and cameras alike the possibility for an integrated road safety network monitoring roadways and detecting obstructions and possible risk has become achievable.

The central element in the design and implementation of this system will be based on how adaptive and flexible the program will be, needing to be applied and utilized on a range of different environments and camera capabilities. Current technologies for both the hardware and software will need to be analyzed in order to determine what methods and processes will be the most effective. The aim of this project will be determining how to effectively develop and apply these processes to the Australian road network. However a huge component of this project will be in the comparison of the developed code against one already existing comparing them in a range of different datasets and environments. With the comparisons and analysis completed the project can then move to an iteration stage where further improvements can be implemented consistently throughout its development.

With a competent and fully developed process of analysis the further implication and uses of the program can then be expanded. Looking at what types and different applications the system can be applied to and its comparison to what is available in that field. This is to determine what possible further modifications can be made and what future work can be applied and built upon the model.

Objectives

The overall objectives of this project are very straight forward. The basis of this can be outlined in four steps

1. Literature analysis and review – look into current form of 2d monocular camera tracking and find similar and necessary elements to implement into a working code
2. Develop a working code through Matlab with the incorporation of leant and proven techniques demonstrated and analyzed through the literature review
3. Compare current version of the code to current technologies – Analysis its current strengths and weaknesses and further improve until satisfied
4. Apply in a real world scenario's at a range of different environments

The bulk of the work of this project is centralized around comparing the current technologies and determining which method is the best suited to the project. This will require an in-depth literature review looking at and discussing all the current applications of visual tracking and their overall success. This background research will be critical in developing a working code and an overall successful project.

The field of video processing is one that has expanded massively over the past few decades with the implementation of security cameras and visual systems throughout many different industries. This has inevitably led to new possibilities in developing programs which can be run through footage to create a very inexpensive interactive system.

The current research around this topic is extensive with the field of machine vision being a huge overlaying topic. The application of this technology to road safety could open new possibilities ways in how accidents are averted, monitored and studied. The two main components of this system creation would be hardware development and software and the software development. Both these sections have many intricate parts and details that need to be analyzed in order to create the most effective prototype.

The development stage will rely heavily on the results and analysis take from the literature analysis. These results will then have to be applied to my current build making sure to evaluate and modify the body of the code for the desire output. This must then be constantly compared against current methods to see if the results are up standard. Seeking out and finding literature with comparisons and testing on each different type and method of tracking will be paramount on the design of a successful project.

Introduction

The need for tracking and trajectory prediction of dynamic objects is one that applies itself to a very broad range of situations and solutions. The ability to successfully plan and record the movements of pedestrians is becoming an extremely relevant technique in road safety. Simply detection of the amount of pedestrian traffic through a road way could be easily used in the decision making and implementation of safer road networks.

The current technologies available for application in this environment are near endless with many different methods each with their own individual characteristics. This is where the extreme diversity in different algorithms for not only background subtraction and entity tracking come into play with many of these being interchangeable giving a range of different strength and weaknesses for each combination.

The focus of this project will be on 2d monocular stationary cameras. This form of camera has been chosen for its comparable and similarities to modern security cameras. This gives the project a large base of possible applications as well a range of different testing opportunities. This form of camera however only receives information from direct video feed having no outside information such as depth being applied. This forces whatever method that is to be applied to use no outside data sources and must perform all operations within information provided. This in comparison to utilizing multiple cameras for such methods stereovision or applying information taken from sensors will not be applicable within the scope of this project.

Literature Review

Basic tracking systems can be broken down into three central stages. The first is object detection; this stage is essentially separates the background from the desired targets. This initial step is extremely important in developing the ground work for the rest of the operations. The second stage is noise removal and reduction, utilizing various methods to clear up the overall quality of information that can then be used in the final stage of tracking. These three stages have a massive range of various ways to approach and conduct them all with different attributes and disadvantages.

Initial investigation into the possible methods showed a range of different goals different papers wanted to achieve. These goals can be placing into two main sections; increased accuracy and quick response. Specific papers focused on making the accuracy of detection more important than having the ability for live analysis. An example of this can be seen in a paper by Führ Gustavo on calibrated monocular pedestrian tracking. This report utilized a median filter and a range of post processing to calculate the position of the pedestrians. When run this code was extremely accurate detecting all the required targets and estimating there dimensions. However heavy calibration needs to be conducted on the camera and environmental dimensions entered into its operation. Not only this but the time taken to evaluate each passing frame was upwards five seconds, destroying its possibility for live relay (Fuhr, 2012). This in contrast to a paper based on surveillance systems utilizing live footage to analyze and track targets showed the two different goals. The Kalman Filter based tracking system was designed to be extremely efficient attempting to use as little computational resources as possible. It explains that the two biggest factors in tracking are *"the accuracy to distinguish between contacts passing through the scene and the speed to process the video feed in real-time"* (SULIMAN, CRUCERU, & MOLDOVEANU, 2010). It overcame this problem by selecting the specific algorithms that work most effectively however not exceptionally accurate. This is the balance that is needed in designing this system looking at how much accuracy you can achieve while still acquiring real time performance. With the goal of creating a system that can instantly warn pedestrians and motorist of hazards the live analysis method will need to be taken ensure that any formulas utilized are optimized. This will require looking at all the stages of detection and tracking and comparing the available methods and there overall benefits and disadvantages.

.

Background Subtraction

One of the necessary decisions in creating effective tracking software understands what best type of background subtraction or foreground detection method to apply. This initial step separates the background from the desired targeted entities. This can be argued is the most crucial step becoming the foundations and base in which the rest of the program will be built upon. Background subtraction is a way of localizing moving objects in a video shot by a static camera. This is the initial step in motion detection of a multi-stage computer vision system (Benezeth, Jodoin, Emile, & Laurent, 2012).

Background subtraction can be done in a myriad of different ways with each having its own specific advantages. Some of the simplest methods such as color separation rely heavily on what target you are attempting to track. Having a target that can be replicated in size and color lends its self perfectly to simpler methods. In most cases the simplest method is usually the best for most effective over a range of different areas. For example in color separation and detection the colors that most appropriately match the target are taken as the foreground removing all other colors into background. This simple method applied with post processing can be extremely effective in separating what is to be tracked. With most tracking methods stationary cameras are required however with color separation having the changing background being taken as all the wrong color and ignored opens it up to the possibility of tracking through a moving image (Su & Fang, 2012). This single example begins to illustrate how each specific method has is benefits and problems. The problem with color tracking being the need for a predetermined color constraint on the targeted entity with little opportunity for variable change after setup. This limitation makes it unviable for tracking pedestrians due to the varying color attributes. This forces techniques that need other possible indicators to be used to determine targets.

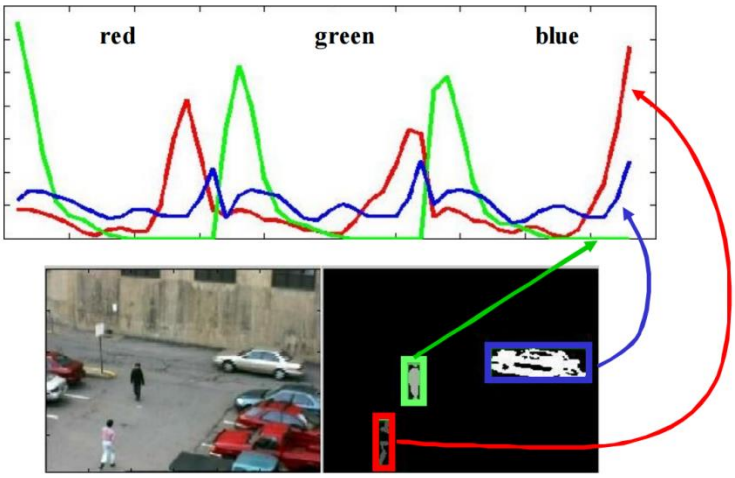


Figure 1 - Color separation Histogram

The next step up in complexity is edge detection. This form of detection creates boundaries and outlines throughout the image by finding discontinuities in the brightness (Ray, 2013). This can then be used to detect and segment between each of the objects in the frame and its background. This can then look at if the edges are in motion and highlight them for tracking. This method works exceptionally well in stationary environments where there is little movement in the background frame. However when a large background movement is applied it cannot differentiate between what is to be targeted and what is the background. Due the central detection being around changes in brightness between the background and the moving image further noise cancelation has problems with the resultant edge detection being much similar intensities to those of the background. With this in mind however edge detection can be done through range of different algorithms each of which changing and affecting on the output (MathWorks, 2015).

Figure 2 - Sobel Method

Figure 3 - Canny Method

Figure 4 - Furry Logic Method

One of the biggest categories of background subtraction however for stationary 2d monocular vision is focused around the changing of pixels though out frames. This method is confined to a stationary camera with its ability to operate very much dependent on the frame motion comparing one frame to another. If the video frames are in constant motion it is unable to successfully filter out the background and foreground with the detected motion being merged into one. This process of looking at pixel changes has been a highly researched and experimented topic yielding many different techniques that can be experimented with. In a basic study of comparing the background subtraction methods conducted by Yannick Benezeth and published in the Journal of Electronic Imaging showed how these difference formulas affected the overall output of the projection. This paper specifically analyzes seven different techniques rating them against different environments and computational requirements. The seven methods they used included:

1. Basic Motion Detection
2. One Gaussian modeling
3. Inter-Frame Difference
4. Gaussian Mixture Modeling
5. Kernel Density Estimation
6. Codebook Method
7. Eigen Backgrounds

Each of these methods have their own specific formula's and algorithms however they all share a common denominator that being the assumption that the observed video sequence is separated into two different parts. These being a static background in which the moving objects and entities can be observed. This can be summarized in the formula

$$\mathcal{X}_t(s) = \begin{cases} 1 \text{ if } d(\boldsymbol{I}_{s,t}, \boldsymbol{B}_s) > \tau \\ 0 \quad \text{otherwise,} \end{cases}$$

Where $\boldsymbol{\tau}$ is a threshold, $\mathbf{X_t}$ is the motion label field at time **t**, **d** is the distance between $\mathbf{I_{s,t}}$ the color at time **t** and pixel **s**, and $\mathbf{B_s}$,the background model at pixel **s**. This essentially explains that to break into the threshold there has to registration of a comparative change from the background model and the subsequent next frame. This is the central basis in which all the different algorithms are based off finding more accurate detection or quicker less computational taxing algorithms (Benezeth, Jodoin, Emile, & Laurent, 2012).

Through testing these seven techniques over a range of different environments began to expose the expose the more effective and versatile models.

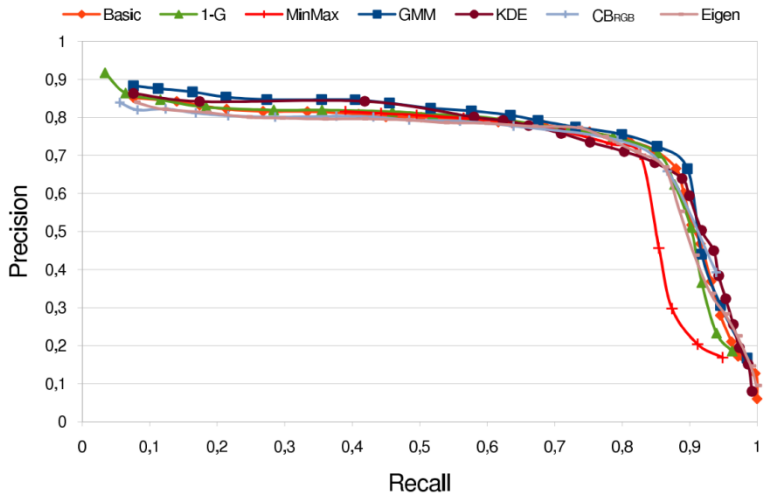


Figure 5 - Low noise environment

In a noise free background the results showed very little difference in the overall ability of the algorithms. This however changed drastically with the introduction of noise in the background separating the more effective models out.

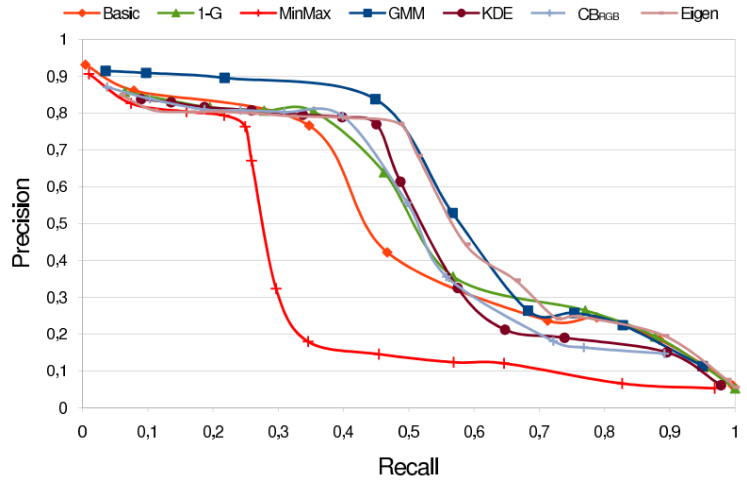


Figure 6 - High noise environment

With the addition of noise specific method become significantly better and more effective than others. This conclusion of this study comparing the various background subtraction methods weighted each method against the three various environments and further against its computation requirements.

the method

	Basic	1-G	MinMax	GMM	KDE	CB_{RGB}	Eigen
Static background	***	***	**	***	***	***	***
Multimodal background	*	**	*	***	***	***	*
Noisy background	*	***	*	***	***	***	***
Computation time	***	***	***	**	*	-	*
Memory requirement	***	***	***	**	*	**	*

Figure 7 - Background subtraction Comparison

The final results explained that each various technique had its own specific strengths and weakness although some forms were much more robust in applying to different situations. The standout algorithm from this paper was the Gaussian Mixture Modeling (GMM) performing extremely well in the range environments and as well in computational requirements.

This consensus in the robustness of GMM is further explained in another background subtraction comparative study by Andrews Sobral of the University of De La Eochelle in France shown in appendix E. This Study again focused on a comparison of background subtraction algorithms on synthetic and real environments (Sobral & Vacacant, 2013).This selected the five best algorithms with Gaussian Mixture Modeling being one of the strongest of the five. The paper further explained that *"Gaussian Mixture Modeling has shown good performance for the analysis of outdoor scenes, and has become a very popular BS algorithm. Even if this method is able to handle with low illumination variations, rapid variations of illumination and shadows are still problematic"* (Sobral & Vacacant, 2013). These problems however are relatively small with the majority of detection and the higher precision of the algorithm much more effective than the majority of others. This can be further collaborated with another paper titled *"Implementation and Performance Evaluation of Background Subtraction Algorithms"* where three chosen methods derived from prominent proven papers. These three methods again were compared in a range of situations coming to the conclusion that "*This method has the best result among all the above method discussed because it can deal with slow lighting changes and other challenges. It also deals with multi-modal distributions caused by moving branches of tress, snow falls, shadows, flying birds and other troublesome features"* (Das & Saharia, 2014).

From these comparative studies Gaussian Mixture Modeling stand out as the most effective background subtraction and foreground detection method. Its ability to reduce noise in an environment with small changing factors such as light changes and its all-round robustness shown through the three comparative studies makes it an extremely promising method. For this reason the details on what factor affect it and the most effective way to implement will need to be analyzed.

Morphological Filters

After the primary separation between the background and foreground is conducted secondary processing may take effect. The goals of these processes are to reduce the overall amount of noise in the image resulting in a cleaner and easier to detect targeted entity. Mathematical morphology is the branch of image processing and analysis that employs morphology as the key instrument to interpret and extract features from images (Roeder, 2012). This outlines that the various different mathematical choices each have their own respective effect on the type of morphology.

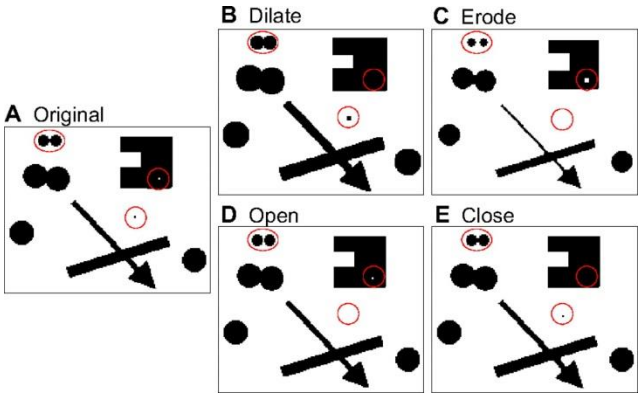


Figure 8 - Morphological Operations

One of the biggest aspects in using these filters is the range and ratio of how much each individually should be applied. By using a combination of these operations the user can fine tune for a myriad of different environments. This is why these processes are so necessary giving the system the ability to be applied in range of different environment with simple adjustment being made to the post processing.

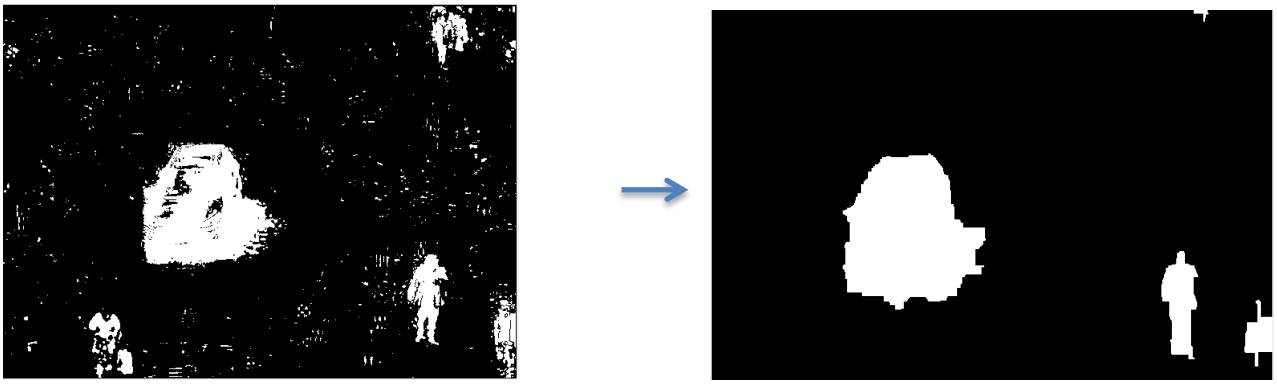
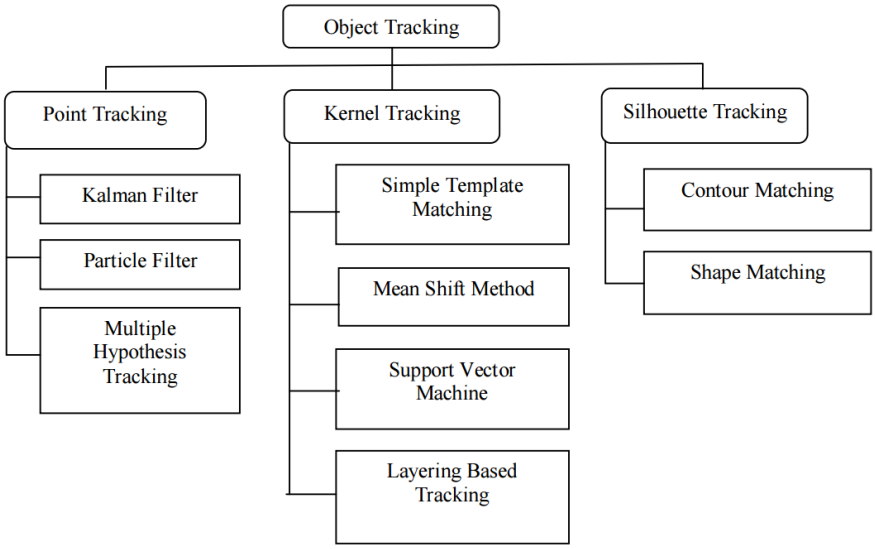


Figure 9 - Morphological Application

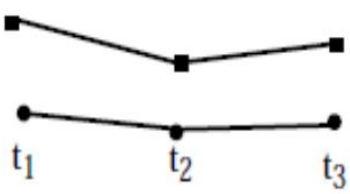
Tracking algorithms

From detection and secondary processing the next stage is applying a tracking algorithm. This is done to accommodate the problem of collision and divergence. With the use of monocular cameras depth is extremely to attribute to entities and in the process of passing or standing very close relation to each other this can cause the model to register them as a single unit. This can cause huge issues in more advanced areas other than detection. For example needing to accurately count and differentiate between pedestrians is a critical element in advanced tracking and association. Not having the ability to give identities and labels to a specific unit when introduced to a cluttered environment is a massive weakness of the system. For this reason the introduction of tracking algorithms are implemented in a large amount of surveillance systems to ensure and add another level of security to the level of information being received. For this reason the use of predictive tracking filers are applied in order to remove this problem and successfully identify each target before and after their foregrounds diverge or collide.



Points Tracking

Object tracking can be broken down into three direct categories; Point tracking, Kernel tracking and silhouette tracking. Point tracking is the process of representing moving object features as points during tracking. Recognition can be done relatively simple, by thresholding, at of identification of these points (Athanesious, 2012). This essentially takes the targeted entity and represents it a series of points over a specific times.



From these points further analysis can be completed looking into various characteristics such as the point's proximity in relation to others, Changes in velocity both minor and major and its mutual motion when looking at points in the surrounding area and time intervals. These applied in practice gives the three distinct point tracking types.

(a) Kalman Filter: This specific filter is based on Optimal Recursive Data Processing Algorithm. In other words, they are tracked based on the criteria chosen to evaluate performance. Optimal point will be taken based on criteria that make sense. The Kalman Filter performs the restrictive probability density propagation (Athanesious, 2012). This ability to have two phases of prediction and correction combined together to achieve a result with a higher probability. It does this through having specific weighted values for noise and environmental conditions. Using these values to weight the equation towards a more predicted state in which high noise causes the equation to predict when the point is or low noise causing the algorithm to use the observed information. With this ability the Kalman filter always gives the most optimal solutions (Parekh, Thakore, & Jaliya, 2014).

(b) Particle Filter: The particle filter generates all the models for one variable before moving to the next variable. This algorithm usually uses contours, color features, or texture mapping to determine its points. The particle filter is a Bayesian sequential importance Sample technique, which recursively approaches the later distribution using a finite set of weighted trials (Athanesious, 2012). Like the Kalman Filter this method also uses a predictive and update filtering making is very effective in giving optimal solutions. However unlike the Kalman Filter is does not rely heavily on variable that are normally distributed (Poole & Mackworth, 2010).

(c) ***Multiple Hypothesis Tracking:*** If motion correspondence is recognized using only two frames, there is always a limited chance of an incorrect correspondence. Better tracking outcomes can be acquired if the correspondence choice is overdue until several frames have been observed (Athanesious, 2012). Essentially this means that Multiple Hypothesis tracking is an iterative algorithm. With each hypothesis the algorithm predicts the objects position in the next frame and then these predictions are compared by calculating what the distance change in feeding this information back into the iterations. This then suggests based of the comparison in the previous frames what the most like set of correspondences will be over the observed time period tracking the object.

Kernel Based Tracking

Kernel tracking is the method of using real-time illustration of objects geometric features, appearance and shape of the object (Athanesious, 2012). The object can be detected in rigid and non-rigid states and are tracked based on what representation was given of the desired targeted entity. This tracking has a massive range of different methods all of which changing drastically in how they implement the kernel tracking.

(a) Simple Template Matching: Template Matching is a technique for processing digital images to find small parts of an image that matches in corresponding frames. The matching procedure contains the image template for all possible positions in the source image and calculates a numerical index that specifies how well the model fits the picture that position (Parekh, Thakore, & Jaliya, 2014). Basically a template of the targeted entity is input into the algorithm and comparisons are made to identify the position of that target from its represented features. This forces this method to only be compatible with a small number of tracking targets and the overall identity and features of the target to be known.

(b) Mean Shift Method: Mean-shift tracking tries to find the area of a video frame that is locally most similar to a previously initialized model. The image region to be tracked is represented by a histogram. A gradient ascent procedure is used to move the tracker to the location that maximizes a similarity score between the model and the current image region. (Parekh, Thakore, & Jaliya, 2014). This again limits the number of tracking targets possible with the identity histograms needed of each desired target. However if the target and its representation is well defined tracking is very effective especially in partial occlusion.

(c) Support Vector Machine: This method utilizes a set of positive and negative training values to distinguish between the tracked object. This forces all samples that correspond with the negative values to be ignored and the remaining value to be compared between was represented in the positive values (Athanesious, 2012). This again required the definition of the targeted entity forcing down the overall amount of possible tracked targets.

(d) Layering Based tracking: This method has the ability to track multiple targets through the kernel method however a representation is still needed to be input. This input is a layered representation of the target at various different rotations, translations and intensities. This then analyses the probability of each possible outcome and compares it to the tracked entity (Athanesious, 2012). This algorithms use of multiple representations gives it’s the ability to successfully identify the target over significantly more environments.

Silhouette Based Tracking

Silhouette Based tracking utilizes the objects composite shape and defines it into an object model. This model is then used as to verify the tracked object in each frame by comparing it to the designated models. This can be done in two central ways either though matching the shapes of the represented and tracked object or through contour tracking (Athanesious, 2012). This is also furthered by the ability to redefine the silhouette depending on the previous frames. With represented object definitions occlusion, divergence and merging of targets can be easily handled making it extremely useful in high unit environments.

(a) Contour Tracking: Contour tracking uses state space models to continuously update its targeted model over a series of frames. Initially developing an original contour in the foregoing frame the comparing its new position in the present frame through overlapping of object between the current and next frame. This then redefines the new contour and uses this new representation for the subsequent frames updating based off probability each time (Athanesious, 2012).This expands the scope of what targets can be tracked and what the targets translations can be throughout the frames.

(b) Shape Matching: Shape Matching is extremely similar to kernel based template tracking in that a representation of the targeted entity is entered and the algorithm matches its shape to the target entity in the frame. This method also looks at successive frames for matching silhouettes. Attempting to find if the designated silhouette continues on the path similar to point tracking (Parekh, Thakore, & Jaliya, 2014).

The most effective class of tracking was determined to be the point tracking. Without the need for any reference images and its simple application made it the most desirable choice. The problems facing Kernel and Silhouette tracking is the small number of target that can be tracked as well as need for references of the desired targets. The type of point tracking however will need to be investigated further and how the algorithm may be applied for video processing.

Evaluation

The literature review yielded large amount of useful information as well as cementing a number of algorithms and techniques that should be used in the latter project work. Background subtraction had huge amounts of material all of which included in depths analysis and comparative studies. These comparative studies outline and explaining the strength and weakness of the various algorithms made the decision of Gaussian Mixture Modeling a well thought out explained and credible one. However information based around special aggregators was fairly scarce, with only small amount of papers explaining the comparing the various techniques. Although there was a lack of comparative studies on these second stage filters the overwhelming response from them was implementation of a morphological filter. This filter was deemed to be superior in its ability to preserve, uncover and detect geometric structure of image objects making it significantly better than any other linear filters (Maragos, 2005). The fluidity and cooperation between Gaussian Mixture modeling and Kalman filters made its choice extremely easy and well founded. Its ability was outlined in the journal of *Transactions on Signal Processing* exclaiming that *" For a particular problem, if the assumptions of the Kalman filter or grid-based filters hold, then no other algorithm can outperform them"* (Arulampalam, Maskell, Neil, & Clapp, 2002). Unlike Kernel based and Silhouette based tracking the particle filters needs little inputs in order to begin tracking summarizing the detected entities into single set of points which can then have tracking performed on them. However the exact formulas for each of the step in both the Gaussian Mixture Modeling and how to successfully apply a Kalman Filter must be investigated.

Gaussian Mixture Modeling (GMM) and special aggregation

Gaussian Mixture modeling is probabilistic based model that has been used in a range of different areas and applications. It utilizes functions that represent itself as a weighted sum of Gaussian component densities that can then be applied to create a dimensioned representation of the data (Reynolds, 2009). This given GMM the ability to create form smooth approximation of arbitrarily shaped densities applying itself perfectly to detecting the change in pixels. The equation determine if these pixels are background is described by

$$P(\mathbf{z_t}|\mathbf{m_t}) = \sum_{n=1}^{N} \frac{\alpha_n}{(2\pi)^{d/2}|\mathbf{\Sigma_n}|^{1/2}} e^{-\frac{1}{2}(\mathbf{z_t}-\boldsymbol{\mu_n})^T\mathbf{\Sigma}^{-1}(\mathbf{z_t}-\boldsymbol{\mu_n})}$$

Where $\mathbf{Z_t}$ is the dimensioned pixel at the dimension, **d** at a specific time, **t**

Each Gaussian, **n** is described by both its mean $\boldsymbol{\mu_n}$ and covariance matrix $\boldsymbol{\Sigma_n}$

$\boldsymbol{\alpha_n}$ is the prior probability or the weight of the Gaussian (Sobral & Vacacant, 2013)

This shows how it separate from foreground and background. This is done by determining its weighted probability ie that it has been affected in the previous frames and multiplying it by not only its mean but also by the covariance matrix. By creating a probability based network small changing in singular pixels by unwanted factors are ignored not having enough to create a sizeable probability factor.

This given the GMM an advantage over simple frame difference models by letting it adapt and ignore unwanted noise though the use of Gaussian probability. This as explained earlier lets the model create smooth approximations of the shapes by removing jarring outliers and smoothing rough edges through weighting differences. This can be further improved through the influence of spatial aggregation, a process that eliminated small isolated abnormalities reinforcing the smoothness of the foreground detection (Benezeth, Jodoin, Emile, & Laurent, 2012).

These spatial aggregators work as extra filters that are applied after the bulk of the work is done in this case after Gaussian Mixture Modeling. This is explained by Jinanbin Feng in his report on *Decision-based adaptive morphological filters* explaining *"In the process of image formation, image recording, image transmission via sensors or communication channels, image will be inevitably corrupted by impulse noise due to sensor malfunction, transmission errors, storage faults, and difficult acquisition conditions. Characterized by short, abrupt alterations of the intensity values in the images, impulse noise will cause image degradation by producing the significant intensity difference between the corrupted pixel and its local neighborhood"* (Feng , Ding, & Zhang, 2013). This is extremely relevant with faults and anomalies still know to make it through the Gaussian mixture model. The type of filter recommended in this paper is a morphological filter explaining it benefits over median based filters. The reason for the choice of a morphological filter is due to its capability of detecting the corrupted or false pixels and using erosion to reduce their effect. This however cannot be done with median based filter due to the fact they cannot differentiate between the required pixels and the false positives (Feng , Ding, & Zhang, 2013). This view point is expressed further with a paper outlining morphological filtering and its rivals of median, rand and stack filters. This paper named *"Morphological filtering for Image and Feature Detection"* and published in "*The Image and Video Processing Handbook"* explains that its ability to preserve, uncover and detect geometric structure of image objects is significantly better than any other linear filters (Maragos, 2005).

Point tracking - Kalman Filter

Another huge problem in tracking pedestrians and objects is collision and divergence. With this happens a basic detection algorithm will not be able to determine which entity was which and cause immense problems in counting and prediction there movements. With the use of monocular cameras depth is impossible to attribute to entities and in the process of passing or standing very close can cause the model to register them as a single entity. For this reason the use of predictive tracking filers are applied in order to remove this problem and successfully identify each target before and after their foregrounds diverge or collide. With the targeted entities being pedestrian and their movement being somewhat linear and predictable it makes sense to apply a linear based particle filter. In a paper published in the Journal *"Transactions on Signal Processing"* Dr. Arulmaplam conducts a comparison and analysis on the different types of filtering methods both linear and non-linear. In his conclusions he exclaims that*" For a particular problem, if the assumptions of the Kalman filter or grid-based filters hold, then no other algorithm can outperform them"* (Arulampalam, Maskell, Neil, & Clapp, 2002). This relates perfectly to the requirement of a prediction tracking in which the targeting entities will be bound by specific laws and requirements that can be inputted into the Kalman filter.

The basic concept of the Kalman filter can be summarized as a particle filter that utilizes a weighted particle set to estimate the system state in conjunction with the sequential importance sampling method (Zhou, Wu, & Zhu, 2016). This essentially states that the filter finds the most optimum factors from the each state while add/including date from the previous states. This is perfect in the case of detecting entities diverging and moving though each other with the filter tracking the last states and using linear preset laws to determine which entity is which. The algorithms defining this filter are as follows

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

$\dot{\mathbf{x}}_\mathbf{k}$ is the state prediction

$\mathbf{A}\ \dot{\mathbf{x}}_{\mathbf{k-1}}$ is the prior pixel matrix state

$\mathbf{Bu_k}$ is the control input

K is steps in frames

$$P_k^- = AP_{k-1}A^T + Q$$

$\dot{\mathbf{P}}_\mathbf{k}$ is the predicted co-variance

$\mathbf{AP_{k-1}A^T}$ is the previous co-variance

Q is the expected co-variance

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

$\mathbf{K_k}$ is the Kalman gain

H is the observed pixel matrix

R is the error co-variance

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$\mathbf{x_k}$ is the final state

$$P_k = (I - K_k H)P_k^-$$

$\mathbf{P_k}$ is the final co-variance

I is an identity matrix

How these formulas apply to the Gaussian Mixture modeling is extremely effective and ingenious. Firstly the state prediction is made through the addition of the previously measured pixel state and added to the control input defined in the model. Then the predicted co-variances are calculated through the similar process of adding the previous variance state an expected co-variance.

The Kalman gain formula is essentially a measurement of the information given. It can be dived into two separate parts; firstly $\mathbf{\dot{P}_k * H}$ which is simply the predicted co-variance multiplied by the observed and $\mathbf{(H* \dot{P}_k * H^T + R)^{-1}}$ which is extremely important. The seconds half of this equation incorporates the Error (**R**) this error factor has a huge effect on the Kalman gain. The larger the error calculated the smaller the Kalman gain. This makes perfect sense as the gain will be used to determine how much and how accurate the information from the new pixel matrixes are. However this can also being used the other way with an extremely low error value giving the Kalman gain a larger number directly relating to how accurate the information is in comparison from the predicted and observer matrixes.

This is all incorporated into the final state where the prediction is added to the Kalman gain weighting which is multiplied by the correction. The final step determines the final co-variance. This formula utilizes an identity matrix to formulate how accurate and how much impact the measure values ($\mathbf{K_k*H}$) will have on the predicted co-variance. With a larger ($\mathbf{K_k*H}$) the smaller the final co-variance will be and the better the estimation. The smaller the measured value the larger the co-variance will be relying more on the predicted co-variance.

Implications and Consequential effects

The implication of creating a working model would be extremely beneficial in accelerating and furthering the research and development of safety systems. The current technology is on the verge of being applicable to huge different areas and situations all with their own specific needed and environments. Simply creating a software that can count, monitor and record pedestrian movements could be immensely important in a huge range of situations. Detection how many pedestrians move past a camera on a certain day or if there is a sudden large influx of pedestrians can be used not only as data for statistics but also how to deal with rapidly changing situations. An example of this is if a sports game suddenly finished and a huge amount of pedestrians are pushed out onto the reads. This information could be relayed directly to an arrangement of areas to increase pedestrian crossing times and frequencies and alert public transport to these unforeseen circumstances. Situations such as monitoring roadways for pedestrians or simply counting traffic autonomously all have huge benefits in a multitude of areas. Having a system that can automatously count and relay information back to a central hub or simple processing on site can benefit a massive range of traffic and safety systems.

The further applications of a system in which detections, counting and tracking can be applied are enormous. The surveillance applications are extensive having a simple system to detect when people enter undesired locations or apply to basic counting of pedestrians walking down a path. This information could then be utilized in a myriad of ways in crowd analytic and further research. With security camera becoming a bigger and bigger part of government tools there application towards safety seems an inevitable application. Looking at and designing a cost effective system that can be applied to a range of environments will do nothing but benefit anyone under its care.

Feasibility

The goal of this project would be to develop a working system that can detect, track and warn cars of incoming obstacles. This system in concept is very achievable with many of the technologies already being applied in various different industries. The goal of this project would be in applying them technologies successfully to improve roadway safety. This will require heavy adjustments in order to create system that can be applied to many different environments. This will be down the effectiveness of how the algorithms are applied the sensitivity of the program.

Currently there is various surveillance based computer vision systems applied in in the field of security. This field has had massive expansion over the last 20 years with the advancements of cheap micro-computers and cameras reducing the cost of implementing an integrated system. This advancement has also opened the application of these systems into a range of other fields. This has increase the feasibly of applying this detection and tracking to roadway safety exponentially with many problems already been resolved. The two central elements of this project will be designing a working program and then applying that program to hardware.

The requirements of applying a working system to a range of roadways will require the design and contemplation of how the system will operate. With the low cost approach cheap and readily available components will need to be utilized. This will require looking at what new technologies in the way of micro-computers, cameras and networked systems can be applied in order to make the project feasible for application into the real world.

Methodology

The project work and implementation have been broken down into

- Startup Phase – gathering research, resources and information based around the project
- Creation Phase – apply this knowledge to developing a working code that can be applied to simple environments
- Enhancement Phase – from the rough working model developed start applying to more difficult scenarios improving performance
- Comparison Phase – compared designed code against already developed models and advance and improve current code
- Testing – Test in a real world environment
- Prototyping – Apply the developed software to hardware
- Write up – Dissertation

Phase 1	**Startup Phase**
1A	Resource Check – Gather additional information on how other projects attempted the problem
1B	Comparison Check – Find key elements throughout predesigned projects and make links in how they applied there algorithm
Phase 2	**Creation Phase**
2A	Development - Create an outline of the needed elements gathered from research
2B	Implement – Being adding and creating the required sections shown in other work
Phase 3	**Enhancement Phase**
3A	Improvement – with a basic working program begin running through different scenarios and environments
3B	Analyze – Being noting what environment/conditions work best and worst
3C	Compare check – Check back through research for solutions and example of how to improve/ overcome problems
Phase 4	**Comparison Phase**
4A	Testing - Compare research code against current and not differences
4B	Analyze data - find how each code works and how similar technique s can be applied

Phase 5	**Testing**
5A	Test –Test current program in a real world situation
Phase 6	**Prototyping**
6A	Design - Design a possible system that can operate on the program
Phase 7	**Write Up**
7	Dissertation – Summarize all gather knowledge and development processes

Design Application

Before the program can be sufficiently designed it must first have what its specific goals and operations outlined. With its application to road safety its must obviously perform operations that benefit both the pedestrian and motorist. This implies that the system can identify the locations and movements of both pedestrians and motorists signalizing when each enters warning areas. For this reason placement and application of the camera its critical requiring a position where it can monitor both traffic and pedestrian movements. This can be seen with current security camera placements with high vantage points being the most common application of its system. This is not the only problem that faces a system that will be applied to the real world. Problems such as powering the devices as well as finding a way to have consistent light over the required area all need to be addressed.

Looking at the most cost effective method of solving these problems come to the conclusion of utilizing streetlights. This already abundant infrastructure would need very little modification to solve the problems of not only powering the device but giving a consistent light levels needed in video processing. This will need heavy testing to ensure that the lighting is sufficient and that it can suffice in giving an accurate overview of the road way. However this is an extremely cost effect strategy that overcomes a myriad of problems associated with applying the system.

The next step in application would be defining which areas of the roadway should be regarded as unsafe or dangerous. This is a problem that will need to be analyzed and judged based on the current environmental variables. Things such as pedestrian footpaths, crosswalks, car speeds and distance from footpath to road must all be investigated individually.

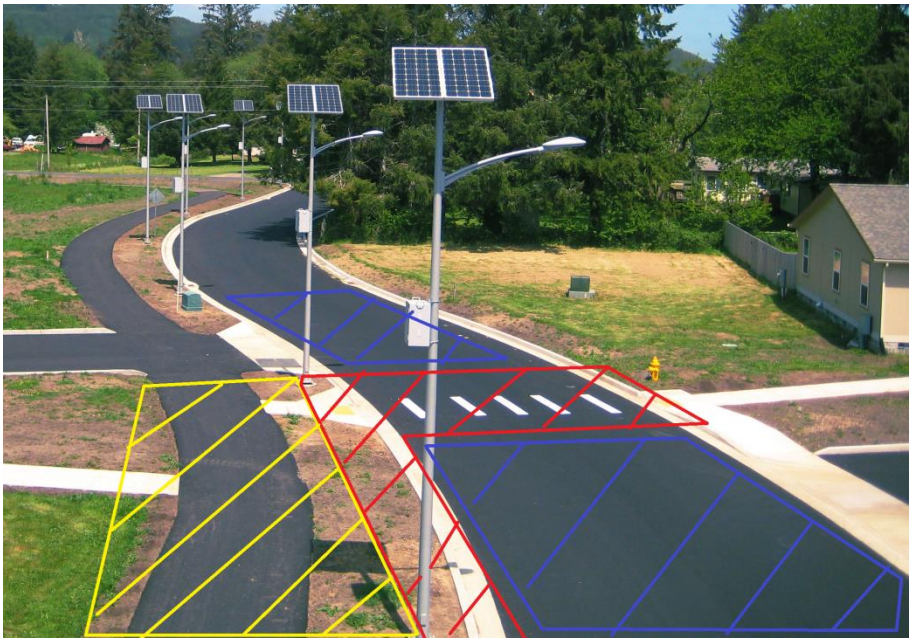


Figure 10 - Zoning of Roadway

Although powering the device could be powered through already established infrastructure the ability for the system to be a self-contained and independent device opens opportunities in more applications. This will require looking at power sources primarily at solar options making it suitable for widespread application.

With cost being a huge factor in this design the prototype will need to mimic what hardware the final device will have. Currently small computing devices are become more readily available and easier to use. One of the most prominent of these devices is the Raspberry Pi. This computing board has already begun it application into computer vision systems and is prices exceptionally well. To successfully make this an independent system it would require a range of different added devices to ensure its stable use. Although this is starting to leave the scope of this project; with it being primarily biased on the software development it still needs to be considered how the system will operate in the real world.

A Raspberry Pi based system is compatible with Matlab Simulink models and should have no problem running the program. However further devices are needed for it to run autonomously. Problems such as power creation and storage can be simply overcome by looking through the massive range of photovoltaic cells and battery possibilities and finding the best fit. This can be said for camera choices as well as networking options. The main focus of this report will be on the computer vision side while showing its potential application and cost. Rather than determining the most effective design.

Risk analysis

The risk associated with the project is extremely small with the majority of the work being done indoors working with computers. One area were possible harm could be achieved is when installing a camera for testing. This being near a road will need to have safety precaution taken.

TASK	HAZARD	RISK	Minimization
1A	NONE	low	Computer lab – Non applicable
1B	NONE	low	Computer lab – Non applicable
2A	NONE	low	Computer lab – Non applicable
2B	NONE	low	Computer lab – Non applicable
3A	NONE	low	Computer lab – Non applicable
3B	NONE	low	Computer lab – Non applicable
3C	NONE	low	Computer lab – Non applicable
4A	NONE	low	Computer lab – Non applicable
4B	NONE	low	Computer lab – Non applicable
4A	NONE	low	Computer lab – Non applicable
5A	NONE	low	Ensure appropriate safety standards are met
6A	Installing Camera around roadway	Medium	1. Have high vision clothing 2. Check for traffic when moving across roadway 3. If installing at height ensure working with partner

Resource Requirements

The resource requirements of this project are very small with the majority of the cost being optional in the case of creating a working prototype. With USQ supplying Matlab software and access to countless research papers the bulk of the research can be completed for little to no cost. In the requirement of a working prototype build from scratch costing's will have to be completed on different cameras and other electronic components. The video to be used in the testing phases can be found and captured in a myriad of ways. However there are specific datasets that need to be used in order to calculate the models effectiveness against other established methods.

Task	Item	Amount	Cost
Phase 2 – Phase 5	Matlab Software	1	$0.00
Phase 5	Raspberry Pi Model B	1	$69.00
Phase 5	Raspberry Pi Camera Board V2	1	$40.47
Phase 5	10400mAH Dual USB Battery	1	$25.15
Phase 5	6 Watt Solar Panel	1	$59.00
Phase 5	Wi-Fi Dongle - Ultra Long Range High Gain w/ 5dBi Antenna	1	$19.55
Total			$213.17

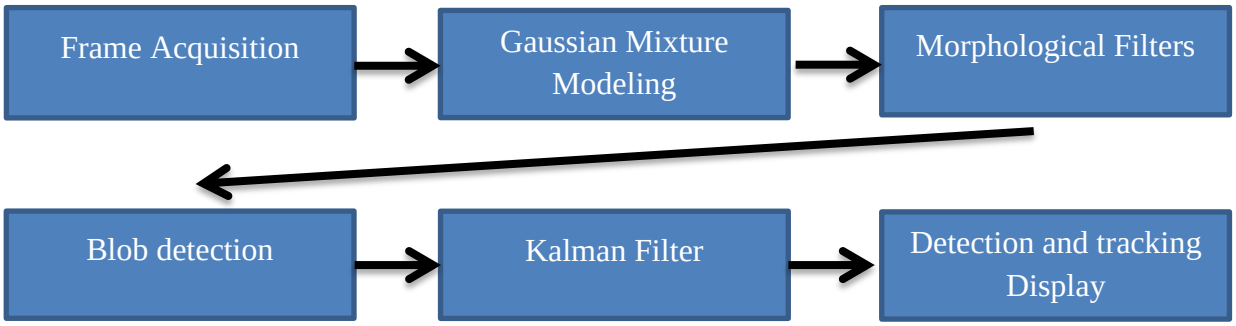
As a rough estimate of the overall cost of building this project components already utilized in the field of surveillance were used. All the components listed have been proven to work in systems and represent what will be needed in the hardware application. However this hardware will be shown as a proof of concept rather than the definitive design with a large amount of testing and development needed on what will make the most effective device.

Development

The primary focus of this report is to effectively design a program that can detect and track pedestrian in a roadway environment. This information can then be used in a range of ways to prevent and analyze accidents and problems with pedestrians and motorists. The choice of Matlab and Simulink was made for its extremely quick prototyping and prior knowledge. The development for this project was done in three iterations all of which are documented. These three show the elements that we focused on and the progression from one to another explaining the reasons for each change. These were tested against multiple environments and there results and performance outlined.

Model design

The process of how the model will work is one that follows a logical design.



Frame Acquisition – Video input separated into frame by frame in order to process changes in pixels.

Gaussian Mixture Modeling – Using the Gaussian modeling to define the foreground and background through the changing in pixels in the comparison of the current and previous frames. Output is a matrix consisting of one and zeros defining where the changes are detected.

Morphological Filters – The process of clearing and removing undesired noise from the frame difference matrix. This has many possible filters each having a specific effect on the overall frame matrix.

Blob detection – Detection of clusters of blobs/ pixels from the frame difference matrix.

Kalman Filter – Applying a Kalman filter to the blob detection to give a weighting and tracking estimates.

Detection and tracking Display – Outputting the detections and tracking into a video format

Iteration 1 – Appendix B

The initial code developed was from a series of tutorials through Matlab central. Running it against PETS DATA SET 2009 shows it currently still needs a large amount of tweaking and progress to get to an effective working model. This initial system utilized a few key features that are already implemented in Matlab. These are some of the crucial elements in this program reducing runtime and making the overall code run much more effectively. This program followed the Model design exceptionally having all the elements required for detection and tracking. Some of the most key functions have been outlined and explained

Detector = vision.ForegroundDetector (Name,Value)

This function utilizes the Gaussian Mixture modeling to successfully differentiate the background and foreground moving targets. This has three major settings that were adjusted to the environments and specific videos tested.

NumGaussians
Number of Gaussian models in the mixture model – Adjusted from range of 3 to 5

NumTrainingFrames
Number of frames given to differentiate between foreground and background. This was adjusted to 10 to 20 frames and is utilized in the beginning of the program to acquire an initial reference for the background.

MinimumBackgroundRatio
This setting was to determine the threshold for which would become the background model. This threshold was changed constantly but left in a range of 0.5 – 0.9.

This initialization of the targeted objects was then run through the next important function of blob analysis. Again a Matlab function was utilized and applied. This function bounds the tracked areas of the detected targets. This information can then be passed onto further analysis such as the Kalman filter.

obj.blobAnalyser = vision.BlobAnalysis

This output of this function as the centroid of the block giving the required points for Kalman tracking. However morphological operations were performed before the final centroid was created. This was done through the range of masks

```
mask = imopen(IM,SE)
mask = imclose(IM,SE)
mask = imfill(IM,SE)
```

These operations worked by inputting what the required type of geometric shape to be applied and that shapes scaling. This can be seen with rectangular fills and closes used at a scaling size dependent on the environment.

From these operations the centroid could be easily defined and passed through the Kalman filter. The Kalman filter worked just as described in the literature review with its inputs being acquired from the centroids of blob analysis.

Figure 11 - Iteration 1 - PETS2009

From this iteration a lot was leant in the limited capabilities of the Kalman filter. The Overall operation of this code showed problems in defining individuals when they get to close together and in groups. Although this is where the Kalman filter was supposed to separate and identify each individual its effeteness was unnoticed. However detection through Gaussian mixture modeling and the morphological filter all operated as expected. Looking at the accuracy in detecting there were a range of small problems that force counting and identification down to roughly 60% with the system only acquiring half of the pedestrians over the 2009 PETS Dataset. This was the best result with a range of different combinations of values for both the Gaussian modeling and the morphological filters.

One needed introduction into this system is the incorporation of regions of interest. In this iteration the program has not defined which areas are important to monitor. This will need to be implemented in order to identify the possible safety hazards for the pedestrian and where they are in relation to them. For instants If they are crossing the road or if they are located in a safe area. This iteration has also encountered another problem in the area of quick prototyping. Without the utilization of Simulink any changes or alterations take a large amount of time needed to be debugged and resolved every time. For this reason Simulink will be applied in the further developments and remaining iterations.

Iteration 2

This iteration is where the transfer was made to Simulink block based coding perfect for rapid and experimental testing and design. The process however remains the same as before with all the processing conducted in the code transferred into block based interactions. This made conducting large changes and addition extremely easy being able to manipulate huge portions of the code without large amount of debugging. From this iteration the regions of interest were beginning to be implemented this meant segmenting the desired locations and areas for further analysis such as counting and estimating. For testing with this new iteration a range of improvements were made over iteration 1.

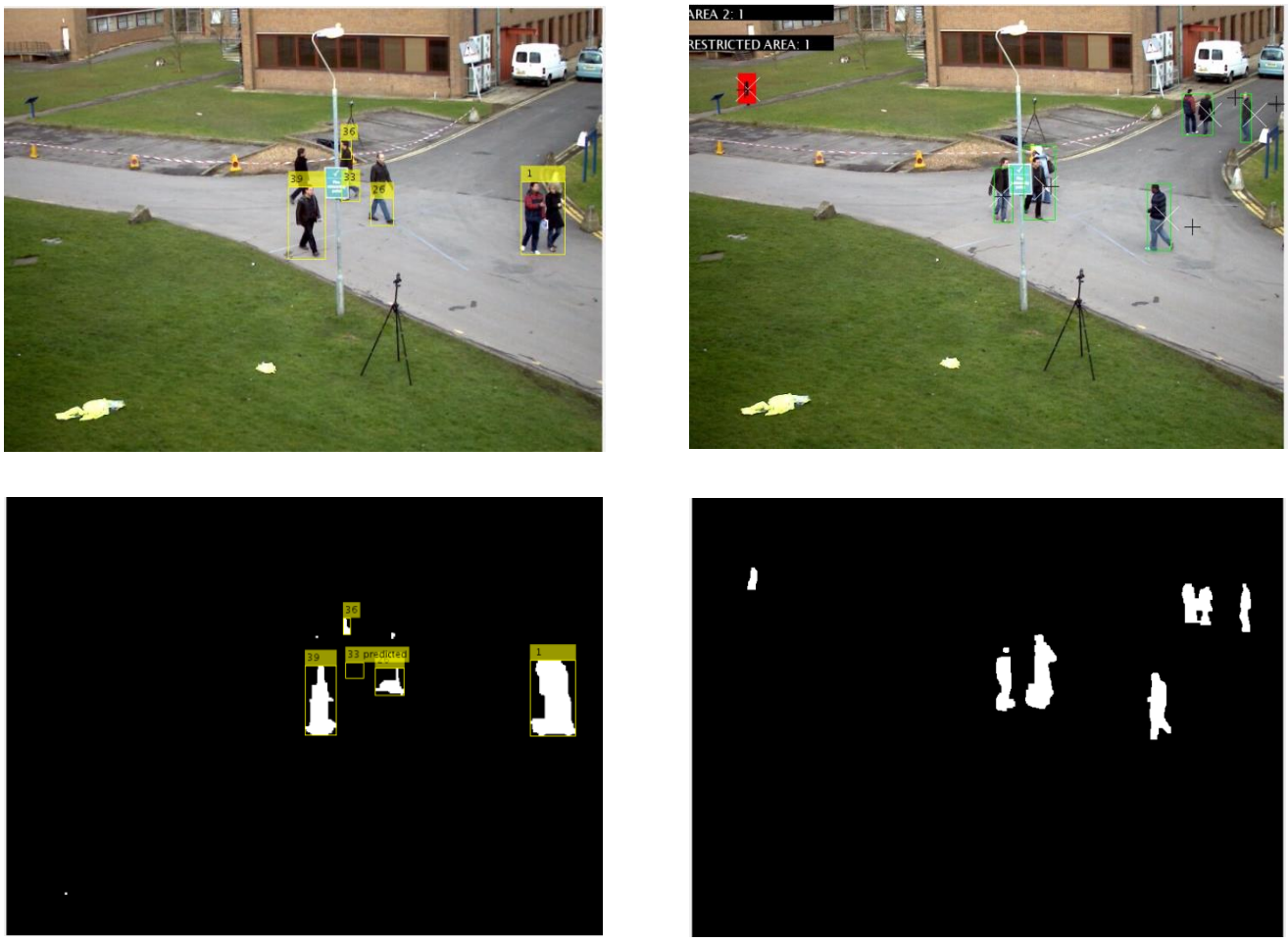


Figure 12 - Comparison of Iteration 1 and Iteration 2

Clarity in detection and noise removal were significantly better. With the much quicker and more effective Simulink model optimal designs and combinations were much easier to implement. This as well as the region of interest implementation made the model much more useful in application. Its design can be seen here:

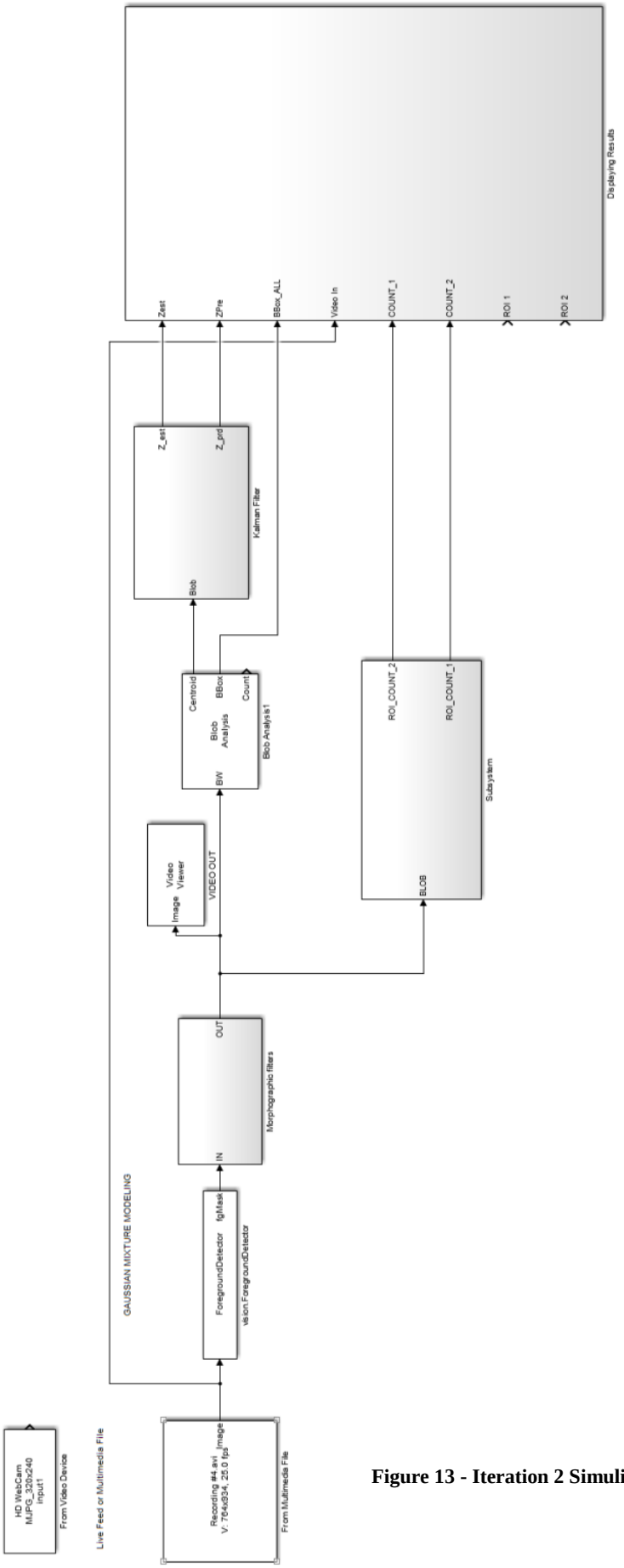


Figure 13 - Iteration 2 Simulink Model

The model broken up into its separate parts shows the process of how the blocks interacted to create the detection and final tracking output.

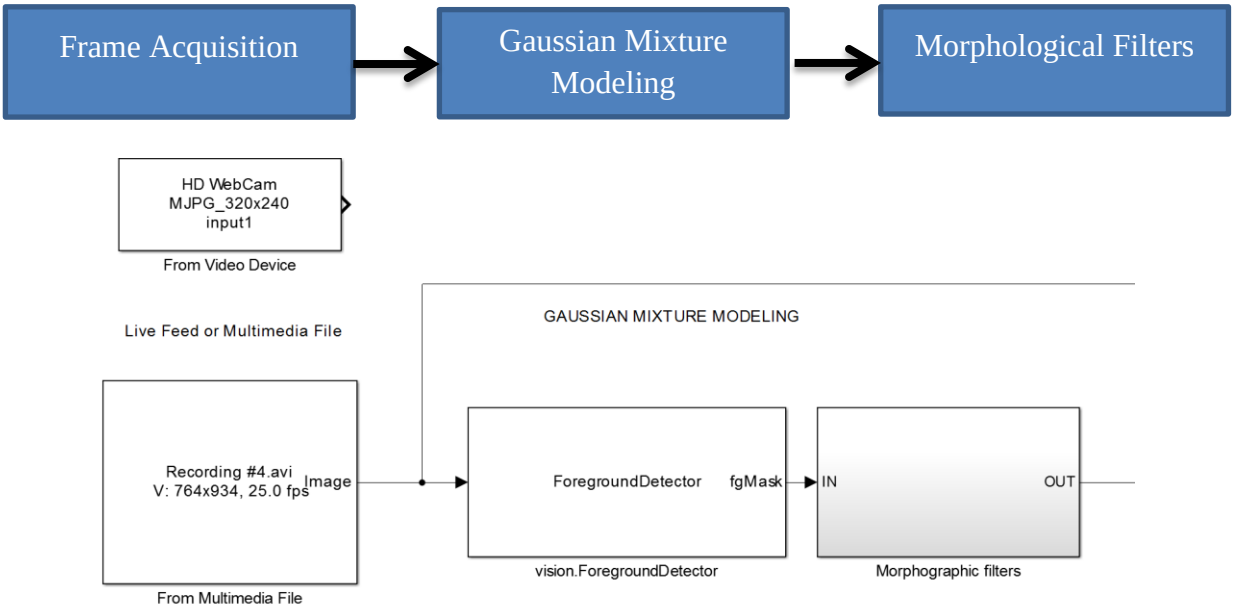


Figure 14 - GMM and Morphological Filter Simulink Model

The initial few steps were extremely simple especially the Gaussian models having a pre-defined function block shown in Appendix C. The output of this block is the pixel matrix then goes through the morphological altering and removing the noise of the matrix.

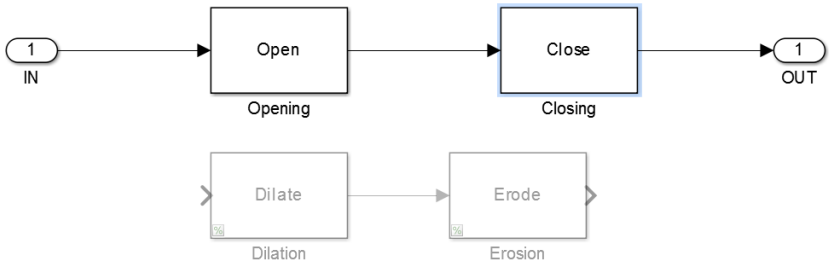


Figure 15 - Morphological filters subsystem

This subsystem is where a large amount of experimenting and fine tuning takes place with each different environment and tracking targets having a massive effect on the levels of how much each filter will effect and improve the image. The goal of the Gaussian model and the filters is to create a noise free environment highlighting the specific entities that can then be used in further analysis.

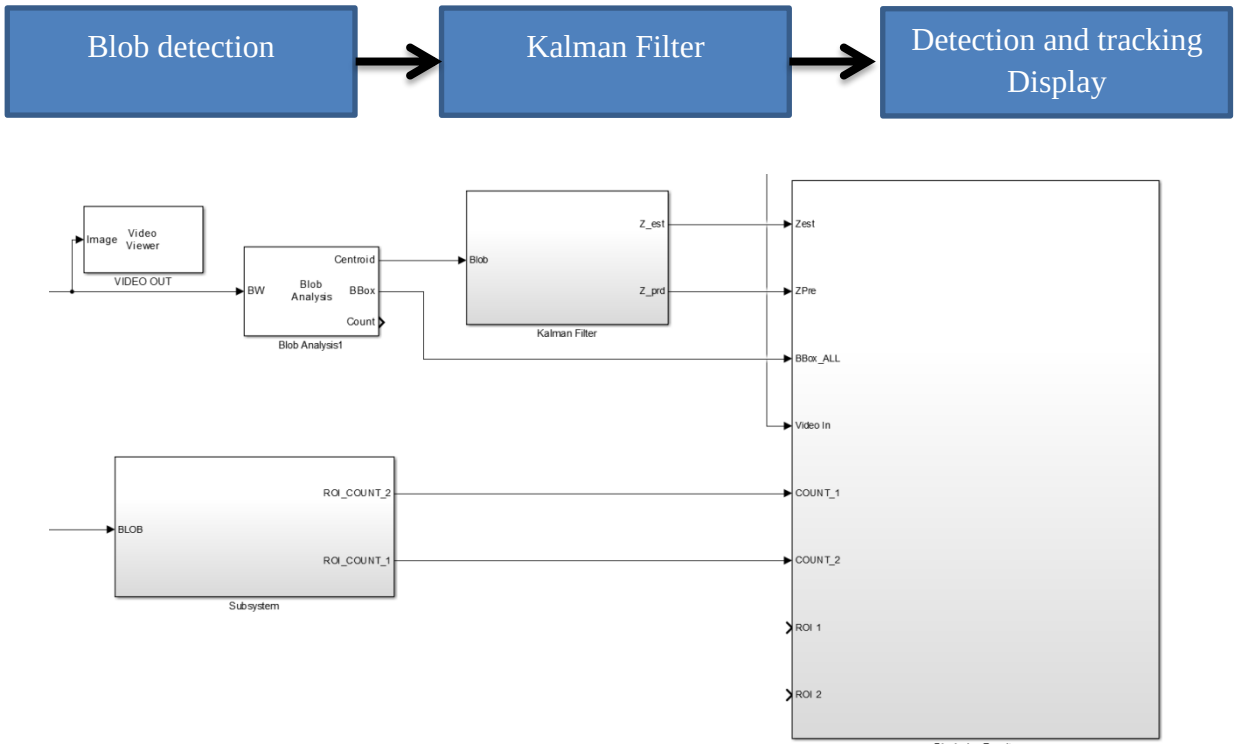


Figure 16 - Blob analysis, Kalman Filter and Display Simulink Model

The blob detection is conducted through the block function blob analysis this block has a range of different setting and output that can be used in tracking with the Kalman filter.

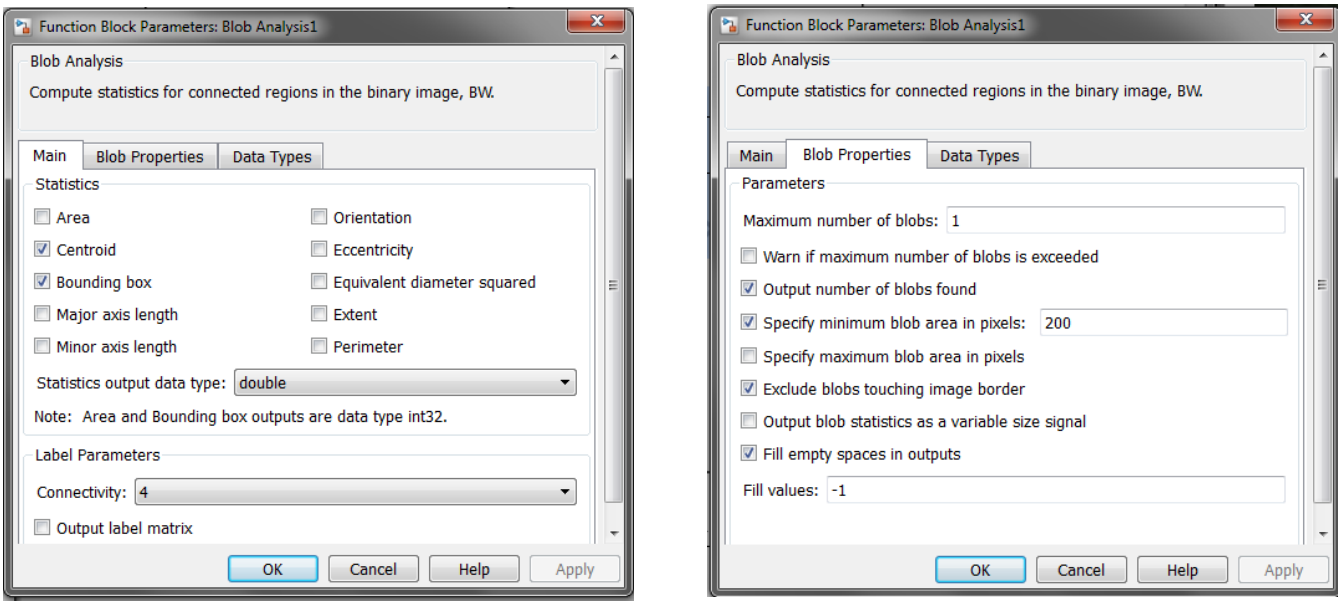


Figure 17 - Blob Analysis settings

The import of these outputs is making sure the centroid of the tracked blob is outputted and transmitted into the Kalman filtering process.

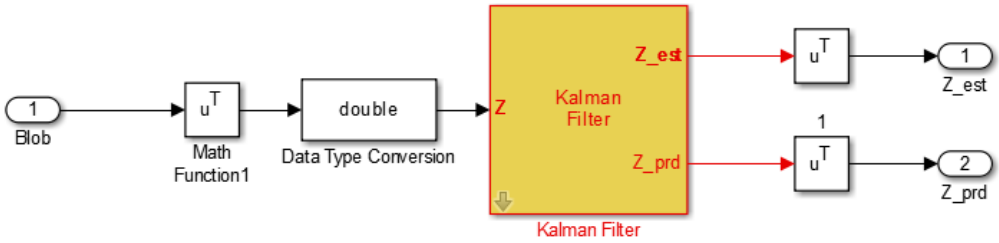


Figure 18 - Kalman Filter subsystem

The Kalman filter was conducted two different ways. One was through the predesigned block available through Mathworks Simulink and the other was a function based block with the individual formulas inputted. From testing the most effective block was the Mathworks predesigned block. Huge problems of efficiency and tracking began immerging with the implementation of the function block. The code shown in Appendix B had massive problems in implementation. Both blocks had the same problem of blob association with them not being able to successfully identify which tracking target they were assigned to. Not only this but the function implemented block saw massive performance drop making the predesigned block the most effective choice.

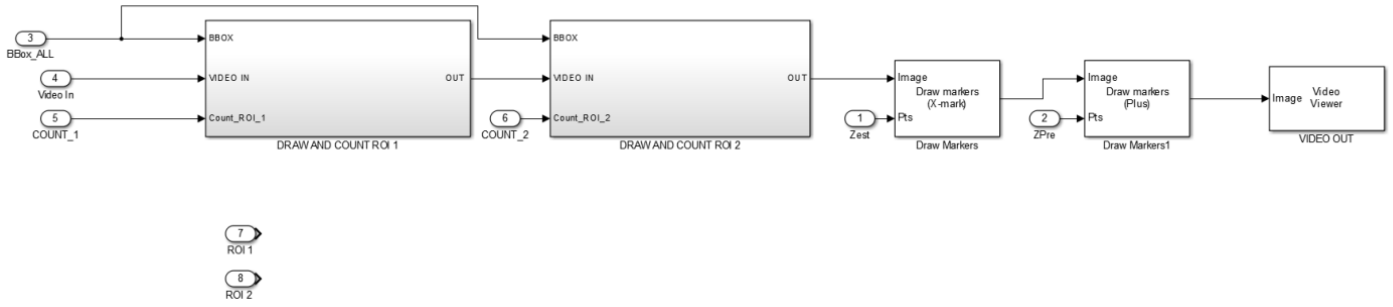


Figure 19 - Display Subsystem

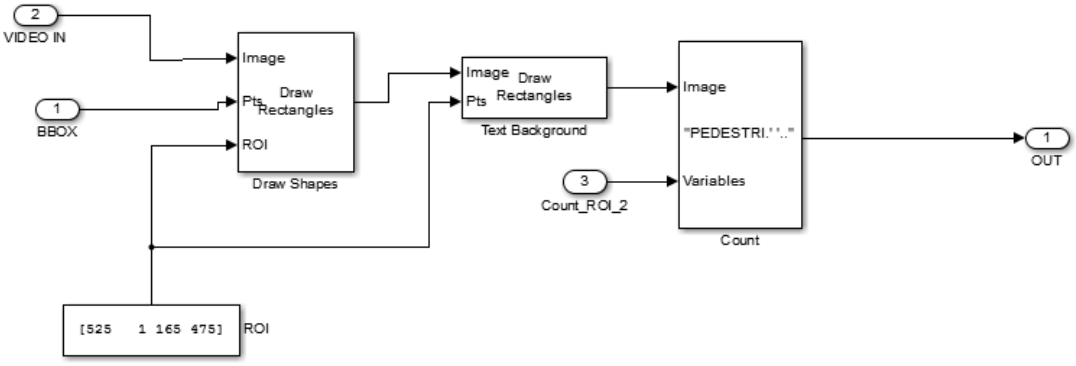


Figure 20 - Draw and ROI Subsystem

Displaying the information was done through a series of drawing and marker blocks. This created the box identifying what are the target was in, its centroid and what the Kalman filter prediction. The rectangular outline identifies the objects size and what designated area it is located in by the color change. This is also fed into a counter in the top left corner monitoring the numbers of targets in each region of interest. Added to this is the identification of the centroid as a white 'X' and the Kalman filter prediction as a black **'+'**.

To test this model a video was run though of a person walking up to a fence. The goal was to see if the unit can be identified its position and how many units were there. The detection output a green box when the target was a one zone to red when it became in proximity to the fence.

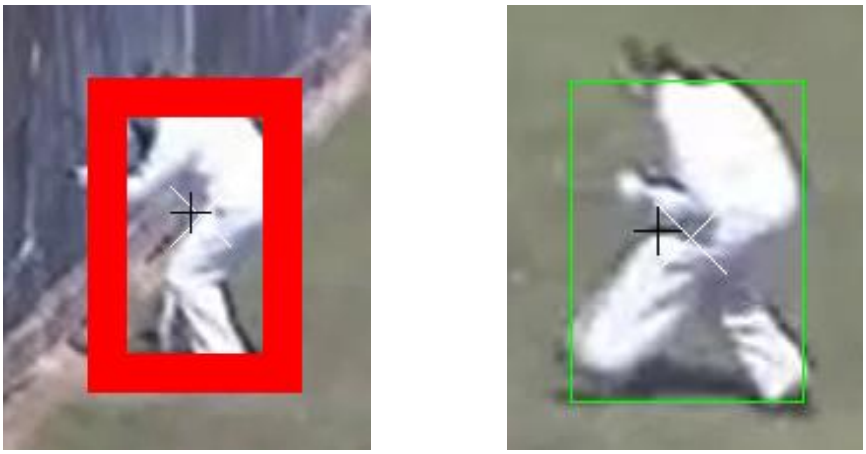


Figure 21 - Iteration 2 test - Fence

This was successful and could easily track and identify where units are in relation to the regions of interest. Results from this iteration showed a serious improvement in the accuracy of both the detection and the Kalman filter tracking. However a problem emerging is the difficulty in associating Kalman filter tracks to the blob analysis. Meaning that utilizing the Kalman filter for counting is extremely difficult with it dropping in and out creating an array of wrong counting numbers. However the detection side is working extremely well with the incorporation of the regions of interest. This has let the system identify two separate area both of which can detect and count how many targets are in each.

Looking at the accuracy of the system it is reaching levels of 80%. Running it through the 2009 PETS dataset showed a massive increase in detection of close entities than in iteration one. All round improvements have been made however the Kalman filter is still having trouble in association and tracking. The next stage of this development will be in applying this system to a roadway and monitoring its overall effectiveness.

Final Design

The final design was applying heavy improvement to iteration 2 in order for it to be capable to run in a roadway situation. This meant applying multiple regions of interest as well as giving it the ability to distinguish between pedestrians and automobiles. This was done by setting ranges of pixel sizes of the tracked entities. This let the differentiation of cars and when they enter the designated crossings.

Many of the central functions of the code have not changed from iteration two with the Gaussian mixture modeling shown in appendix B being utilized again as well as the range of morphological filters just in different values for the changes in environment.

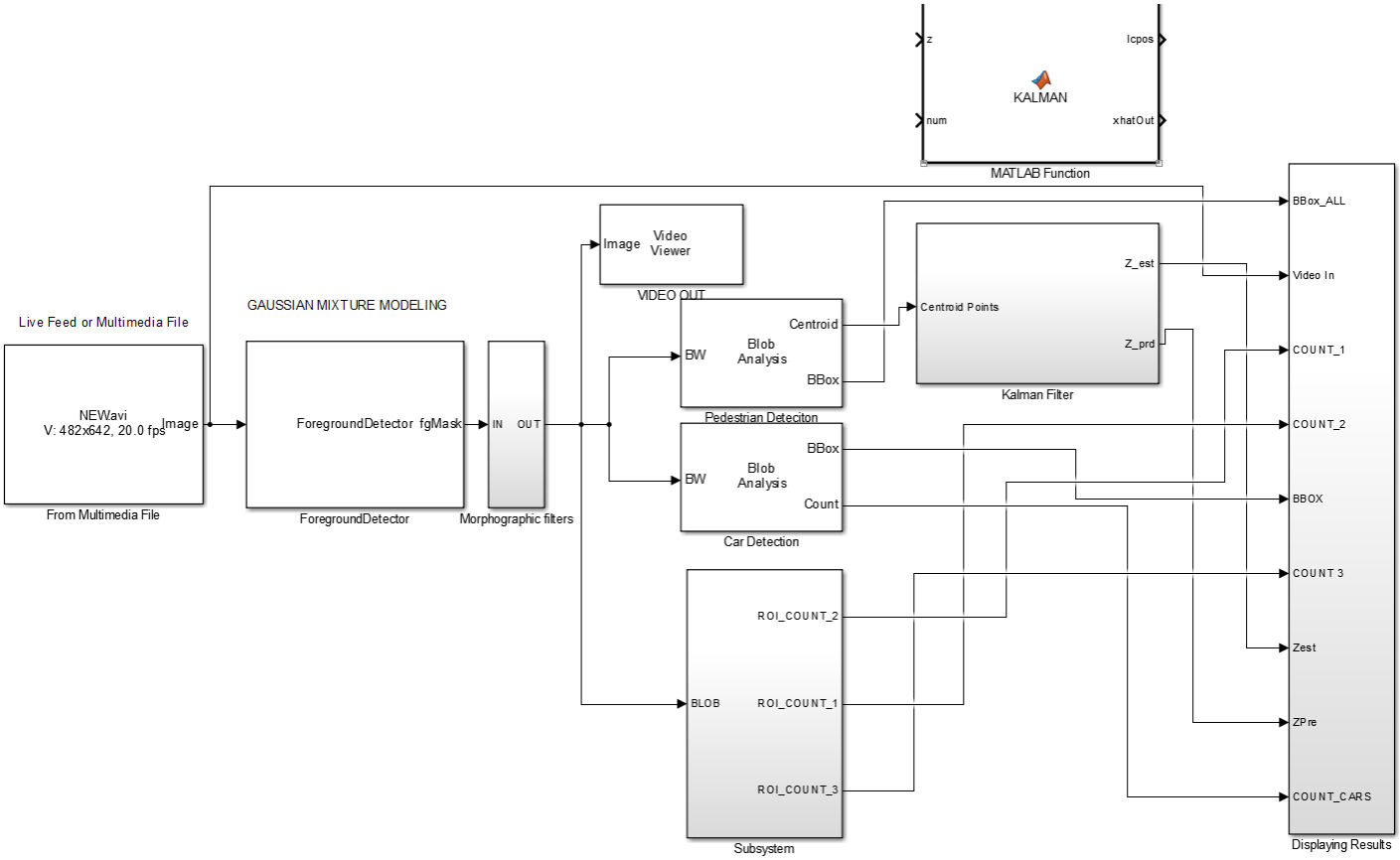


Figure 22 - Final Iteration Simulink Model

Testing this iteration against the PETS 2009 dataset showed results exactly the same as iteration two as expected due to the same overall methods being utilized. The results were showing an accuracy of around 80% again however the reduction of close targets does increase that accuracy. Determining an overall accuracy is extremely difficult because of the due to the changing environment requiring heavy alteration towards the Gaussian modeling and the morphological filters. However the systems accuracy increases substantially when it is utilized in

environments with low noise and clear concise moving patterns from the targets. Utilizing it in highly crowed and dynamic environment showed a massive drop in the accuracy of the system.

Initial testing of the system it was run though a roadway situation of crossing pedestrians through a road way. The success of this test would be if the program could count and identify not only where the pedestrians were but also when a car enters the frame. To do this the environment was divided into its three respected zones. First zone is designated safe with the pedestrians using the sidewalk with a safe distance from the road shown as green. The second was the warning zone outlining any area that is getting to close to the road designated yellow. Finally is the danger zone where possible accidents between motorist and pedestrians may occur shown as red.

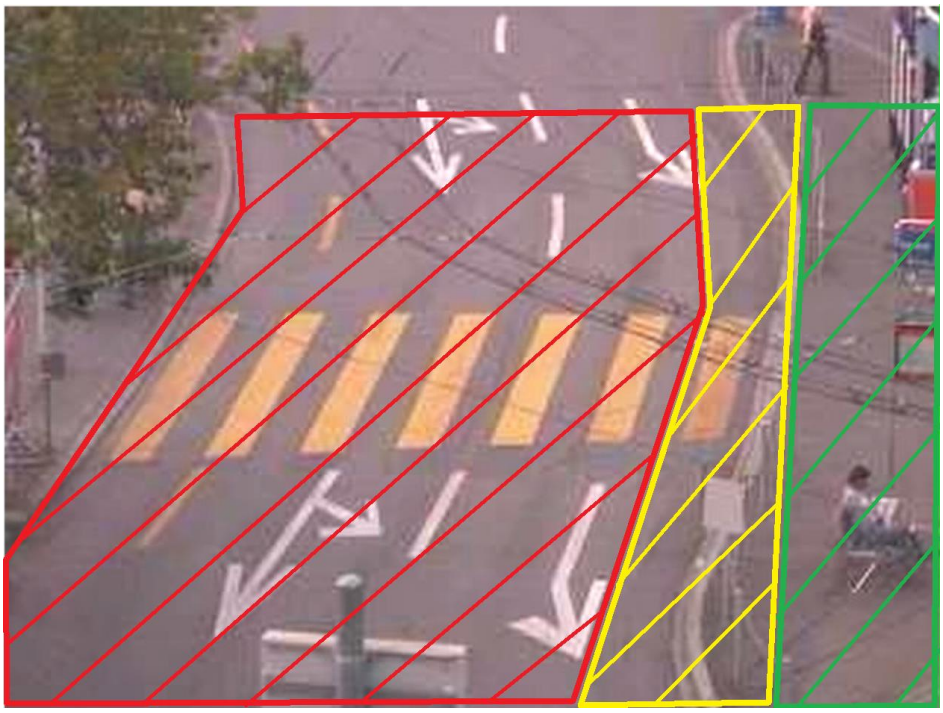


Figure 23 - Zoning Roadway Test

The count of how many pedestrian in each is calculated as well as when a car enters the frame. This count can then be used in both instant signaling such as digital signage as well simple counting analytics for future reference. Testing this scenario showed very promising results. Detection of all pedestrians was acquired as well as the tracking from zone to zone. Counting was consistent successfully numbering how many people are in are zone as well as the instant detection when car enters the frame. Looking at the testing it was an overall success.

Figure 24 - Roadway Test - 1

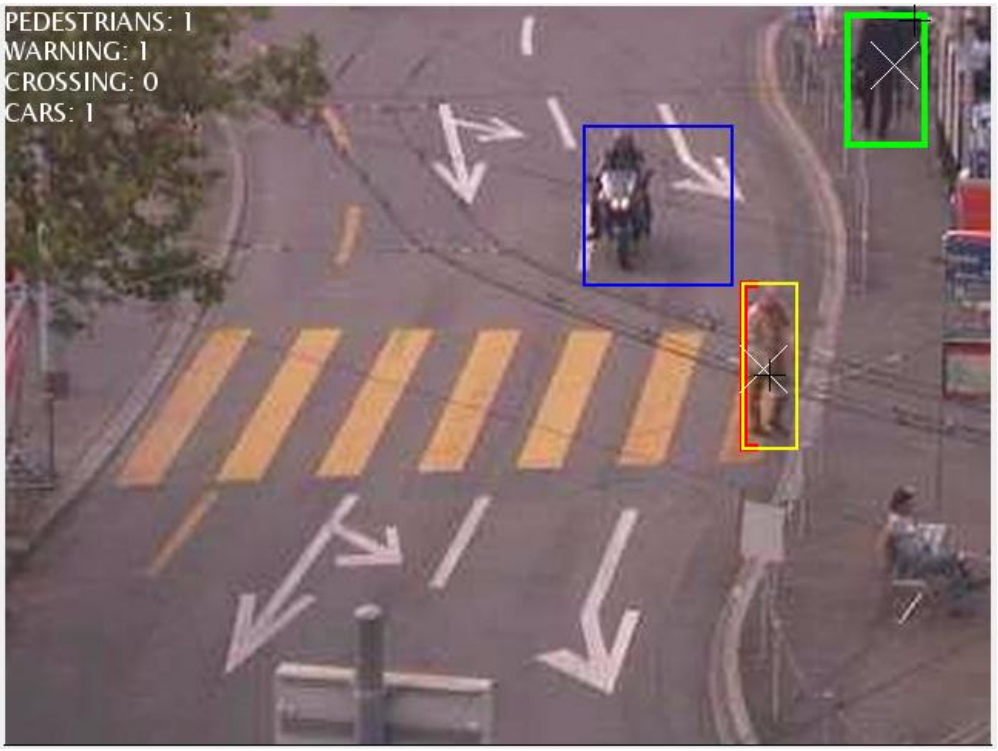


Figure 25 - Roadway Test - 2

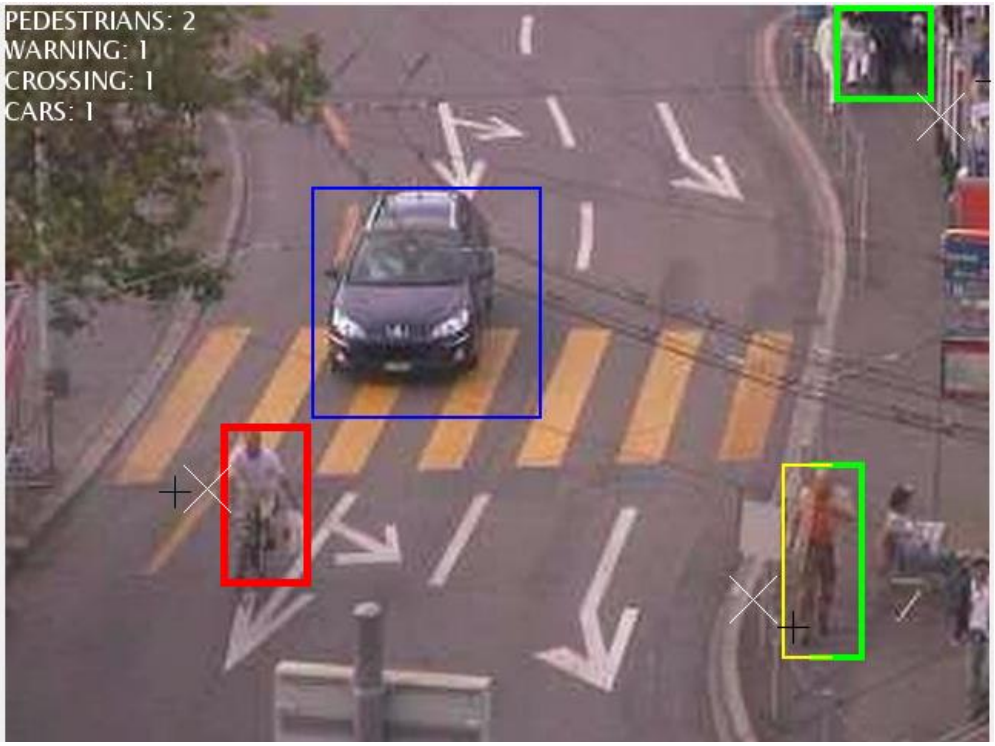


Figure 26 - Roadway Test - 3

The added features in the finial iteration made the system a worthwhile addition for roadways to improve safety. The system successfully identified the position s of each pedestrian as well as when a car enter the environment. However problems with the Kalman tracking and blob association still persist with the Kalman filter not being able to update the tracks when collision and divergence occur. Other problems such as defining bicycles and motorcycles are also shown with them being categorized into either cars or pedestrians.

Results

The results from the three iterations showed a gradual improvement in the systems capabilities moving from just basic detection to system that can identify between cars and pedestrians. Not only this but the final iterations ability to detect various regions of interest lets the system track when a pedestrian enters a dangerous area. The three iterations can really be reduced down to the code based model and the Simulink model. The Simulink model ended up being the most effective due to its extremely easy to manipulate and prototype friendly format. This let additions such as the multiple regions of interest and counting systems to be implemented extremely effortlessly. However both systems had major issues in the implementation of the Kalman filter. In regard to computing cost it was extremely low having both models operating at a live footage speed. This can obviously be manipulated with the introduction of larger video formats over the chosen small resolutions.

Looking at the accuracy levels both were tested against the PETS 2009 dataset. This data set has been used extensively as a common comparison in a range of different reports and was a solid challenge for testing. With both these systems utilizing the same essential function in each there is very little difference in the overall performance. With the same setting in the Gaussian Mixture Modeling, morphological filters and blob analysis results from the two were almost identical as expected. Running the codes through a range of other environments however showed that the Simulink model was significantly easier to adapt.

In the case of running the system in a roadway scenario the model showed great promise. The model performed extremely well detecting all pedestrians as well as differentiating between automobiles. This in conjunction with the ability to outline dangerous areas and possible hazards makes the system an overall success. However its application needs to be applied into significantly more environments. High clutter environments with significantly increased numbers of pedestrians moving through crosswalks need to be investigated, determining if the system can uphold its accuracy. Although the system performed well in this specific environment it needs to be run for an extended amount of time over a range of different scenarios.

Looking at how the filters were implemented shows the strength and weaknesses of the system. The Gaussian Mixture Modeling was implemented through a function block shown in appendix B. This was implemented extremely well performing all the require modeling and background subtraction. This stable detection flowing into the morphological filter made the overall detection of the targets extremely effective. The blob analysis was all conducted with ranging pixel limits differentiating between the cars and the pedestrians worked well enough. However any sized object in-between can be thrown into a group it does not belong such as monocycles. This also forces the system to be calibrated in every new environment with the sizing of pedestrians and their pixel sizes changing depending on the application. Furthermore any movement of the camera will distort the calculated regions of interest.

Future Work

The area of computer vision is a massive expanse of rapidly improving technologies and methods. The current model designed throughout this report has many positive attributes however before it can be capable enough to be applied in real world scenarios huge improvements need to be made. The biggest flaw and letdown throughout this project was the implementation of the Kalman filter. This had serious problems in outputting concise and useable data. Although it was run over the model its poor results throughout made it extremely hard to be given any actual application. This will need to be investigated heavily as track identification is necessary in high crowded and noisy environment.

The testing of this model was done over a small range of well lighted noise free environments. For this system to be even remotely close for real application it must be tested in a range of not only roadway scenarios but also weather and lighting conditions. Investigation must be made in how much lighting is needed for the model and if I can operate with simple installation on a light post. Not only this but what will be the most effect viewing angle, high considerations and data retrieval all need to be considered. This requires a lot of analysis into how the system will be mounted and all the environmental effects that it may have to endure. This would require a massive review of available hardware and how each of them could be applied and there overall performance. A quick estimate of costing was conducted however increase scrutiny and actual application would be require for the system to be considered ready.

The possibly of cheap counting systems could be utilized in a range of different areas. However massive hurdle will need to be overcome in order for it useful. Data retrieval would a huge part of this problem in how would the counted and calculated data get to the sources it wants. An interconnected network of cheap counting system could make huge advancements in city planning as well a range of different areas. This can also be said for cheap security surveillance systems being able to successfully identify and notify when targets have entered restricted areas.

The possible future applications for a cheap computer vision analysis system are massive. The shear amount of future work and investigation that circles computer vision makes it a topic that has near endless application. The choice of application to road safety is one that has already begun and will continue to advance as the technologies and methods improve. The current model detects tracks and identifies when and where pedestrians are and if they are in dangerous circumstances. With improvements on the Kalman filter and hardware investigation this system could reach a point where it becomes common in roadway safety.

Conclusion

The goal of this dissertation was to successfully design a roadway safety system that could be applied and utilized in a range of different scenarios. The examination into current technologies and algorithms showed the myriad of possible ways and solutions on how to approach this problem. Through testing and prototyping a model was designed that could detect, track and identify when pedestrians and motorists enter possible dangerous situations. Background subtraction was conducted though a Gaussian Mixture model and performed extremely well creating well defined entities to track. With the addition of a range of morphological filters the resulting detection was to an extremely high standard. This detection was then processed through a Kalman Filter which then predicted and tracked the required blobs. The overall operation of the system was run through few different scenarios most notably a roadway with pedestrian and motorist mixing. The system was successfully able to identify pedestrians from cars and the location of those pedestrians in relation to the roadway. This information can then be applied to a range of system to ensure notification to the motorist or display warning to the pedestrians. This is capable due to the system's ability to be run of live video feeds.

The possible application of this system in not only roadway safety but in other industries such as home security or data analytics is also apparent. With simple changes in defining the regions of interest and pixel size ranges the system can be applied to almost any situations requiring detection. The ability to easily count pedestrian numbers over long periods without human intervention also applies itself to a range of different areas. However further work will need in applying the system to new and challenging environments to successfully trial the system. This type of investigation will also be needed in the development in hardware capable of house and running the models.

In conclusion the overall development of this system was a success. The model could detect, differentiate and track the pedestrians in situations that would be commonly encountered in roadways. Although not all aspects of the model worked effectively; such as the Kalman filter the system was operating as it was intended. The application of computer vision systems towards roadways safety is inevitable with the rapid advancement and effectiveness of these systems there uses and benefits are extensive. This report was dedicated to investigating and applying current methods and techniques that could benefit pedestrians, in this regard it has successfully represented the possible uses and application of such a model.

Appendices

Appendix A – Project Specification

Project Specification

For: Russell Birkett

Title: Video road processing road safety system

Major: Mechanical Engineering

Supervisors: Toby Low

Enrolment: ENG4111 ONC S1, 2016 ENG4111 ONC S1, 2016

Project aim: To develop and create robust video processing software that can be run through road footage to detect entities and obstructions on the roads that could be subsequently avoided.

Programme: Issue A, 16th March 2016

The project work and implementation have been broken down into

- Startup Phase – gathering research, resources and information based around the project
- Creation Phase – apply this knowledge to developing a working code that can be applied to simple environments
- Enhancement Phase – from the rough working model developed start applying to more difficult scenarios improving performance
- Comparison Phase – compared designed code against already developed models and advance and improve current code
- Testing – Test in a real world environment
- Write up – Dissertation

Phase 1	**Startup Phase**
1A	Resource Check – Gather additional information on how other projects attempted the problem
1B	Comparison Check – Find key elements throughout predesigned projects and make links in how they applied there algorithm

Phase 2	**Creation Phase**
2A	Development - Create an outline of the needed elements gathered from research
2B	Implement – Being adding and creating the required sections shown in other work
Phase 3	**Enhancement Phase**
3A	Improvement – with a basic working program begin running through different scenarios and environments
3B	Analyze – Being noting what environment/conditions work best and worst
3C	Compare check – Check back through research for solutions and example of how to improve/ overcome problems
Phase 4	**Comparison Phase**
4A	Testing - Compare research code against current and not differences
4B	Analyze data - find how each code works and how similar technique s can be applied
Phase 5	**Testing**
5A	Test –Test current program in a real world situation
Phase 6	**Prototyping**
6A	Design - Design a possible system that can operate on the program
Phase 7	**Write Up**
7	Dissertation – Summarize all gather knowledge and development processes

RESOURCE REQUIREMENTS

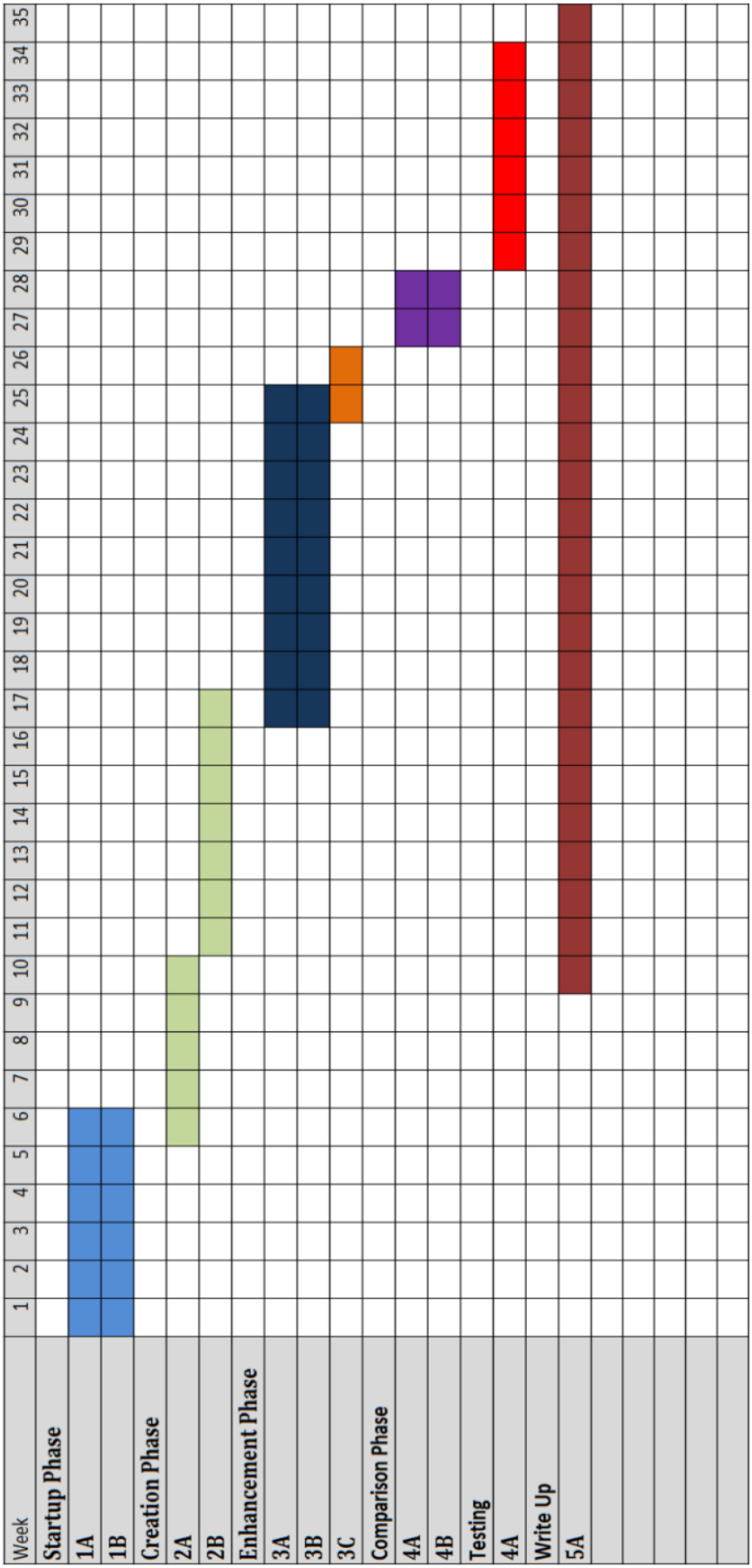
The resource requirements of this project are very small with the majority of the cost being optional in the case of creating a working prototype. With USQ supplying Matlab software and access to countless research papers the bulk of the research can be completed for little to no cost. In the requirement of a working prototype build from scratch costing's will have to be completed on different cameras and other electronic components.

Task	Item	Amount	Cost
Phase 2 – Phase 5	Matlab Software	1	NIL
Phase 5	Camera	1	UNKNOWN

RISK ANALYSIS

The risk associated with the project is extremely small with the majority of the work being done indoors working with computers. One area were possible harm could be achieved is when installing a camera for testing. This being near a road will need to have safety precaution taken.

TASK	HAZARD	RISK	Minimization
1A	NONE	low	Computer lab – Non applicable
1B	NONE	low	Computer lab – Non applicable
2A	NONE	low	Computer lab – Non applicable
2B	NONE	low	Computer lab – Non applicable
3A	NONE	low	Computer lab – Non applicable
3B	NONE	low	Computer lab – Non applicable
3C	NONE	low	Computer lab – Non applicable
4A	NONE	low	Computer lab – Non applicable
4B	NONE	low	Computer lab – Non applicable
4A	NONE	low	Computer lab – Non applicable
5A	Installing Camera around roadway	Medium	1. Have high vision clothing 2. Check for traffic when moving across roadway 3. If installing at height ensure working with partner


Week
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
Startup Phase
1A
1B
Creation Phase
2A
2B
Enhancement Phase
3A
3B
3C
Comparison Phase
4A
4B
Testing
4A
Write Up
5A

Appendix B – Kalman Filter (Semko, 2007)

```
%kalman_10
%The following code processes the output measurements from the Optical
%Flow Analysis block (z) and the number of contacts (num) at the output
%of the same block and performs a Kalman Filter Tracking operation on
%each measurement to determine an estimated position (xhatOut). The
%code also monitors changes in the number of contacts and produces the
%last known location of a contact (lcpos) if a contact is lost.
function [lcpos, xhatOut]=KALMAN(z,num)
%The following variables are set as persistent in order to make them
%available frame after frame.
persistent XHAT
persistent PP
persistent lastmed
persistent numbin
persistent P
persistent index
persistent XHATtemp
%Initial conditions for various variables are set here for use throughout
%the embedded MATLAB function
if isempty(PP)
 XHAT = 0.1*ones(6,10); %Estimated pos./vel/accel.
 XHAT(1,:)=144*ones(1,10); %Estimated pos. (row)
 XHAT(2,:)=192*ones(1,10); %Estimated pos. (column)
 pp = diag([9 9 25 25 49 49]); %Initial P-covariance matrix (6x6)
 PP=[pp pp pp pp pp pp pp pp pp pp];%10 P matrices side by side
 numbin=num*ones(25,1); %Bin tracks num over succ.frames
 lastmed=num; %Bin median
 P=pp; %Initial value for P
 index=[1:10]'; %Index vector for filter assignment
 XHATtemp=zeros(6,10); %Temporary estimated pos.
end
%Numbin is continually updated to include information from the current
%frame and 24 previous frames. Median value for number of contacts (num)
%is calculated as mednum.
numbin(1:24,1)=numbin(2:25,1);
numbin(25,1)=num;
mednum=median(numbin);
64
%Each measurement (z) is compared to the position estimates which have
%remained persistent from last frame. Each measurement is paired with
%its closest position estimate and the index of said estimate is recorded.
%Two “for loops” are used to ensure that no two successive measurements are
%matched with the same filter and position estimate. The first loop
%calculates the index for the first measurement and the second calculates
%it for all subsequent measurements.
if mednum~=num
 normMIN=10e10;
 for j=1:lastmed
 normCALC=norm(z(:,1)-XHAT(1:2,j));
 if normCALC<normMIN
 index(1,1)=j;
 normMIN=normCALC-eps;
 end
 end
```

```
 for k=2:num
 normMIN=10e10;
 for j=1:lastmed
 normCALC=norm(z(:,k)-XHAT(1:2,j));
 if normCALC<normMIN && index(k-1)~=j
 index(k,1)=j;
 normMIN=normCALC-eps;
 end
 end
 end
end
%The measurements are adjusted here to account for the possibility that
%a contact was lost or gained for less than 13 frames. Any measurement
%that is missing is replaced with the position estimate from the previous
%frame.
ztemp=XHAT(1:2,:);
for i=1:num
 ztemp(:,index(i,1))=z(:,i);
end
z=ztemp;
65
%If lastmed is greater than mednum, then a contact has been missing for
%13 out of the past 25 frames and the contact is deemed lost. The index
%of that contact is determined and subsequently the position corresponding
%to that index is sent to the post-processing block. The filter which
%corresponded to the lost contact is re-set to the initial conditions.
%*If LCindex is less than or equal to zero then there is an error in the
%index assignments and no lost contact information is sent to the post-
%processing block.
if lastmed>mednum
 sumSOME=0;
 sumALL=0;
 for i=1:mednum
 sumSOME=sumSOME+index(i,1);
 end
 for j=1:lastmed
 sumALL=sumALL+j;
 end
 LCindex=sumALL-sumSOME;
 if LCindex>0
 LCpos=XHAT(1:2,LCindex);
 for i=1:10
 XHATtemp(:,i)=XHAT(:,index(i,1));
 end
 XHAT=XHATtemp;
 XHAT(:,num+1)=[144;192; 0.1; 0.1; 0.1; 0.1];
 index=[1:10]';
 else
 LCpos=[0;0];
 end
else
 LCindex=0;
 LCpos=[0;0];
end
%lastmed is updated for use in the processing of the next frame of video.
lastmed=mednum;
```

```
66
%The Kalman filter is implemented based on the number of adjusted data
%points that are present at the input of the block. Each data point is
%processed separately and filterindex is used to index the correct pos.
%estimates and covariance matrices.
for i=1:mednum
 if index(i,1)>0
 filterindex=index(i,1);
 else
 filterindex=10;
 end
 xhat=XHAT(:,filterindex);
 low=filterindex*6-5;
 P(:,1)=PP(:,low);
 P(:,2)=PP(:,low+1);
 P(:,3)=PP(:,low+2);
 P(:,4)=PP(:,low+3);
 P(:,5)=PP(:,low+4);
 P(:,6)=PP(:,low+5);

 % 1. Compute Phi, H, Q, and R
 Phi = [1 0 1 0 0.5 0; 0 1 0 1 0 0.5; 0 0 1 0 1 0; 0 0 0 1 0 1; 0 0 0 0 1 0; 0 0 0 0 0 1];
 H=zeros(2,6);
H(1:2,1:2)=diag(ones(1,2));H(1:2,3:4)=diag(ones(1,2));H(1:2,5:6)=0.5*diag(ones(1,2));
 Q =diag(0.5*ones(1,6));
 R =diag(ones(1,2));
 %2. Compute Kalman Gain and Update State Covariance
 K=P*H'*((H*P*H'+R)^-1);
 P = (eye(6,6)+K*H)*P;
 % 3. Propagate the track estimate::
 xhat = xhat+K*(z(:,i)-H*xhat);
 xhat=Phi*xhat;

 % 4. Update Covariance Matrix
 P = Phi*P*Phi'+Q;

 % 5. Refill filter bank
 XHAT(:,filterindex)=xhat;
 PP(:,low)=P(:,1);
 PP(:,low+1)=P(:,2);
 PP(:,low+2)=P(:,3);
 PP(:,low+3)=P(:,4);
 PP(:,low+4)=P(:,5);
 PP(:,low+5)=P(:,6);

end
67
%Updated position estimates and lost contact position (if any) are sent to
%the post-processor.
lcpos=LCpos;
xhatOut=XHAT(1:2,:);
```

Appendix C – ForgroundDetector (MathWorks, 2010)

```
classdef ForegroundDetector < matlab.System
   properties(Nontunable)
      NumGaussians = 5;
      MinimumBackgroundRatio = 0.7;
      InitialVariance = 'Auto';
   end

   properties (Nontunable, Logical)
      AdaptLearningRate = true;
   end

   properties (Nontunable)
      NumTrainingFrames = 150;
   end

   properties
      LearningRate = 0.005;
   end

   properties(Constant, Hidden, Nontunable)
      VarianceThreshold = 2.5*2.5;
      InitialWeight = 0.05;
   end

   properties(Access=private)
      Time;
      Weights;
      Variances;
      Means;
   end

   properties(Access=private, Nontunable)
      ClassToUse;
      FrameSize;
      NumChannels;
      pInitialVariance;
   end

   properties(Access=private, Hidden, Nontunable)
      ImageClass=coder.internal.const('double'); %only for codegen
      StatClass =coder.internal.const('double');
   end

   properties(Access=private, Hidden, Logical)
      hasConstructed = false; %only for codegen
   end

   methods(Access=private)
      function initializeStates(obj, classToUse)
         obj.Time = 0;
         numPixels = prod(obj.FrameSize);
         obj.Weights = zeros([obj.NumGaussians numPixels], classToUse);
```

```
        obj.Variances = obj.pInitialVariance *...
            ones([obj.NumChannels, obj.NumGaussians, numPixels], ...
                classToUse);

        obj.Means = zeros([obj.NumChannels, obj.NumGaussians, numPixels], ...
                    classToUse);
    end

    function initializeParameters(obj, I)
        inputSize = size(I);
        obj.FrameSize = inputSize(1:2);
        if length(inputSize)<3
            obj.NumChannels = 1;
        else
            obj.NumChannels = inputSize(3);
        end

        initInitialVariance(obj, I);

    end

    function initInitialVariance(obj, I)
        if strcmpi(obj.InitialVariance,'Auto')
            if isfloat(I)
                obj.pInitialVariance = (30/255)^2;
            else
                obj.pInitialVariance = 30^2;
            end
        else
            obj.pInitialVariance = obj.InitialVariance;
        end
    end
end

properties (Transient,Access = private)
    pForegroundDetector = [];
end

methods
    function obj = ForegroundDetector(varargin)
        coder.allowpcode('plain');
        setProperties(obj, nargin, varargin{:});
        if isempty(coder.target)
            obj.pForegroundDetector = vision.internal.ForegroundDetector();
        else
            obj.hasConstructed = false;

        end
    end

    function set.AdaptLearningRate(obj, value)
        validateattributes( value, { 'logical' }, ...
                        { 'finite', 'scalar', 'nonsparse', 'real' },...
                        'ForegroundDetector', 'AdaptLearningRate');
```

```
    obj.AdaptLearningRate = value;
end

function set.NumTrainingFrames(obj, value)
    validateattributes( value, { 'numeric' }, ...
                    { 'scalar', 'real', 'integer', 'positive', 'nonsparse' },...
                    'ForegroundDetector', 'NumTrainingFrames');

    obj.NumTrainingFrames = value;
end

function set.LearningRate(obj, value)
    validateattributes( value, { 'double', 'single' }, ...
                    { 'finite', 'scalar', '>', 0, '<=', 1, 'nonsparse', 'real' }, ...
                    'ForegroundDetector', 'LearningRate');

    obj.LearningRate = value;
end

function set.MinimumBackgroundRatio(obj, value)
    validateattributes( value, { 'numeric' }, ...
                    { 'finite', 'scalar', '>=', 0, '<=', 1, 'nonsparse', 'real' }, ...
                    'ForegroundDetector', 'MinimumBackgroundRatio');

    obj.MinimumBackgroundRatio = value;
end

function set.NumGaussians(obj, value)
    validateattributes( value, { 'numeric' }, ...
                    { 'real', 'integer', 'scalar', '>', 0, 'nonsparse' }, ...
                    'ForegroundDetector', 'NumGaussians');

    obj.NumGaussians = value;
end

function set.InitialVariance(obj, value)
    if ischar(value)
        [numericValue, isValidNumeric] = vision.internal.codegen.str2num(value);

        if isValidNumeric
            validateInitialVariance(numericValue);
            obj.InitialVariance = numericValue;
        else
            str = validatestring(value, {'Auto'},  'ForegroundDetector', 'InitialVariance');
            obj.InitialVariance = str;
        end
    else
        validateInitialVariance(value);
        obj.InitialVariance = value;
    end

end
```

```
end

methods(Access=protected)
  function fgMask = stepImpl(obj, I, varargin)
    obj.Time = obj.Time+1;

    if isempty(varargin)
      if obj.Time < obj.NumTrainingFrames
        learningRate = 1/obj.Time;
      else
        learningRate = obj.LearningRate;
      end
    else
      learningRate = varargin{1};
      coder.internal.errorIf(learningRate <= 0 || learningRate > 1 || isnan(learningRate),...
                  'vision:ForegroundDetector:invalidLearningRate');
    end

    if isempty(coder.target)
      fgMask = obj.pForegroundDetector.step(I, learningRate);
    else
      if coder.internal.isTargetMATLABHost
        fgMask = ...
          vision.internal.buildable.ForegroundDetectorBuildable.ForegroundDetector_step(...
            obj.pForegroundDetector, ...
            obj.ImageClass, ...
            obj.StatClass, ...
            I, cast(learningRate,obj.ClassToUse));

      else
        [fgMask, obj.Weights, obj.Means, obj.Variances] = ...
          vision.internal.detectForeground(I, learningRate, ...
                                  obj.Weights, obj.Means, obj.Variances, ...
                                  obj.ClassToUse, ...
                                  obj.NumGaussians, obj.VarianceThreshold, ...
                                  obj.MinimumBackgroundRatio, ...
                                  obj.InitialWeight, obj.pInitialVariance);
      end
    end
  end

  function setupImpl(obj, I, varargin)

    setupTypes(obj,I);
    initializeParameters(obj, I);

    if isempty(coder.target)
      obj.pForegroundDetector.initialize(I, obj.NumGaussians, ...
                                obj.pInitialVariance, ...
                                obj.InitialWeight, ...
                                obj.VarianceThreshold,...
                                obj.MinimumBackgroundRatio);
    else
      if coder.internal.isTargetMATLABHost
```

```
            obj.pForegroundDetector = ...
                vision.internal.buildable.ForegroundDetectorBuildable.ForegroundDetector_construct(...
                    obj.ImageClass, obj.StatClass);
            obj.hasConstructed = true;
            vision.internal.buildable.ForegroundDetectorBuildable.ForegroundDetector_initialize(...
                obj.pForegroundDetector, ...
                obj.ImageClass, ...
                obj.StatClass, ...
                I, ...
                obj.NumGaussians, ...
                cast(obj.pInitialVariance,obj.ClassToUse), ...
                cast(obj.InitialWeight,obj.ClassToUse), ...
                cast(obj.VarianceThreshold,obj.ClassToUse),...
                cast(obj.MinimumBackgroundRatio,obj.ClassToUse));
        end
    end
end

function flag = isInputSizeLockedImpl(~,~)
    flag = true;
end

function flag = isInputComplexityLockedImpl(~,~)
    flag = true;
end

function flag = isOutputComplexityLockedImpl(~,~)
    flag = true;
end

function  resetImpl(obj)
    initializeStates(obj, obj.ClassToUse);
    if isempty(coder.target)
        obj.pForegroundDetector.reset();
    else
        if coder.internal.isTargetMATLABHost
            if obj.hasConstructed
                vision.internal.buildable.ForegroundDetectorBuildable.ForegroundDetector_reset(...
                    obj.pForegroundDetector, obj.ImageClass, obj.StatClass);
            end
        end
    end
end

function releaseImpl(obj)
    if isempty(coder.target)
        obj.pForegroundDetector.release();
    else
        if coder.internal.isTargetMATLABHost
            if obj.hasConstructed
                vision.internal.buildable.ForegroundDetectorBuildable.ForegroundDetector_release(...
                    obj.pForegroundDetector, obj.ImageClass, obj.StatClass)
                vision.internal.buildable.ForegroundDetectorBuildable.ForegroundDetector_delete(...
                    obj.pForegroundDetector, obj.ImageClass, obj.StatClass);
                obj.hasConstructed = false;
```

```
            end
        end
    end
end

function s = saveObjectImpl(obj)
    s.InitialVariance = obj.InitialVariance;
    s.LearningRate = obj.LearningRate;
    s.NumTrainingFrames = obj.NumTrainingFrames;
    s.NumGaussians = obj.NumGaussians;
    s.MinimumBackgroundRatio = obj.MinimumBackgroundRatio;
    s.AdaptLearningRate = obj.AdaptLearningRate;
    if obj.isLocked
        s.Time = obj.Time;
        s.ClassToUse = obj.ClassToUse;
        s.ImageClass = obj.ImageClass;
        s.StatClass = obj.StatClass;
        s.FrameSize = obj.FrameSize;
        s.NumChannels = obj.NumChannels;
        s.pInitialVariance = obj.pInitialVariance;
        if isempty(coder.target)
            [s.Weights, s.Means, s.Variances, s.NumActiveGaussians]  =...
                getStates(obj.pForegroundDetector);
        end
    end
end

function loadObjectImpl(obj, s, wasLocked)
    obj.LearningRate = s.LearningRate;
    obj.InitialVariance = s.InitialVariance;
    obj.NumGaussians = s.NumGaussians;
    obj.MinimumBackgroundRatio = s.MinimumBackgroundRatio;
    obj.NumTrainingFrames = s.NumTrainingFrames;
    obj.AdaptLearningRate = s.AdaptLearningRate;

    if wasLocked
        obj.ClassToUse = s.ClassToUse;
        obj.ImageClass = s.ImageClass;
        obj.StatClass = s.StatClass;
        obj.FrameSize = s.FrameSize;
        obj.NumChannels = s.NumChannels;
        obj.Time = s.Time;

        if isfield(s,'pInitialVariance')
            obj.pInitialVariance = s.pInitialVariance;
        else
            obj.pInitialVariance = s.InitialVariance;
        end

        if isempty(coder.target)
            obj.pForegroundDetector.initialize(ones([obj.FrameSize obj.NumChannels],obj.ClassToUse),...
                                               obj.NumGaussians,...
                                               obj.pInitialVariance,...
                                               obj.InitialWeight,...
                                               obj.VarianceThreshold,...
```

```
                                    obj.MinimumBackgroundRatio);
            obj.pForegroundDetector.setStates(s.Weights, s.Means, s.Variances, s.NumActiveGaussians);

        end
    end
end

function validateInputsImpl(obj, I, varargin)
    validateattributes(I, {'double','single','uint8'},...
                    {'real','nonsparse'},'vision.ForegroundDetector.step','', 2);

    if ~obj.AdaptLearningRate
        validateattributes(varargin{1}, {'double','single'},...
                        {'scalar','real','nonsparse'},...
                        'vision.ForegroundDetector.step','',3);
    end
end

function num = getNumInputsImpl(obj)
    if obj.AdaptLearningRate
        num = 1;
    else
        num = 2;
    end
end

function num = getNumOutputsImpl(~)
    num = 1;
end

function flag = isInactivePropertyImpl(obj, prop)
    props = {'VarianceThreshold', 'InitialWeight'};

    if ~obj.AdaptLearningRate
        props{end+1} = 'NumTrainingFrames';
        props{end+1} = 'LearningRate';
    end
    flag = ismember(prop, props);
end

function setupTypes(obj,I)
    if (isa(I,'double'))
        obj.ClassToUse = 'double';
        obj.ImageClass = coder.internal.const('double');
        obj.StatClass =  coder.internal.const('double');
    else
        obj.ClassToUse = 'single';
        obj.StatClass = coder.internal.const('float');
        if (isa(I,'uint8'))
            obj.ImageClass = coder.internal.const('uint8');
        else
            obj.ImageClass = coder.internal.const('float');
        end
    end
```

```
        end
    end

end

% -------------------------------------------------------------------------
function validateInitialVariance(value)
validateattributes( value, { 'numeric' }, ...
    { 'finite', 'scalar', '>=', 0, 'nonsparse', 'real' }, ...
    'ForegroundDetector', 'InitialVariance');
end
```

Appendix D – Iteration 1 - Full code - (MathWorks, 2010)

```
%% ITERATION 1 - RUSSELL BIRKETT


function multiObjectTracking()
 %% Creating the system objects used in reading selected video
obj = setupSystemObjects();
tracks = initializeTracks();
nextId = 1;
 DETECTING THE MOVING OBJECTS ACROSS FRAMES
while ~isDone(obj.reader)
   frame = readFrame();
   [centroids, bboxes, mask] = detectObjects(frame);
   predictNewLocationsOfTracks();
   [assignments, unassignedTracks, unassignedDetections] = ...
      detectionToTrackAssignment();
    updateAssignedTracks();
    updateUnassignedTracks();
    deleteLostTracks();
    createNewTracks();
    displayTrackingResults();
end
%%
    function obj = setupSystemObjects()

        obj.reader = vision.VideoFileReader('PETS_2009_VIEW_001.avi');
        obj.videoPlayer = vision.VideoPlayer('Position', [20, 400, 700, 400]);
        obj.maskPlayer = vision.VideoPlayer('Position', [740, 400, 700, 400]);
%GAUSSIAN MIXTURE MODELING
        obj.detector = vision.ForegroundDetector('NumGaussians', 2, ...
            'NumTrainingFrames', 10, 'MinimumBackgroundRatio', 0.5);
%CREATING BOUNDING BOXES
        obj.blobAnalyser = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
            'AreaOutputPort', true, 'CentroidOutputPort', true, ...
            'MinimumBlobArea', 100);
    end
%CONVERTING MASK TO RGB
      function displayTrackingResults()
        frame = im2uint8(frame);
        mask = uint8(repmat(mask, [1, 1, 3])) .* 255;
        minVisibleCount = 12;
        if ~isempty(tracks)
            reliableTrackInds = ...
                [tracks(:).totalVisibleCount] > minVisibleCount;
            reliableTracks = tracks(reliableTrackInds);
%DISPLAYING PREDICTED BOUNDRY BOX
            if ~isempty(reliableTracks)
                 bboxes = cat(1, reliableTracks.bbox);
                 ids = int32([reliableTracks(:).id]);
                 labels = cellstr(int2str(ids'));
                predictedTrackInds = ...
                    [reliableTracks(:).consecutiveInvisibleCount] > 0;
```

```
            isPredicted = cell(size(labels));
            isPredicted(predictedTrackInds) = {' predicted'};
            labels = strcat(labels, isPredicted);
%DRAWING ON PREDICTED BOXES ON FRAME AND MASK
             frame = insertObjectAnnotation(frame, 'rectangle', ...
               bboxes, labels);
            mask = insertObjectAnnotation(mask, 'rectangle', ...
               bboxes, labels);
          end
        end

        obj.maskPlayer.step(mask);
        obj.videoPlayer.step(frame);
      end
  %%
    function tracks = initializeTracks()
%% create an empty array of tracks
        tracks = struct(...
          'id', {}, ...
          'bbox', {}, ...
          'kalmanFilter', {}, ...
          'age', {}, ...
          'totalVisibleCount', {}, ...
          'consecutiveInvisibleCount', {});
    end
  %%
    function frame = readFrame()
        frame = obj.reader.step();
    end
%%DETECT THE FORGROUND FOR MORPHOLOGICAL FILTER
     function [centroids, bboxes, mask] = detectObjects(frame)

        mask = obj.detector.step(frame);

        mask = imopen(mask, strel('rectangle', [3,3]));
        mask = imclose(mask, strel('rectangle', [50, 50]));
        mask = imfill(mask, 'holes');
%% BLOB ANALYSIS TO FIND CONNECTED COMPONENTS
        [~, centroids, bboxes] = obj.blobAnalyser.step(mask);
      end
%% PREDICTING NEW LOCATIONS OF TRACK
     function predictNewLocationsOfTracks()
        for i = 1:length(tracks)
          bbox = tracks(i).bbox;
           predictedCentroid = predict(tracks(i).kalmanFilter);
           predictedCentroid = int32(predictedCentroid) - bbox(3:4) / 2;
          tracks(i).bbox = [predictedCentroid, bbox(3:4)];
        end
      end
      function [assignments, unassignedTracks, unassignedDetections] = ...
          detectionToTrackAssignment()
        nTracks = length(tracks);
        nDetections = size(centroids, 1);
%% COST OF ASSIGNING EACH DETECTION A TRACK
        cost = zeros(nTracks, nDetections);
```

```
        for i = 1:nTracks
            cost(i, :) = distance(tracks(i).kalmanFilter, centroids);
        end
         costOfNonAssignment = 20;
        [assignments, unassignedTracks, unassignedDetections] = ...
            assignDetectionsToTracks(cost, costOfNonAssignment);
    end
%%
    function updateAssignedTracks()
        numAssignedTracks = size(assignments, 1);
        for i = 1:numAssignedTracks
            trackIdx = assignments(i, 1);
            detectionIdx = assignments(i, 2);
            centroid = centroids(detectionIdx, :);
            bbox = bboxes(detectionIdx, :);
%% REPLACING PREDICTED BOUNDING BOX WITH DETECTION ONE AND UPDATE
            correct(tracks(trackIdx).kalmanFilter, centroid);
             tracks(trackIdx).bbox = bbox;
             tracks(trackIdx).age = tracks(trackIdx).age + 1;
             tracks(trackIdx).totalVisibleCount = ...
                tracks(trackIdx).totalVisibleCount + 1;
            tracks(trackIdx).consecutiveInvisibleCount = 0;
        end
    end
%%
    function updateUnassignedTracks()
        for i = 1:length(unassignedTracks)
            ind = unassignedTracks(i);
            tracks(ind).age = tracks(ind).age + 1;
            tracks(ind).consecutiveInvisibleCount = ...
                tracks(ind).consecutiveInvisibleCount + 1;
        end
    end
%% DELETING TRACK IS LOST
       function deleteLostTracks()
        if isempty(tracks)
            return;
        end
        invisibleForTooLong = 10;
        ageThreshold = 8;

        ages = [tracks(:).age];
        totalVisibleCounts = [tracks(:).totalVisibleCount];
        visibility = totalVisibleCounts ./ ages;

        lostInds = (ages < ageThreshold & visibility < 0.6) | ...
            [tracks(:).consecutiveInvisibleCount] >= invisibleForTooLong;

        tracks = tracks(~lostInds);
       end
%% CREATING NEW TRACKS WITH KALMAN FILTER
    function createNewTracks()
        centroids = centroids(unassignedDetections, :);
        bboxes = bboxes(unassignedDetections, :);
        for i = 1:size(centroids, 1)
```

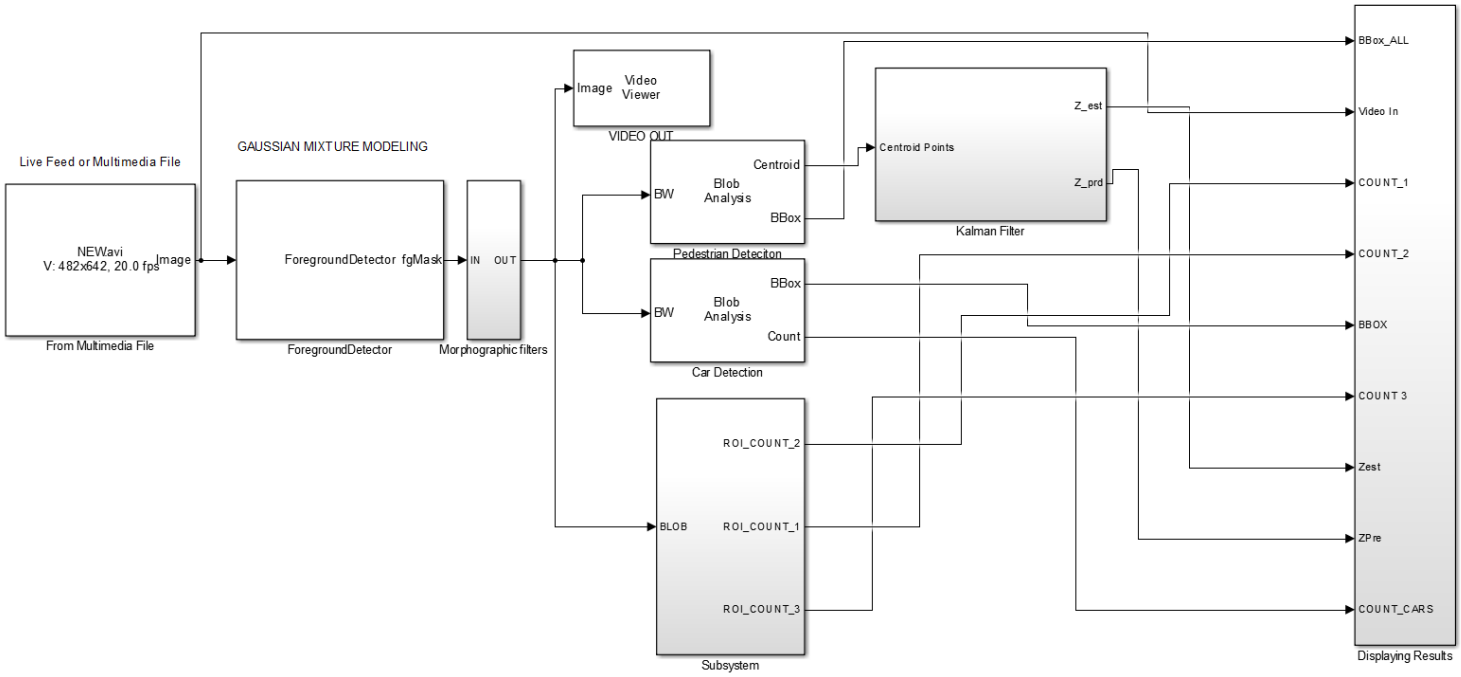
```
            centroid = centroids(i,:);
            bbox = bboxes(i, :);

            kalmanFilter = configureKalmanFilter('ConstantVelocity', ...
               centroid, [200, 50], [100, 25], 100);

            newTrack = struct(...
               'id', nextId, ...
               'bbox', bbox, ...
               'kalmanFilter', kalmanFilter, ...
               'age', 1, ...
               'totalVisibleCount', 1, ...
               'consecutiveInvisibleCount', 0);

            tracks(end + 1) = newTrack;

            nextId = nextId + 1;
         end
      end
end
```

Appendix E – Comparison (Sobral & Vacacant, 2013)

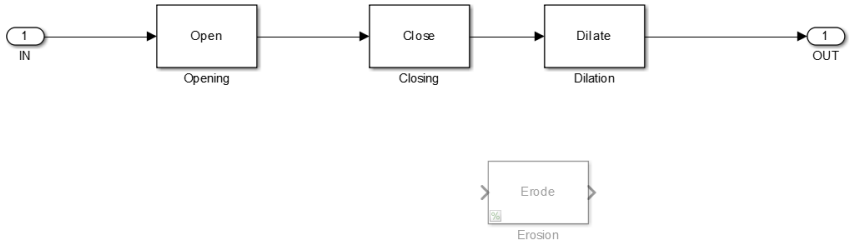
Global score of BS algorithms evaluated in BMC data set. The Top 5 best methods are highlighted in yellow. The best methods in each group are in red. The bold metric values show the best score of the *F*-Measure, *D*-Score and SSIM.

Method ID	Recall	Precision	*F*-Measure	PSNR	*D*-Score	SSIM	FSD
Basic methods, mean and variance over time							
StaticFrameDifferenceBGS	0.885	0.660	0.750	32,238	0.011	0.884	0.119
FrameDifferenceBGS	0.702	0.925	0.798	51,626	0.002	0.993	0.799
WeightedMovingMeanBGS	0.723	0.915	0.807	51,454	0.002	0.993	0.818
WeightedMovingVarianceBGS	0.721	0.912	0.805	51,427	0.002	0.993	0.814
AdaptiveBackgroundLearning	0.808	0.884	0.844	50.684	0.002	0.993	0.896
DPMeanBGS	0.597	0.935	0.729	51,881	0.002	0.992	0.642
DPAdaptiveMedianBGS	0.829	0.779	0.795	43,267	0.003	0.967	0.691
DPPratiMediodBGS	0.814	0.871	0.837	49,580	0.001	0.991	0.888
Fuzzy based methods							
FuzzySugenoIntegral	0.778	0.897	0.832	50.976	0.001	0.993	0.874
FuzzyChoquetIntegral	0.805	0.876	0.837	50.366	0.001	0.992	0.884
LBFuzzyGaussian	0.909	0.740	0.808	42,364	0.003	0.974	0.738
Statistical methods using one Gaussian							
DPWrenGABGS	0.795	0.922	0.853	51,394	0.001	0.993	0.922
LBSimpleGaussian	0.855	0.770	0.805	45,073	0.002	0.982	0.767
Statistical methods using multiple Gaussians							
DPGrimsonGMMBGS	0.717	0.913	0.802	51,445	0.002	0.993	0.808
MixtureOfGaussianV1BGS	0.793	0.912	0.847	51,107	0.001	0.993	0.910
MixtureOfGaussianV2BGS	0.893	0.813	0.850	48,383	0.002	0.992	0.899
DPZivkovicAGMMBGS	0.665	0.928	0.774	51,717	0.002	0.992	0.746
LBMixtureOfGaussians	0.868	0.834	0.848	48,794	0.001	0.991	0.907
Type-2 Fuzzy based methods							
T2FGMM_UM	0.661	0.935	0.774	51,792	0.002	0.992	0.745
T2FGMM_UV	0.800	0.747	0.762	43,395	0.003	0.971	0.628
T2FMRF_UM	0.852	0.670	0.743	31,732	0.012	0.872	0.030
T2FMRF_UV	0.678	0.888	0.763	50.900	0.002	0.991	0.716
Statistical methods using color and texture features							
MultiLayerBGS	0.893	0.863	0.875	49,398	**0.001**	0.993	0.974
Non-parametric methods							
PixelBasedAdaptiveSegmenter	0.923	0.852	**0.885**	49,412	0.002	**0.994**	0.985
GMG	0.947	0.703	0.803	41,826	0.003	0.979	0.730
VuMeter	0.722	0.842	0.775	50.296	0.002	0.992	0.735
Methods based on eigenvalues and eigenvectors							
DPEigenbackgroundBGS	0.879	0.658	0.747	32,843	0.011	0.891	0.114
Neural and neuro-fuzzy methods							
LBAdaptiveSOM	0.838	0.907	0.867	50.553	0.001	0.992	0.952
LBFuzzyAdaptiveSOM	0.877	0.811	0.836	46,182	0.002	0.982	0.848

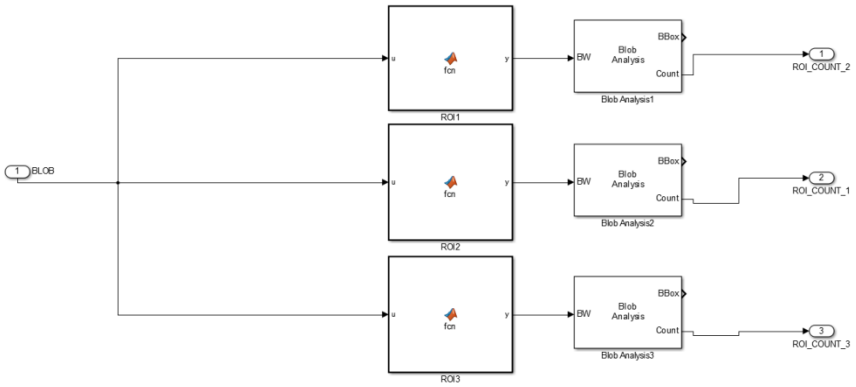
Appendix F – Finial Iteration



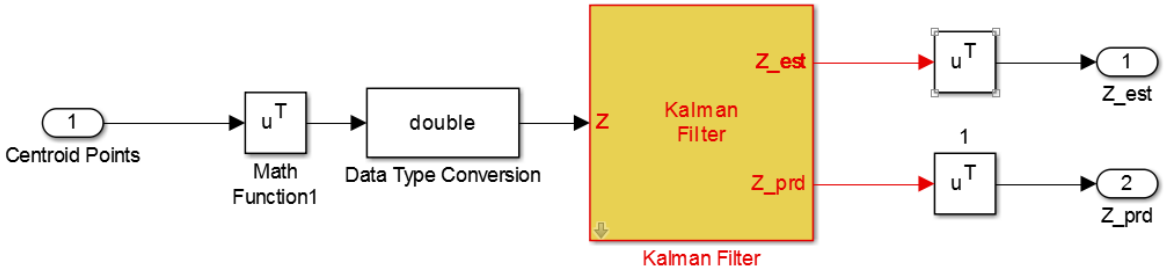
Morphological Filters



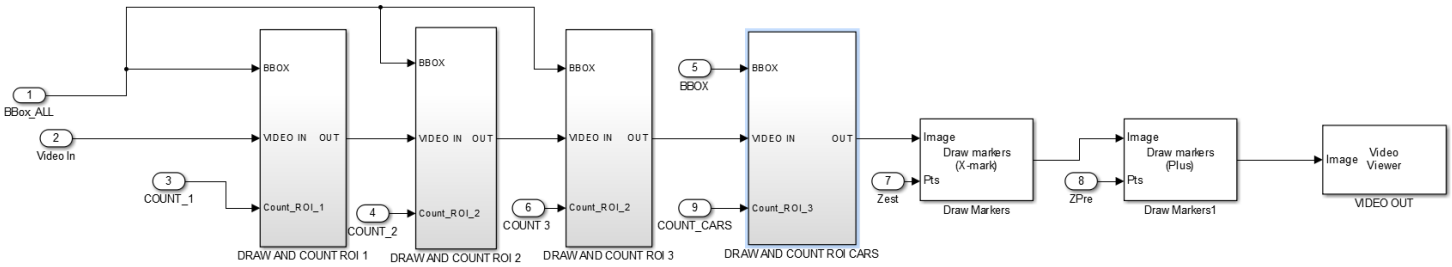
ROI_COUNT



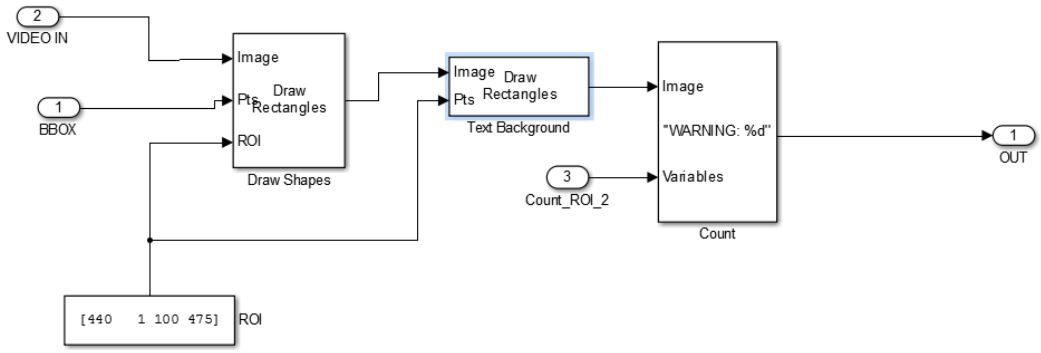
Kalman Filter



Display



DRAW AND COUNT ROI - Subsystem



Bibliography

Arulampalam, S., Maskell, S., Neil, G., & Clapp, T. (2002). A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE TRANSACTIONS ON SIGNAL PROCESSING VOL 50*, 174-188.

Athanesious, J. (2012). Systematic Survey on Object Tracking Methods in Video. *International Journal of Advanced Research in Computer Engineering & Technology*, 242-247.

Benezeth, Y., Jodoin, P.-M., Emile, B., & Laurent, H. (2012). Comparative study of background subtraction. *Journal of Electronic Imaging*, 2-4.

Das, D., & Saharia, S. (2014). Implementation And Performance Evaluation O fBackground Subtraction Algorithms. *International Journal on Computational Sciences & Applications* , 48-53.

Feng , J., Ding, M., & Zhang, X. (2013). Decision-based adaptive morphological filter for fixed-value impulse. *ScienceDirect*, 2-8.

Fuhr, G. (2012). Robust Patch-Based Pedestrian Tracking using. *Institute of Informatics*, 1- 12.

Maragos, P. (2005). Morphological filtering for Image and Feature Detection. *The Image and Video Processing Handbook 2nd Edition*, 135-156.

MathWorks. (2010). *Motion-Based Multiple Object Tracking.* Retrieved July 12, 2016, from MathWorks: http://au.mathworks.com/help/vision/examples/motion-based-multiple-object-tracking.html

MathWorks. (2010). *Tracking Cars Using Foreground Detection.* Retrieved August 2, 2016, from ForegroundDetector: http://au.mathworks.com/help/vision/examples/tracking-cars-using-foreground-detection.html#vision_product-viptraffic

MathWorks. (2015). *Edge detection methods for finding object boundaries in images*. Retrieved October 1, 2016, from MathWords: https://au.mathworks.com/discovery/edge-detection.html

MathWorks. (2016). *Detecting Cars Using Gaussian Mixture Models.* Retrieved August 12, 2015, from MathWorks.com: http://au.mathworks.com/help/vision/examples/detecting-cars-using-gaussian-mixture-models.html

Parekh, H. S., Thakore, D. G., & Jaliya, U. K. (2014). A Survey on Object Detection and Tracking Methods. *International Journal of Innovative Research in Computer and Communication Engineering*, 2970 - 2978.

Poole, D., & Mackworth, A. (2010). *Artificial Intelligence - Foundations of Computational Agents.* Cambridge: Cambridge University Press.

Ray, D. (2013). Edge Detection in Digital Image Processing. *University Of Washington*.

Reynolds, D. (2009). Gaussian Mixture Models∗. *MIT Lincoln Laboratory*, 2-4.

Roeder, A. (2012). A computational image analysis glossary for biologists. *The Company of Biologists*, 3071-3080.

Semko, D. (2007, June). OPTICAL FLOW ANALYSIS AND KALMAN FILTER TRACKING IN VIDEO SURVEILLANCE ALGORITHMS. CALIFORNIA , United States Of America: Naval Postgraduate School MONTEREY CALIFORNIA .

Sobral, A., & Vacacant, A. (2013). A comprehensive review of background subtraction algorithms. *Computer Vision and Image Understanding*, 8-19.

Su, F., & Fang, G. (2012). Adaptive Colour Feature Identification in. *Mathematical Problems in Engineering*, 234-252.

SULIMAN, C., CRUCERU, C., & MOLDOVEANU, F. (2010). Kalman Filter Based Tracking in an Video Surveillance System. *10th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS*, 54-58.

University of Reading. (2009, June 25). *PETS 2009 Benchmark Data.* Retrieved January 26, 2016, from Computer Society Conference on Computer Vision and Pattern Recognition: http://www.cvg.reading.ac.uk/PETS2009/a.html

Zhou, Z., Wu, D., & Zhu, Z. (2016). Object tracking based on Kalman particle filter with LSSVR. *Optik - International Journal for Light and Electron Optics*, 613-619.