UNIVERSITY OF LIVERPOOL

MOTION TRACKING IN LOW RESOLUTION IMAGERY

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy

By

Vivekanand Govinda

May 2009

ABSTRACT

This thesis describes the development of an automatic monitoring system for indoor environments. The aim is to develop a system that allows general movement of one or more people to be determined using a very low resolution colour sensor. The sensor has sufficiently low resolution that objects are not fully resolved – to protect privacy. It is demonstrated that movement can be monitored to a high degree of accuracy using spectral un-mixing techniques adapted from satellite remote sensing applications. These algorithms allow the fractional contributions from different colours within each pixel to be estimated and this can then be used to assist in the detection and tracking of small objects. Using colour as a tracking/monitoring aid is not new but the current application presents significant difficulties. The detection and monitoring algorithms have to take into account that the colour measurements made by a vision system often depend on the ambient illumination and in rooms with open windows and blinds for example, the illumination conditions can change very rapidly. In order to recognise an object in different lighting environments, the system must have the ability to discount the effects of the illumination changes. An important task in this technique adaptation for indoor environment use is to find an automated way to fit a triangle around a set of data points as closely as possible for each frame in a sequence of images. Several optimisation algorithms are evaluated to determine which one is most appropriate for this triangle fitting task. The detection mechanism alone is not enough to be able to monitor the subjects. Tracking algorithms are also applied to be able to generate continuous tracks of the people being monitored. The State-Space approach is used to model the motion of the objects of interest with the Kalman Filter being the chosen predictive filter to find the optimal state. The monitoring algorithms developed are tested in a series of real-life challenging scenarios such as quickly changing illumination conditions and occlusion of objects.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr Jason Ralph for his support, advice and encouragement throughout the whole duration of my studies, and my secondary supervisor Professor Joe Spencer for making this PhD a reality. I would also like to thank Dr Yannis Goulermas who was always there whenever I needed help with the programming and implementation of the various optimisation techniques. The amount of time spent on coding is not reflected in this thesis. My thanks also go to the other colleagues and friends who I met during my time at the University of Liverpool and who have accompanied in the various diversions from work that a research student normally takes. Special thanks here are due to Elias whose help with Matlab programming was of utmost importance when I was in the early days of my PhD, and Bao who has been a lead actor in the various test video scenarios during this project. I am also grateful to Dr Ralph and the Department of Electrical Engineering and Electronics for their financial support.

And last but not least, I would like to thank my parents for all their support in numerous ways over the years, which has enabled me to be where I am today, and for which I could never thank them enough. I would like to dedicate this thesis to little Jash whose birth has brought a lot of happiness to our family and relatives.

CONTENTS

LIST OF FIGURES

CHAPTER 1

1. Introduction

1.1 Significance and motivation

While many pieces of legislation aimed at protecting citizens' privacy exist, providing a single definition of privacy is difficult. The boundaries and content of what can be considered private differ among cultures and individuals, but they often share basic common ideas - one of them being the concept of being observed by a camera. The ability to monitor the activity of people within an enclosed area without intruding on their privacy is an important factor for many medical and security systems. The use of high resolution imaging is not welcome all the time as people are not always comfortable with being 'watched' and there are also laws that safeguard our privacy. Besides the presence of an intrusive camera in places such as a hospital ward, an office or other secure locations may influence the true behaviour of the people being monitored. One way to get round this is by using very low-resolution cameras which cannot form true images of the subjects that they are filming. The condition of 'no true image formation', whereby the human eye will find it extremely hard to distinguish different objects in a scene, can be achieved when all the objects of interest are comparable in size to the pixel/sensor elements and hence cannot be fully resolved. By not resolving the details of a scene in this manner, monitoring systems minimise the intrusion factor whilst still being able to monitor the motion of people and objects within the field of view. Places such as offices, hospitals or nursing homes are the environments most likely to benefit from these systems as they often require monitoring capability for safety and medical reasons but with a high degree of privacy.

The emphasis of this thesis is to use data from a very low resolution video camera to monitor the motion of objects in these sensitive indoor environments. The conventional way of detecting moving objects in spatial image processing is by differencing successive or neighbouring image frames in a sequence. This is not a viable option in this case as insufficient detail is available and background noise will also be detected. The low spatial resolution means that it can become very difficult to separate a very small object from the noise. An alternative method is devised to achieve detection within the low number of pixels available from low-resolution cameras. Techniques borrowed from satellite sub-pixel detection systems – which make use of multiple light wavebands or colours – are adapted. In satellite reconnaissance, sub-pixel target detection is very common and many algorithms have been developed to deal with situations where each pixel contains several distinct types of object or background, each with a different colour signature. In this context, 'colour' is a general term that is related to the intensities in different visible and/or thermal (i.e. infrared) wavebands – the number of wavebands can be significantly larger than the usual three colours that are familiar in the visible band. The efficacy of the algorithms depends on how mixed the pixels are and how much colour contrast they contain. If each pixel contains too many different coloured objects/backgrounds it can become difficult to discern any common colour combinations between different pixels, and if the pixels contain colours that are too similar it becomes difficult to identify distinct colours. The second problem arises because of the underlying assumption that the pixels can be represented by a mixing model that can be decomposed to find 'pure' colours. Often this is done by algorithms such as the Simplex Shrink-Wrap algorithm [1] or the vertex component analysis [2].

Using colour as a tracking/monitoring aid is not new but the current application presents significant difficulties. The detection and monitoring algorithms have to take into account that the colour measurements made by a vision system often depend on the ambient illumination and in rooms with open windows and blinds for example, the illumination conditions can change very rapidly. In order to recognise an object in different lighting environments, the system must have the ability to discount the effects of the illumination changes. An important task in this technique adaptation for indoor environment use is to find an automated way to fit a triangle around a set of data points as closely as possible for each frame in a sequence of images. Several optimisation algorithms are evaluated during the development stage of the system to determine which one is most appropriate for this type of environment and the subjects of interest. The detection mechanism alone is not enough to be able to monitor the subjects, i.e. people. Tracking algorithms need also be applied to be able to generate continuous tracks of the people being monitored.

1.2 Structure of thesis

This thesis is divided into five further chapters:

- Chapter 2, *Literature review,* describes the main components of a monitoring system with special importance given to the object detection and tracking part of such systems.
- Chapter 3, *Related information on processing techniques,* shows the various ways of representing colour information, and detection algorithms used in spectral imaging applications are introduced.
- Chapter 4, *Detecting objects,* examine how the triangle wrapping algorithms can be adapted for indoor environments to detect small objects. Another detection technique involving the use of human skin is also introduced.
- Chapter 5, *Data collection, experiments and results*, contains all the tests and results of the algorithms when applied to both artificial and real imagery.
- Chapter 6, *Conclusions and further work*, is an overall evaluation of the work done, and ideas for further work.

1.3 Publications

Parts of this thesis have been presented at conferences and published in a journal paper.

1. V. Govinda, J. F. Ralph, J. W. Spencer, J. Y. Goulermas, *Small object monitoring for sensitive indoor compounds*, Measurement (2009), doi: 10.1016/j.measurement.2009.01.005.
2. V. Govinda, J. F. Ralph, J. W. Spencer, J. Y. Goulermas, *Pixel decomposition for tracking in low resolution videos*, Signal and Data Processing of Small Targets 2008 Proc. of SPIE Vol. 6969.
3. V. Govinda, J. F. Ralph, J. W. Spencer, J. Y. Goulermas, *Spectral unmixing for tracking human motion in low resolution imagery*, 2007 Journal of Physics: Conference Series 76, paper 012031.
4. V. Govinda, J. F. Ralph, J. W. Spencer, J. Y. Goulermas, D. H. Smith, *Tracking subpixel targets in domestic environments*, Proc SPIE 6236 Signal and Data Processing of Small Targets 2006, 623603/1-8.

CHAPTER 2

2. Literature Review

2.1 Introduction

The main aim of this thesis is to describe the development of a monitoring system that is able to track small objects relative to an image plane of low resolution size. Object sizes can range from less than a pixel to a few pixels. Low resolution video cameras and other similar non-intrusive image sensors form part of the equipment that can be used to capture this information. This project's emphasis is to use this data to monitor the motion of objects. The next two chapters consist of the theoretical background knowledge necessary to understand the development of a complete low resolution image monitoring system. This chapter contains a review of the techniques used in two important tasks of most video monitoring systems, object segmentation and tracking. Object segmentation is the first step in the overall tracking procedure. This is usually achieved by selecting certain features within an object that are common, for example colour, edges and shapes. One detection approach involving colour which is introduced in the next chapter 3 is the use of remote sensing techniques to detect very small objects. The satellite reconnaissance community often has to deal with small objects in images and an attempt is made in this thesis to adapt these techniques augmented by using time-dependent colour intensity profiles of the low resolution images to detect moving objects. Once an object is detected, algorithms exist to track it. These algorithms are assumed to be part of a filtering and data association process involving prior information about the scene or object, dealing with object dynamics, and evaluation of different hypotheses. Tracking is an established field employed in various sectors, and a brief survey of the techniques that exist is given in this chapter. While not all the techniques mentioned here can be used when dealing with small objects in low resolution image sequences, it is nevertheless very important to know how they work as often techniques originally designed for a purpose can be modified to answer the demands of another purpose with different requirements.

2.2 Monitoring Systems

A generic monitoring system can be broadly defined as consisting of several main building blocks as shown in Figure 2-1. The environment being monitored could be a hospital ward or another type of domiciliary care room and depending on the chosen environment, one can then decide on a particular type of sensor or sensor network arrangement to collect the data. The sensors in this case are low resolution imagers or other similar non-intrusive sensors. The data collected in this way is then processed to finally give useful information about the environment being monitored. The data processing part of the monitoring system chain is the section where most effort was devoted during this project and a detailed description of the various steps involved is given in the next sections.

Video-based monitoring systems can be seen in many places, such as in car parks, supermarkets, airports and train stations. The surveillance systems are usually used to detect abnormal, dangerous situations and prevent their happening as soon as possible. Many systems require human operators monitoring the scene continuously via many video displays to detect suspicious activities and as a result some of these systems contain some disadvantages such as the cost of running them and the fact that human operators are not always fully reliable. These surveillance systems are also not true monitoring systems as they are just networks of cameras that collect images over time but with limited capability of interpreting the collected data. The above disadvantages of manual surveillance systems have led to much research in developing automated video-based surveillance systems. For the purpose of this thesis, only systems with single-image capturing devices are analysed. However the techniques developed can be extended and adapted for multi-sensor systems.

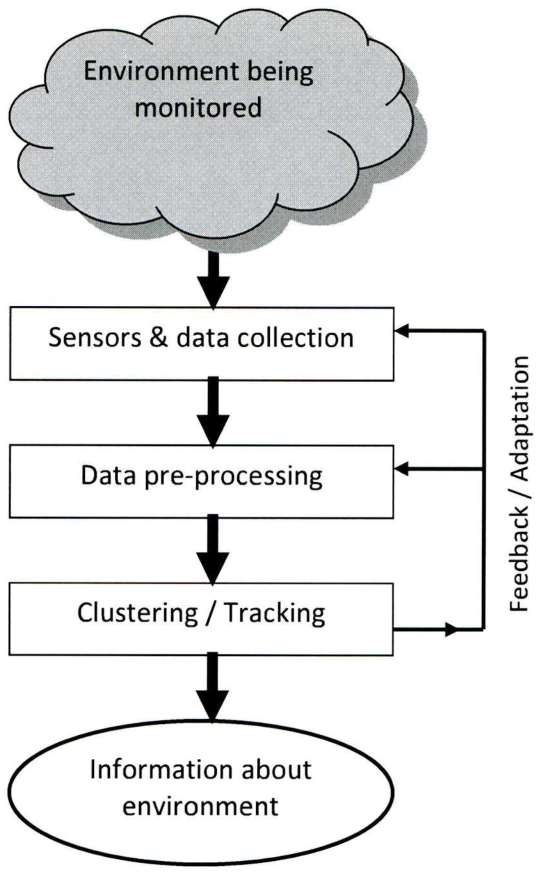

Environment being monitored
Sensors & data collection
Data pre-processing
Clustering / Tracking
Information about environment
Feedback / Adaptation

Figure 2-1 : Elements of a video monitoring System

2.2.1 Examples of current indoor monitoring systems

One very good example of an indoor monitoring system is called KidsRoom which is a fully automated and interactive narrative play-space for children developed at the MIT Media Laboratory [3]. KidsRoom uses a single camera, which is placed on the ceiling to minimise object occlusion, to track multiple non-rigid objects in a room and these objects often interact with others. The system does not require people in the space to wear any special clothing or hardware, and it can accommodate up to four people simultaneously. KidsRoom uses knowledge about objects being tracked and their current relationships to one another, also known as *contextual information,* to track multiple, complex, non-rigid objects simultaneously. KidsRoom uses a *closed-world* assumption to select weights of the position, size, colour and velocity used in the object matching algorithm. A closed-world is a region of space and time in which the specific context of what is in the region is assumed to be known. The tracking algorithm uses four data structures: (1) Each object has a data structure that stores the object's estimated size, colour, velocity, and current and past position. This information is used for matching each object in the last frame to a blob in the new frame. A blob is a group of connected pixels based on certain conditions of the pixels' brightness and spatial adjacency [4]. (2) Image blobs are computed in each frame using background differencing; each blob's size, colour, and position is recorded. (3) A local closed-world data structure exists for every blob and stores objects that are assigned to the blob. (4) Finally, the system uses knowledge about the global closed-world, which stores information about which objects are in the entire scene [3].

Another example is the Pfinder ("Person Finder") system [5]. It is also a single-camera system that tracks the movement of the human body parts. Although this system's main aim is to recognise human gestures, it is mentioned here because the technique used can be improved to be able to track the movement of a whole body. The colour space used is the YUV representation of colour. YUV is used in video transmission where Y is the luminance channel that contains the images that would be displayed on a black-and-white television receiver while the U and V components carry the information about the colour of the images [6]. No information is given by the authors on their choice to use this colour space but it is assumed that this system was developed with a view to be compatible with legacy video systems used in video transmission. Pfinder uses a multi-class statistical model of colour and shape to detect parts of a person such as head, hands and feet. As the Pfinder processes scenes that consist of relatively static situations such as an office, and a single person with moving parts, the developers of this system used different types of characterisation model for the scene and for the person. The scene surrounding the human is modelled as a texture surface; each point on the texture surface is associated with a mean colour value and a distribution about that mean. The colour distribution of each pixel is modelled with a Gaussian described by a full covariance matrix. People are represented as blobs with each blob having a spatial (x, y) and colour (Y, U, V) component. The statistics of each blob are updated recursively to combine information contained in the most recent image with knowledge contained in the current class statistics and the priors. This system also claims to be able to compensate for small or gradual changes in lighting by updating the texture maps over time. It cannot however compensate for large or sudden changes in the scene as they get mistaken as being part of the foreground region. Another limitation is that the domain-specific assumptions used to model the gestures of the human body to make the task tractable can be broken fairly easily because they represent only a limited number of gestures [5].

W^4 is another single camera surveillance system for tracking people and detecting human activities using a single camera [7]. W^4 employs a combination of shape analysis and tracking to locate people and their parts (head, hands, feet, torso) and to create models of people's appearance so that they can be tracked through interactions such as occlusions. It can learn and model background scenes statistically to detect foreground objects, even when the background is not completely stationary (e.g., motion of tree branches). W^4 can also determine whether people are carrying objects, and can segment objects from their silhouettes, and construct appearance models for them so they can be identified in subsequent frames. However it does contain certain limitations that will not make it appropriate for this project. It has been designed for outdoor purposes using only grey-scale video imagery captured at a high resolution of 320 x 240 pixels at a frame rate of 25 Hz. The fact that it uses a silhouette-based method to perform foreground detection critically affects the camera orientation. A fairly oblique view of the scene is required because whole body silhouettes are needed, and hence its field of view is limited. Another problem with W^4 is that it is severely affected by shadows present in the scene [7].

The systems mentioned so far use fairly high resolution cameras and do not address the need for privacy that exists in certain sensitive indoor environments such as care homes for the elderly or hospital wards. One example system that makes use of low resolution sensors has been developed by the Centre for Intelligent Monitoring Systems (CIMS) in the Department of Electrical Engineering and Electronics at the University of Liverpool and has been demonstrated successfully in situ in a domiciliary care facility in the London Borough of Merton [8]. This system, called the Merton Intelligent Monitoring System (MIMS), is based around a two-dimensional optical array and infrared detectors (see Figure 2-2(a)) which monitor the activity of occupants within a number of rooms within the facility [9]. The two-dimensional optical sensor array – referred to as a Pseudo-Optical Radar System (PORS) – monitors the colour state of selected individual pixels (or clusters of pixels) that correspond to regions of the room where different types of activity are likely to be occurring. Examples of such regions might be pixels corresponding to chairs, the floor area or the entrance/exit zones (Figure 2-2(b)). The colour information is collected in *RGB* (Red, Green, Blue) space and then converted to the *HLS* (Hue, Lightness, Saturation) colour space for further processing [10]. More information about colour spaces is given in Chapter 3. This pixel based monitoring system is augmented by the use of three infrared detectors arranged in a triangular formation that monitor movement from one region of the room to another.

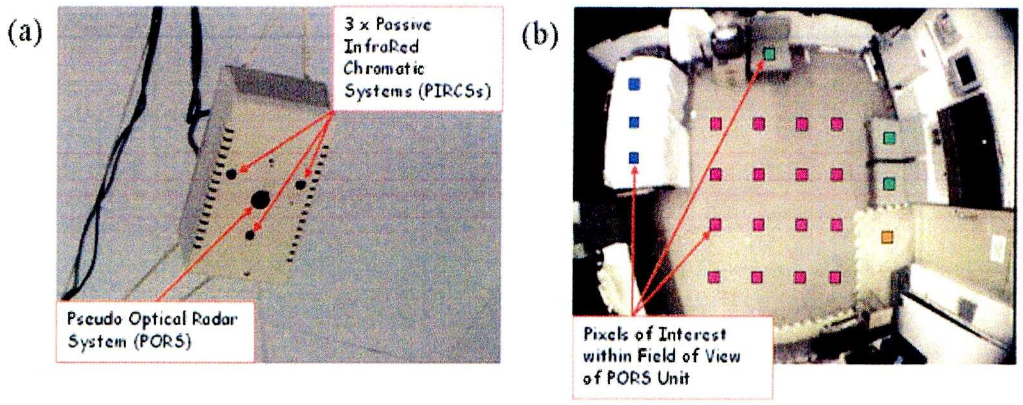

(a)
3 x Passive InfraRed Chromatic Systems (PIRCSs)
Pseudo Optical Radar System (PORS)
(b)
Pixels of Interest within Field of View of PORS Unit

Figure 2-2 : (a) MIMS Sensor, (b) Pixels within the field of view of the PORS imager that are monitored by the MIMS processing system, indicating regions of activity: e.g. chairs, floor and door.[9]

This approach to monitoring indoor environments can be summarised by several phases. Firstly, the presence of an object (e.g. a person) at a particular location is detected by a change in the RGB colour channel values of the selected cluster of pixels of interest, and if the change occurs in a series of adjacent points over time, movement of the object is assumed to occur. Secondly, a timer is used to record the duration of movement that happen at the various locations of interest. It is at this stage that the *RGB* information is converted to *H*, *L* and *S*, with the intention to make *H* represent the location of the events which occur, *L* to represent the collective time of the perturbation across all locations and *S* to represent the spread of events across the various locations. The last phase is to use the *HLS* information to distinguish between movement over long and short distances with the intent of discriminating between different types of motion. The system has been developed for long term use and the data can be recorded and processed over weeks and months to allow for the detection and identification of emergent patterns of behaviour amongst the occupants of the rooms – of particular interest is the ability to detect and identify the emergence of anomalous behaviour that might indicate the development of a medical condition needing attention from a care giver. One of the main advantages of this system is that it has derived an intuitive description of human activity obtained from processing months of data from multiple rooms in a methodical way. This description can be interpreted by a care giver without specific technical training, whilst maintaining the privacy of the occupant and enhancing their perceptions of safety and security.

In the current project, one of the aims is to use normal cheap low resolution image sensors to capture poor quality imagery at a certain frame rate (10 fps is used during the experiments) and then track the moving objects that exist in the videos in a reliable way. This looks more like a traditional automated video-based monitoring system and these systems usually consist of three main parts: (1) object segmentation and detection (2) object tracking (3) behaviour recognition. Object segmentation aims to extract objects from the background. This task is difficult given the existence of camera noise, object occlusion and unstable environmental conditions. Object tracking aims to label the objects in the scene and track their properties throughout time. The behaviour recognition task recognises a variety of object behaviours such as a person walking, running, falling down or standing. While the uncertainty and complexity of the behaviours are challenging and are introduced in this thesis, the main focus will be on the object segmentation and tracking parts of a monitoring system with particular attention given to cases when input videos are of very low spatial resolution containing objects of small sizes (from less than a pixel to a few pixels wide) relative to the video frame.

2.3 Object detection

Selecting the right features to segment an image plays a critical role in tracking. The most desirable property of a visual feature is its uniqueness so that the objects can be easily distinguished in the feature space. Some of the more common visual features are:

- *Colour:* The apparent colour of an object is influenced primarily by two physical factors, 1) the spectral power distribution of the illumination source and 2) the surface reflectance properties of the object. In digital image processing, colour is represented in different ways called colour spaces. A colour space is a mathematical model describing how colours can be represented as numbers. A few common examples are RGB, HSV and CIE L*a*b*. More information about these spaces is given in Chapter 3 as colour cues are used for the detection process in this thesis.
- *Edges:* Object boundaries usually generate strong changes in image intensities and edge detection is used to identify these changes. An important property of edges is that they are less sensitive to illumination changes compared to colour features. Algorithms that track the boundary of the objects usually use edges as the representative feature and an evaluation of these algorithms is provided by Bowyer et al. [11]. They applied eight different techniques on a collection of real images and compared the results with manually-specified ground truth. They found that the Canny method, with the Heitger method a close second, outperformed the others. The Canny detector uses the first derivative of a Gaussian and it uses four filters to detect horizontal, vertical and diagonal edges [12]. However it can be a time consuming process as it consists of complex computations.
- *Optical Flow:* Optical flow is a dense field of displacement vectors which defines the translation of each pixel in a region. It is computed using the brightness constraint, which assumes brightness constancy of corresponding pixels in consecutive frames and any changes in intensity are solely due to motion [13]. Optical flow is commonly used as a feature in motion-based segmentation and tracking applications and a performance evaluation of the various optical flow methods can be obtained from the survey by Barron[14].

The brightness constancy assumption does however pose certain problems as in reality the brightness constantly changes in a dynamic scene and hence motion vectors can be wrongly allocated even when no displacement has occurred.

In general, many tracking algorithms use a combination of these features. For the purpose of this project, colour is used as the main detection feature, as will be explained in later sections. The optical flow method is not suitable here because of the continuously changing light intensities that occur in such environments. Edge detection method is also not thought to be appropriate for low resolution imagery because of the levels of noise present and also because of the fact that objects of interest can be too small for their edges to be picked up. However, a variant of the edge detection mechanism is implemented to detect objects that are less than a pixel in size as these objects tended to modulate the intensity distributions of certain pixels as they move across pixel boundaries.

Every tracking method requires an object detection mechanism either in every frame or when the object first appears in the video. A common approach for object detection is to use information in a single frame. However, some object detection methods make use of the temporal information computed from a sequence of frames to reduce the number of false detections. This temporal information is usually in the form of frame differencing, which highlights changing regions in consecutive frames. Given the object regions in the image, it is then the tracker's task to perform object correspondence from one frame to the next to generate the tracks. Another closely related technique achieves object detection by building a representation of the scene called the background model and then finds deviations from the model for each incoming frame. Any significant change in an image region from the background model signifies a moving object and the pixels constituting the regions undergoing change are marked for further processing.

The simplest method of temporal frame differencing compares the current image with a static background image. If the difference between a pixel in the current image and the corresponding pixel in the background image is greater than a threshold, the pixel is classified as belonging to a moving object; otherwise it belongs to the background. Noise in the difference image can be removed by using a median filter and then a pixel connectivity algorithm is used to find blobs (there may be more than one).

The connected components are then labelled by scanning the difference image, pixel-by-pixel (from top to bottom and left to right) to identify connected pixel regions, i.e. regions of adjacent pixels which share the same set of intensity values [15]. However since this method uses a static background image, it cannot deal with changes in conditions of the environment such as change of lighting conditions or background motion. This method can be improved by updating the background image regularly and several ways exist in literature to describe this process. Haritaoglu's W^4 system maintains a pair of values (minimum and maximum) for each background pixel [7]. These values are updated over time to adapt to the changes in the scene. A pixel in the current image is classified as a background pixel if its intensity is inside the range (minimum value, maximum value) of the corresponding background pixel. Another background update method is proposed in Koller et al. [16] where the background image is updated differently for pixels belonging to background and pixels belonging to moving objects using weights estimated from the rate of background change. This method can detect moving objects accurately even when they are moving slowly in the scene.

Another approach to temporal frame information is to find the absolute difference between the current frame and the previous frame instead of the background frame (Frame Differencing). Frame differencing is very quick to adapt to changes in lighting or motion and objects that stop are no longer detected. However, frame differencing tends to only detect the leading and trailing edge of a uniformly coloured object and as a result, it can be very hard to detect the entire object that is moving in a scene. One way to solve this problem is to adjust the temporal scale (frame rate) at which frame differencing is done in a technique called 'double-differencing', as defined in the equation below [17]:

$$D(N) = \left| I(t) - I(t+N) \right| \tag{2-1}$$

where $I(t)$ is the frame at time t. $D(+N)$ and $D(-N)$ are computed after choosing a value for N which depends on the size and speed of the object of interest and the frame rate of the input video. $D(+N)$ contains the object's current position and its future position, $D(-N)$ contains the object's current position and its past position. Next as part of a three-frame differencing technique, the logical AND operation is taken between $D(+N)$ and

D(−*N*) and this operation results in an image with only the object in its current position [17]. A pictorial explanation is shown in Figure 2-3.

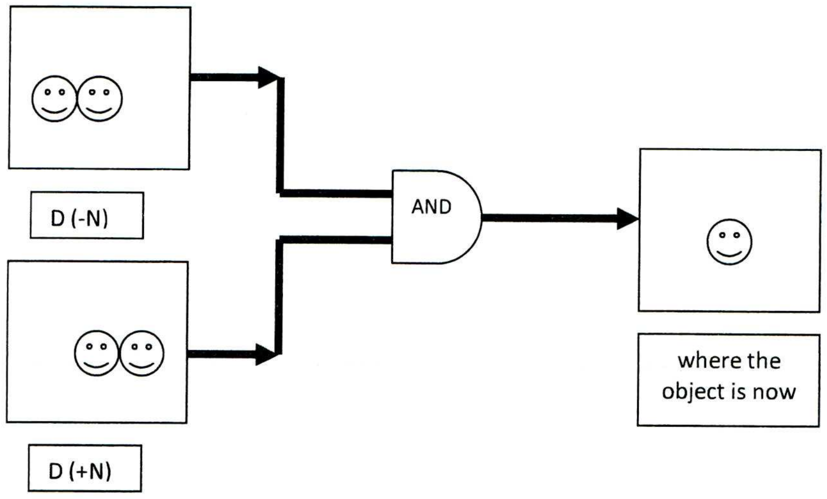


Figure 2-3 : Three–frame differencing method using AND operation. D(-N) contains where object was and where it is now, D(+N) contains where the object is now and where it will be, and finally after performing AND operation, the current location of the object is obtained.

Another way to detect objects is by using image segmentation algorithms which partition the image into perceptually similar regions. Every segmentation algorithm attempts to address two problems, namely the criteria for a good partition and the method for achieving efficient partitioning. Two common techniques are described next.

The mean-shift technique is an approach to find clusters in the joint colour and spatial space, $[l, u, v, x, y]$, where $[l, u, v]$ represents the colour and $[x, y]$ represents the spatial location. Given an image, the algorithm is initialised with a large number of hypothesised cluster centres randomly chosen from the data. Then, each cluster centre is moved to the mean of the data lying inside the multidimensional ellipsoid centred on the cluster centre. The vector defined by the old and the new cluster centres is called the *mean-shift vector*. The mean-shift vector is computed iteratively until the cluster centres do not change their positions [18]. Care must also be taken to ensure that the various parameters to obtain better segmentation, for instance selection of the colour and spatial kernel bandwidths, are correctly set and the high computational complexity of the algorithm is a significant barrier to its scalability to practical applications. Nevertheless, this technique has been adapted to solve other applications such as edge detection, image regularisation, and tracking with various degrees of success [19].

Another method is the *snakes* or active contour method. A snake is a mathematical description of a contour used to delineate a boundary. Object segmentation is achieved by evolving the closed contour to the object's boundary such that the contour tightly encloses the object region [20]. This method is used in many biomedical applications where non-rigid amorphous and even non-existent edges occur [21]. Active contours can be applied either to single images, or to image sequences. In the latter, an additional layer of modelling is required to convey any prior knowledge about likely object motions and deformations as will be introduced in section 2.4.3. After initialising a contour either by a user or automatically, it is then moved until most of the contour points align with image edge points. Evolution of the contour is governed by an energy function described in equation 2-2. The energy function quantifies the 'goodness' of the contour - how smooth it is and how well localised it is with respect to the image edges.

$$E(\Gamma)=\int_0^1 E_{\text{int}}(\text{v}) + E_{ext}(\text{v})ds \qquad (2\text{-}2)$$

where s is the arc-length of the contour Γ, E_{int} characterises the curve itself e.g. degree of bending or size of curve, and E_{ext} characterises the image at the points where the snake is currently located e.g. measurement of edginess of the region through which the boundary passes [21]. The movement of the active contour is then motivated by minimising this energy. The external energy is supposed to be minimal when the snake is at the object boundary position while the internal energy is supposed to be minimal when the snake has a shape which is supposed to be relevant considering the shape of the sought object. The most straightforward approach grants high energy to elongated contours (elastic force) and to bended/high curvature contours (rigid force), considering the shape should be as regular and smooth as possible [22]. The contour can be initialised either by placing it outside the object region and shrinking until the object boundary is encountered if the image gradient is used in the energy function or either inside or outside the object so that the contour can either expand or shrink, respectively, to fit the object boundary if a region based method is used in the energy function. Besides the selection of the energy functional and the initialisation, another important issue is selecting the right contour representation but this level of detail is a subject of research in itself and hence outside the scope of this current thesis. As properties, the snake method can be described as being able to detect non-rigid objects even in situations when the video images are not captured from completely static cameras.

Another object detection method that is introduced next is the 'Point Detectors' method. Point detectors are used to find interest points associated with desirable properties in images. A desirable quality of an interest point can be its invariance to changes in illumination and camera viewpoint for example. There are several point detectors that exist such as the Moravec's interest operator, the Harris interest point detector and the SIFT detector, and each detector has its own way of qualifying a point of interest [23]. As an example, Moravec's operator computes the variation of the image intensities in a 4 × 4 patch in the horizontal, vertical, and diagonal directions and selects the minimum of the four variations as representative values for the window. A point is declared interesting if the intensity variation is a local maximum in a 12 × 12 patch [24]. An evaluation of the point detectors is given by Schmid et al. [25]. Based on their evaluation criteria of repeatability (comparing interest points on images taken under varying viewing conditions) and information content, they found that the Harris detector outperforms the others.

The final detection method to be discussed is one used in hyper-spectral imagery obtained from satellite remote sensing. In satellite imagery, a pixel can often represent more than one object because the spatial resolution of the camera on board the satellite is not strong enough. Spatial resolution refers to the size of the smallest object that can be resolved on the ground. As a result, a pixel can represent an area containing more than one object. To deal with these cases, the remote sensing community has developed techniques that allow very small objects down to sub-pixel sizes to be detected and since in this project sub-pixel objects are also involved, these detection algorithms are explained in the next chapter 3 with a view to adapting them for indoor environments.

2.4 Object tracking

After motion detection, monitoring systems generally track moving objects from one frame to another in an image sequence. Tracking is the problem of following image elements moving across a video sequence automatically and it can be defined as the task of estimating the trajectory of an object in the image plane as it moves around a scene. Tracking is an essential building block for vision systems in robotics, biomedical applications, military guidance applications and surveying. The tracking algorithms usually have considerable intersection with motion detection during processing and tracking over time typically involves matching objects in consecutive frames using features such as points, lines or blobs. A tracker tries to assign consistent labels to the tracked objects in different frames of a video. Additionally, depending on the tracking domain, a tracker can also provide information about the object such as orientation, area, or shape of an object. Tracking objects can be complex due to loss of information caused by projection of the 3D world on a 2D image, noise in images, complex object motion, non-rigid or articulated nature of objects, partial and full object occlusions, scene illumination changes, and real-time processing requirements [27]. Tracking algorithms in general consist of two main parts: correlation (matching problem) and target's trajectory estimation (motion problem) [26]. The matching problem requires a similarity metric to compare candidate pairs of image elements in the previous and current frame. Trajectory estimation is necessary for pursuing an object and to allow a system to move in advance and anticipate an object's movement.

There are three main categories of tracking algorithms [27]. These are the kernel tracking, point tracking and silhouette tracking methods, and each method can be further sub-divided depending on the approaches that are employed to achieve a particular method as shown in Figure 2-4. However it should be pointed out that this classification is not absolute in the sense that algorithms from different categories can be integrated together to obtain better tracking results in certain problem situations.

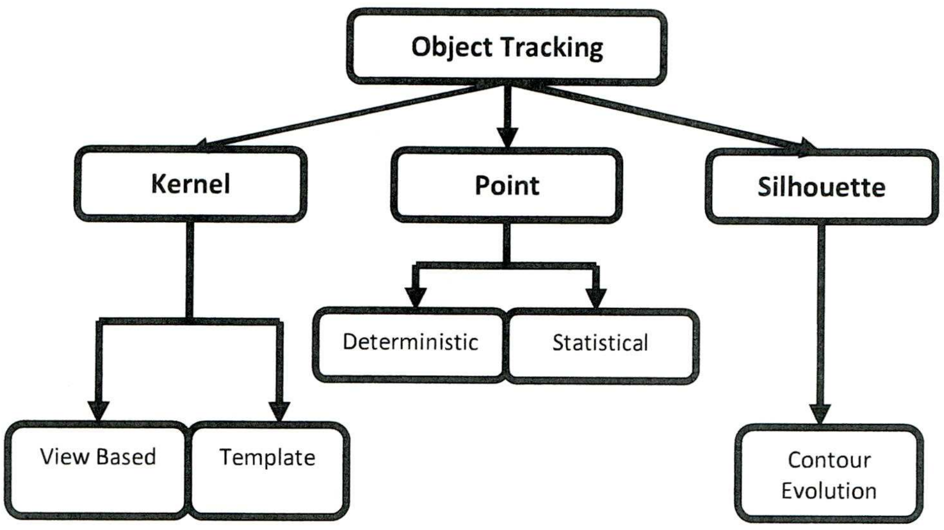


Figure 2-4 : Tracking categories and methods

2.4.1 Kernel Tracking

In kernel tracking, the word kernel refers to the object shape and appearance. For example, the kernel can be a rectangular template or an elliptical shape with an associated histogram. Objects are tracked by computing the motion of the kernel in successive frames. This motion is usually in the form of a parametric transformation such as translation, rotation, and affine or a dense flow field. The algorithms used in kernel tracking differ in terms of parameters such as the appearance representation used which can be either view based or template based models [27].

The view based approach involves constructing a small set of orthogonal basis images from a large set of training images that characterise the majority of the variation in the training set and can be used to approximate any of the training images [28]. For each $n \times m$ image in a training set of p images, a 1-D column vector is constructed by scanning the image from top to bottom and then left to right. Each of these 1-D vectors becomes a column in a $nm \times p$ matrix which is then decomposed using Principal Component Analysis (PCA) [29] to build a subspace of the training set called the eigen-space. A linear combination of these eigen-space representations can be used to reconstruct any of the training images approximately and tracking is then achieved by the parameterised matching between the eigen-space and the image. Hence, the object is not tracked by image motion but by differences in the eigen-space [28]. This approach enables tracking of an object to be maintained even if the latter's appearance model changes dramatically during the tracking process because different views of an object can be learned before the start (offline) of the tracking process but used during tracking. The disadvantage of this technique is that it is not very robust to noise, background clutter and situations involving occlusion.

An alternative algorithm in kernel tracking is the template based approach and more precisely the template matching approach. Template matching involves finding in the current image a region similar to the object template (usually in the shape of a square but there can be other shapes also) defined in the previous image of an image sequence by using a cross correlation similarity measure in a brute force search manner, i.e. shifting the target pattern over every location in an image [6]. The templates can be formed by using features such as image intensities, image gradients and colour histograms. One example is finding the mean colour of the pixels covered by the template as it moves around the image. The similarity between the object model, M, and the hypothesised position, H, is computed by evaluating the ratio between the colour means computed from M and H. The position which provides the highest ratio is selected as the current object location [30]. Another example is the combined use of a weighted histogram computed from a circular region and a mean-shift tracker to represent the object [18]. An advantage of the mean shift tracker over the standard template matching is the removal of brute force search together with its high computational demands but at the same time, this tracker requires that a portion of the object being tracked is inside the circular region upon initialisation [27]. The matching method, as defined so far, is not appropriate for the tracking of multiple objects. For this type of tracking, the whole image is modelled instead of just the object because modelling objects individually does not take into account the interaction among multiple objects and between objects and background during the course of tracking. An example interaction between objects can be one object partially or completely occluding the other. An object tracking method based on modelling the whole image as a set of layers is proposed by Tao et al. [31]. This representation consists of a single background layer and one layer for each object. Each layer consists of shape priors, motion models such as translation and rotation, and layer appearance. Layering is performed by first compensating the background motion modelled by projective motion such that the object's motion can be estimated from the compensated image using 2-D parametric motion. Then, each pixel's probability of belonging to a layer (object) is computed based on the object's previous motion and shape characteristics. The unknowns for each object are iteratively estimated until the layer ownership probabilities are maximised [27]. However the simultaneous estimation of the characteristics of each layer is very difficult and the authors instead estimate one set at a time while fixing the rest.

To summarise kernel tracking, an evaluation of this type of trackers can be obtained based on tracking single or multiple objects, ability to handle occlusion, requirement of training, type of motion model, and requirement of a manual initialisation. Motion of the object can be estimated by maximising the object appearance similarity between the previous and current frame and this estimation process can be in the form of a brute force search. To reduce the computational cost of this type of search, one can limit the object search to the vicinity of its previous position. With the object detected in this way, a possible next step is to use Kalman filtering or particle filtering (discussed in next section) to predict the location of the object in the next frame. It must be pointed out that the tracking described so far is essentially following a point in an image sequence and the benefit of using the Kalman filter is to smooth the object trajectory. The filter has very little effect on improving the object detection part which is crucial for achieving robustness of the tracker.

2.4.2 Point tracking

The second main tracking technique category is point tracking. In this technique, objects detected in consecutive frames are represented by points, and the association of the points is based on the previous object state which can include object position and motion. This approach requires an external mechanism to detect the objects in every frame. Point correspondence methods can be divided into two broad categories, namely, deterministic and statistical methods.

Deterministic methods involve the minimisation of a cost function formulated as a combinatorial optimisation problem which associates each object in frame $t - 1$ to a single object in frame t using a set of motion constraints [27]. These constraints which are usually described as being part of a correspondence problem are : 'proximity' which assumes the location of the object would not change notably from one frame to other, 'maximum velocity' which defines an upper bound on the object velocity and limits the possible correspondences to the circular neighbourhood around the object, 'path coherence' (smooth motion) which assumes the direction and speed of the object does not change drastically, 'common motion' which constrains the velocity of objects in a small neighbourhood to be similar and 'rigidity' which assumes that objects in the 3-D world are rigid, therefore, the distance between any two points on the actual object will remain unchanged [27]. One approach to solve the correspondence problem in a deterministic way is proposed by Sethi and Jain who used the path coherence constraint [32]. This algorithm considers two consecutive frames, and is initialised by the nearest neighbour criterion. It hypothesises the correspondences between the measurements and then repeatedly exchanges correspondences between trajectories that better match the optimisation criterion so as to minimise the cost. A modified version of the same algorithm, which computes the correspondences in the backward direction (from the last frame to the first frame) in addition to the forward direction, also exists. The disadvantages of this algorithm are that it assumes that there are no missing measurements by occlusion or otherwise, and second it does not allow for spurious measurements or false alarms. Another approach to solving the correspondence problem is one that makes use of the common motion constraint together with path coherence as proposed by Veenman et al. [33]. This approach can handle occlusion and misdetection errors, however, it is assumed that the number of objects is the same throughout the sequence, i.e., no object enters or exits.

The other approach to solve correspondence is by using statistical methods. Statistical correspondence methods use the state-space approach to model the object's properties such as position, velocity and acceleration. These methods treat the tracking problem as inferring the object's state by taking the measurement and the model uncertainties into account. Measurements usually consist of the object position in the image, which is obtained by a detection mechanism. In general, object motions undergo random perturbations and measurements obtained from sensors invariably contain noise. Statistical methods can be considered to be more appropriate to deal with these cases as they take the measurement and the model uncertainties into account during object state estimation [27]. The use of predictive filters, e.g. Kalman filter and particle filter, is the answer to deal with these cases.

2.4.2.1 State-space models and estimation

State-space models are a notational convenience for estimation and control problems [34]. The state of a dynamic system contains those variables that provide a representation of the internal condition or status of the system at a given instant of time [35]. A state-space model usually consists of two sets of equations, the system equations and the observation equations. The system (dynamic) equations model the dynamics of state variables based on external influences, such as input and noise, and the observation equations describe how measurement of state variables is done typically in the presence of noise [34]. The problem of state estimation concerns the task of estimating the state of a process while only having access to noisy and/or inaccurate measurements from that process. It is a very common problem setting, encountered in many disciplines within science and engineering. Predictive filters can be used to estimate the optimal state of a system [36]. In general, they use the mathematical model of the system dynamics to propagate the state's values and uncertainties, and they then combine this preliminary estimate with measurements made from observations. There are several predictive filters, each appropriate for a different type of uncertainty representation and dynamic modelling. To understand how predictive filters work, it is useful to understand some basic statistical concepts and then see how they are applied to deal with state estimation.

One way to describe a random variable is by making use of statistical indicators such as the mean and variance. The mean or expected value $E(X)$ of a random variable is given by $\sum_{i=1}^{n} P_i x_i$ for a discrete random variable's case where n possible outcomes and corresponding probabilities exist. This expected value is also known as the first statistical moment of the variable. The second moment, called variance, is a statistical indicator of great use given by $E\left|(x - E(x))^2\right|$. Variance is a measure of dispersion about the mean and it is a very useful statistical property for random signals because knowledge of the magnitude of the variance of a signal that was otherwise supposed to be 'constant' around some value (the mean) enables us to get a sense of how much jitter or 'noise' is in the signal. 'Mean' and 'variance' measures are used to represent random variables in a parametric way. If the data are of multivariate nature (i.e. consisting of multiple random variables), a covariance matrix can be obtained. This covariance matrix is of great importance in state-space modelling. The list of probabilities associated with each possible value of a discrete random variable is called its probability distribution. A special representation of a probability distribution known as the *Normal* or *Gaussian* distribution has historically been popular in modelling random systems for a variety of reasons. As it turns out, many random processes occurring in nature actually appear to be Normally distributed, or very close to a Normal distribution. In fact, under some moderate conditions, it can be proved that a sum of random variables with nearly *any* distribution tends towards a Normal distribution. The theorem that formally states this property is called the *central limit theorem* [37]. The standard normal distribution is the normal distribution with a mean of zero and a variance of one, and it is represented graphically by a familiar 'bell-shaped' curve. In the multivariate case, a Gaussian random variable is described by a mean vector and a covariance matrix. Moreover if a linear transformation is applied to the mean vector and covariance matrix, the resulting random variable will still be a Gaussian. This ease to just use matrix multiplications to propagate covariance and mean across linear transformations combined with the computationally efficient representation of random variables with just a vector and a matrix are used extensively and with great success in Kalman filters [35]. The characteristics of the Kalman filter will be explained in sub-section 2.4.2.2 but next here, the state-space model is described.

To understand this concept of state-space, consider a moving object in a scene. The information representing the object, e.g., location and speed, is defined by a sequence of states represented as state vectors $X_t : t = 1, 2, \ldots$ The change in state over time is governed by the dynamic equation,

$$X_t = \mathrm{F} \cdot X_{t-1} + W_t \qquad (2\text{-}3)$$

where W_t is a white noise process at $t = 1, 2, \ldots$ etc. A discrete-time white noise is a signal that does not have any correlation with itself except at present time, i.e. if W_t is a white noise process, then the random variable W_m is always uncorrelated with W_n unless $m = n$. Another definition of white noise is that its power is equal at all frequencies [35]. The relationship between the measurement and the state is specified by the measurement equation,

$$Z_t = \mathrm{H} \cdot X_t + N_t \qquad (2\text{-}4)$$

where N_t is another white noise process independent of W_t. F is a state transition matrix that relates X_t to its previous value X_{t-1} and H provides the linear connection between the unobserved state vector X_t and the measured vector Z_t. These two equations are often referred to respectively as the *process model* and the *measurement model*, and they serve as the basis for virtually all linear estimation methods. The goal is to estimate the state X_t based on the knowledge of the system dynamics and the availability of noisy measurements. The amount of information that is available to perform the estimation is problem dependent. If all the measurements up to and including time t are available to use in estimating X_t, then this is called an *a posteriori* estimate denoted by $\hat{X}_t^+$ [35]. If all the measurements before (but not including) time t are available for estimation, then the estimate obtained is called an *a priori* estimate denoted by $\hat{X}_t^-$. These two types of estimation are often referred to as prediction and correction respectively, and $\hat{X}_t^+$ is expected to be a better estimate than $\hat{X}_t^-$ because more information is used during the estimation process. If N measurements after time t are available to use in estimating X_t, then a *smoothed* estimate is obtained denoted by $\hat{X}_{t|t+N}$ [35].

To bring the state-space model into context with the tracking review being done here, if there is only one object in the scene, the state can be simply estimated by the two equations 2-3 and 2-4. If there are multiple objects in the scene then measurements need to be associated with the corresponding object states. For the single object case, if F and

H are linear functions and the initial state X_1 and noise have a Gaussian distribution, then the optimal state estimate is given by the simple Kalman Filter. In other cases when the process model is not linear and the errors are not assumed to be Gaussian, there exist other filters such as the Extended Kalman Filter, Unscented Kalman Filter and the particle filter that are more appropriate for these cases [35].

2.4.2.2 Kalman filters

The Kalman filter is named after Rudolf Kalman who first introduced the algorithm in 1960 [38]. It has been employed in various applications including process control systems, vehicle tracking, marine navigation, geology, demographic estimation and stock price prediction. It estimates the instantaneous state of a linear dynamical system perturbed by Gaussian white noise by using measurements that are linearly related to the system state but are corrupted by Gaussian white noise [39]. The filter minimises recursively the mean square estimation error without directly observing the system state or knowing the nature of the modelled system. One of the major characteristics behind the success of the Kalman filter is that it processes all available measurements, regardless of their precision, to estimate the current value of the variables of interest. It does it by using (1) the knowledge of the system and measurement device dynamics, (2) the statistical description of the system noise, measurement errors and uncertainty in the dynamical model, and (3) any available information about initial conditions of the variables of interest [40]. Figure 2-5 shows a schematic summary of a Kalman filter. A system is driven by some known controls and measuring devices provide the values of certain quantities. Knowledge of these system values is all that is explicitly available from the physical system for estimation purposes. The Kalman filter estimates the system state variables based on a signal and noise as input. The system state itself evolves with time under the effect of random perturbations or control inputs. The Kalman filter then provides an optimal estimate of the unobserved system states and their uncertainties based on noisy measurements of the process. The Kalman filter operates online so that the best estimate of the system state and its uncertainty can be computed by updating the previous estimates with new measurements.

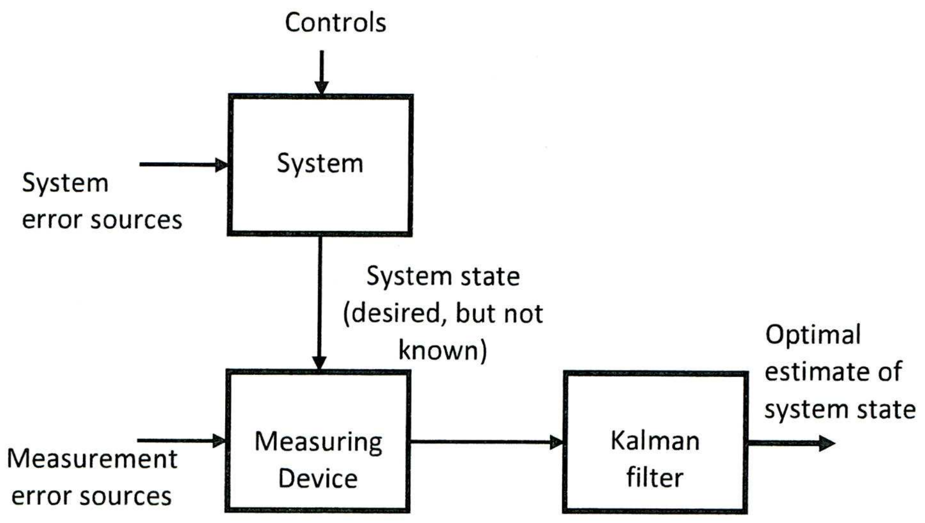

Controls
System
System error sources
System state (desired, but not known)
Measurement error sources
Measuring Device
Kalman filter
Optimal estimate of system state

Figure 2-5 : Typical Kalman filter diagram

To come back to what is contained in the algorithm itself and how the latter executes itself, the Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: *time update* equations and *measurement update* equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. The measurement update equations are responsible for the feedback, i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate. The time update equations can also be thought of as *predictor* equations, while the measurement update equations can be thought of as *corrector* equations [34]. To understand how the Kalman filter, a real case scenario is considered next. For example, consider the case of a ball moving on a field with some sort of sensor mechanism that can detect the location of the ball at certain points in time. The state vector X_t that best describes this motion is one that contains the position and the speed of the object being tracked in two-dimensional Cartesian coordinates at discrete time *t*. It can be defined as:

$$X_t = [x_t \quad y_t \quad vx_t \quad vy_t]^T \tag{2-5}$$

Assuming a constant velocity and white noise acceleration model is used, the filter can be initialised by using the two-point differencing method [41]. This method involves using two consecutive position measurements where the latest one is taken as the initial position estimate and differencing them gives the initial velocity estimate. At any instant in time, the new position (x_t, y_t) is the old position plus the velocity (vx_t, vy_t). This can be represented by equation 2-6 with Δ equal to the time interval between samples, and $vx = dx/d\Delta$ and $vy = dy/d\Delta$.

$$\begin{bmatrix} x_t \\ y_t \\ vx_t \\ vy_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ vx_{t-1} \\ vy_{t-1} \end{bmatrix} \tag{2-6}$$

Once the filter is running, an estimate for the next measurement can be obtained by using information from previous data. The measurement process is represented by a state measurement matrix H and its estimate m_t^- is given by:

$$\mathrm{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{2-7}$$

$$m_t^- = \mathrm{H} \cdot X_t \tag{2-8}$$

When an actual measurement is made, another vector m_t is obtained and the difference $(m_t - m_t^-)$ gives a new value called innovation. This new knowledge is assigned a weighting called the Kalman gain K_t using equation 2-9 where N_y represents the measurement noise and E_t is the covariance error matrix. The main role of the matrix K_t is to decide how much of the innovation should be used in later stages, i.e. a new state is estimated using the Kalman gain matrix and the innovation. In an extreme case when E_t is zero, the prediction is perfect and hence K_t is equal to zero too. The value of K_t is temporary and is only used in the next stage, i.e. it is not carried over for the next iterations of the filter, to perform optimal estimation as it tries to minimise the covariances. The error matrix E_t is also updated at this stage using the information from all previous measurements.

$$K_t = E_t \cdot \mathrm{H}^T \cdot \left(\mathrm{H} \cdot E_t \cdot \mathrm{H}^T + N_y\right)^{-1} \tag{2-9}$$

$$X_t = X_t^- + K_t \cdot \left(m_t - m_t^-\right) \tag{2-10}$$

$$E_t = (1 - K_t \cdot \mathrm{H}) \cdot E_t^- \qquad (2\text{-}11)$$

The tracking (or prediction) part of the filter is then performed when state X_{t+1} and new error matrix E_{t+1} are predicted by calculating equations 2-12 and 2-13.

$$X_{t+1} = \mathrm{F} \cdot X_t \qquad (2\text{-}12)$$

$$E_{t+1} = \mathrm{F} \cdot E_t \cdot \mathrm{F}^T + N_x \qquad (2\text{-}13)$$

$$\mathrm{F} = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2\text{-}14)$$

where F is the state dynamical matrix. N_x is called the process noise and it is represented by a covariance matrix with a small constant noise intensity parameter $\tilde{q}$. This parameter is related to the velocity fluctuations which have to be small compared to the actual nearly constant velocity of the object being tracked [41]:

$$N_x = \begin{bmatrix} \frac{1}{3}\Delta^3 & 0 & \frac{1}{2}\Delta^2 & 0 \\ 0 & \frac{1}{3}\Delta^3 & 0 & \frac{1}{2}\Delta^2 \\ \frac{1}{2}\Delta^2 & 0 & \Delta & 0 \\ 0 & \frac{1}{2}\Delta^2 & 0 & \Delta \end{bmatrix} \tilde{q} \qquad (2\text{-}15)$$

After equation 2-13, the filter returns to equation 2-9 for the next time interval and the whole filter equations are calculated again. When no measurement is made, only equations 2-12 and 2-13 are calculated as part of a prediction only stage. This description of the simple Kalman filtering process is summarised pictorially in Figure 2-6 and it is the basis of why Kalman filters are so useful in many tracking applications.

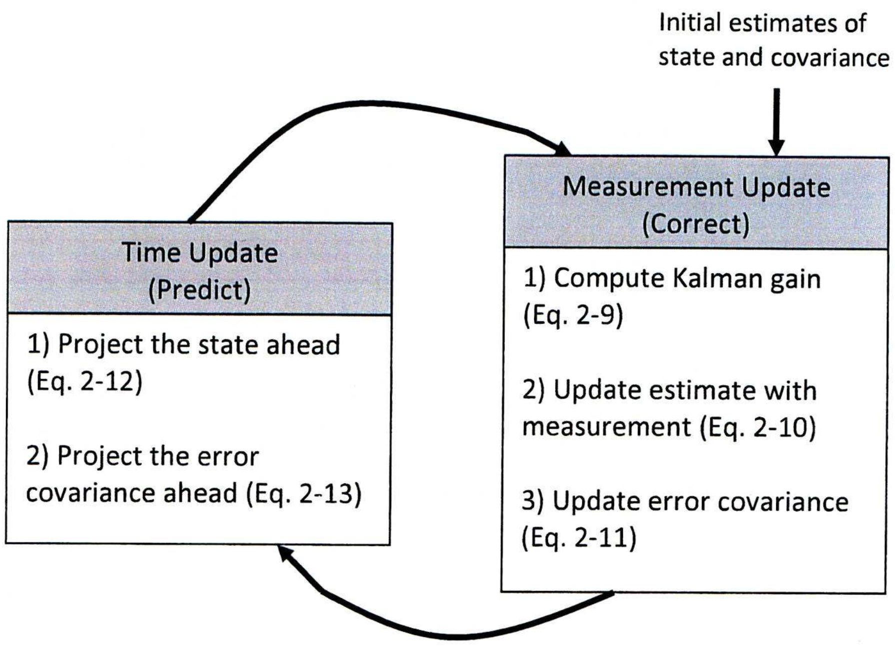

Initial estimates of state and covariance
Measurement Update (Correct)
1) Compute Kalman gain (Eq. 2-9)
2) Update estimate with measurement (Eq. 2-10)
3) Update error covariance (Eq. 2-11)
Time Update (Predict)
1) Project the state ahead (Eq. 2-12)
2) Project the error covariance ahead (Eq. 2-13)

Figure 2-6 : Complete picture of the operation of the Kalman filter

2.4.2.3 Other predictive filters

Many processes in real life are unfortunately not linear. One way to get round this is by making use of the Extended Kalman Filter (EKF). The EKF uses an initial linearisation stage in something akin to a Taylor series to linearise the estimation around the current estimate using the partial derivatives of the process and measurement functions to compute estimates even in the face of non-linear relationships [34]. It is important to note that a fundamental flaw of the EKF is that the distributions (or densities in the continuous case) of the various random variables are no longer normal after undergoing their respective nonlinear transformations. Another way to deal with non-linear systems is by using the Unscented Kalman Filter (UKF) technique. Instead of linearising the functions, this is a transform that uses a set of points, and propagates them through the actual nonlinear function, eliminating linearisation altogether [42]. The points are chosen such that their mean, covariance, and possibly also higher order moments, match the Gaussian random variable. Mean and covariance can be recalculated from the propagated points, yielding more accurate results compared to ordinary function linearisation.

If the noise processes are non-Gaussian or if the degree of non-linearity is significant, then another type of filter, called the particle filter, can be used. Sometimes a Gaussian random variable is not able to describe, not even in an approximate way, the state of a system. For example, a Gaussian random variable is not able to represent simultaneously two strong groups of possibilities or a time series signal with both abrupt and gradual changes. One possible approach to deal with this limitation is to use a particle-based, or sample-based, representation of the probability density function [43]. A particle-based representation of a distribution is not described by a few parameters; instead it is described by a large number of samples of the state space. In Kalman filter approach, time and effort are focused on how different state coordinates or multiple models can be used to limit the approximations. In contrast to this, the particle filter approximates the optimal solution numerically based on a physical model, rather than applying an optimal filter to an approximate model [44].

The particle filter, also known as the Monte Carlo filter or sequential importance sampling filter, is a probability-based estimator of that is very effective for non-linear systems [35]. To understand how it works, consider equations 2-3 and 2-4 used before to describe the state-space model. The difference this time is that F and H are non-linear function mappings describing the evolution of the state transitions and relationship between the unobserved state and the measurement. The goal of the filter is to approximate the conditional probability density function (pdf) of X_k based on the measurements $Z_1, Z_2, \ldots, Z_k$, i.e. a pdf $p(X_k \mid Z_k)$. Assuming that the pdf of the initial state $p(X_0)$ is known, a set of N samples known as particles are generated. These particles are denoted by $X_{0,i}^{+} (i = 1, \ldots, N)$ where the parameter N is chosen by the user as a trade-off between computational effort and estimation accuracy [35]. Then at each time instant k, the following algorithm is performed:

- The *a priori* particles $X_{k,i}^{-}$ are obtained by using process equation 2-3 for each value of i as part of the time propagation step.

- The relative likelihood q_i of each particle is calculated conditioned on the measurement Z_k. This is done by evaluating the pdf $p(Z_k \mid X_{k,i}^-)$ on the basis of the non-linear measurement equation 2-4 and the pdf of the measurement noise.
- A set of *a posteriori* particles $X_{k,i}^+$ are generated on the basis of the relative likelihoods q_i is a step called the resampling step.
- Given the fact the *a posteriori* particles are now distributed according to the pdf $p(X_k \mid Z_k)$, the desired statistical measures of this pdf such as the mean and covariance can then be calculated.

The resampling step is one very important part of this algorithm and several ways to perform this step exist. An additional resampling is sometimes employed to eliminate samples with very low weights. The efficiency and accuracy of the particle filter depend mainly on two key factors: the number of particles used and the resampling function used to re-allocate these particles at each iteration.

As a summary of these predictive filters, one can say that for a linear system with Gaussian noise, the Kalman filter is optimal. In a non-linear system, the particle filter may give better results than the Kalman filter with additional computational effort. The Kalman filter performs poorly in systems with non-Gaussian noise and a particle filter will normally give better results in these situations. For these systems, the UKF provides a balance between the low computational effort of the Kalman filter and the high performance of the particle filter, as shown in Figure 2-7 [35]. When deciding which filter to choose for a particular application, one should always first try to model a system as a linear Gaussian system, and use the Kalman filter. If the results are not satisfactory, and the problem is thought to lie in either the linear dynamics or the Gaussian description of the random variables, then one can proceed to use more complex distribution representation and predictive techniques. The correct characterisation of the distribution of the observation's noise is essential for the quality and correctness of the final state estimation.

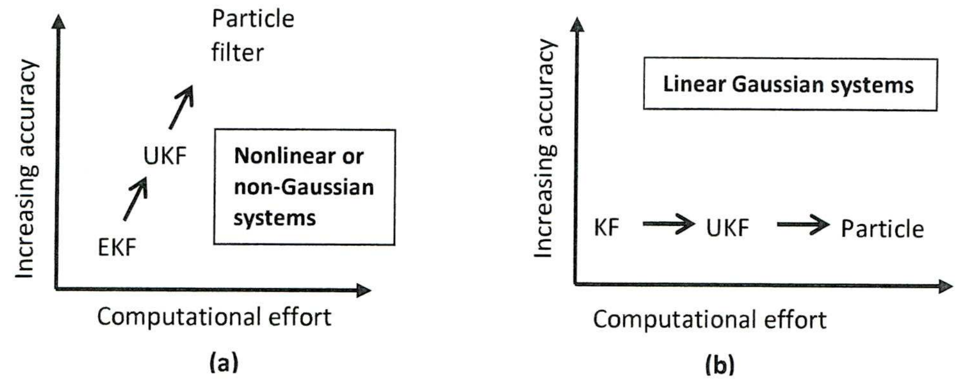


Figure 2-7 : Performance of the filters in (a) non-Gaussian systems (b) Linear Gaussian systems [35]

2.4.2.4 Multiple object tracking

The filters described in the previous sub-section assume a single measurement at each time instant, that is, the state of a single object is estimated. However there are cases when more than one object needs to be tracked. When tracking multiple objects with Kalman or particle filters, the most likely measurement for a particular object must be associated deterministically to that object's state, that is, the correspondence problem needs to be solved before these filters can be applied [27]. The simplest method to perform correspondence is to use the nearest neighbour approach. However, if the objects are close to each other, then there is always a chance that the correspondence is incorrect. The Joint Probability Data Association Filtering (JPDAF) is a popular approach to tracking multiple moving objects [45, 46]. It involves the calculation of a Bayesian estimate of the correspondence between features detected in successive frames of the sensor data and the different objects to be tracked. Kalman filters are then used to estimate the states of the individual objects. A major limitation of the JPDAF algorithm is its inability to handle new objects entering the field of view (FOV) or already tracked objects leaving the FOV [27]. The *Multiple Hypothesis Tracking* (MHT) algorithm, originally developed by Reid [47], does not have this shortcoming. Instead of trying to establish a motion correspondence using only two frames, the correspondence decision is deferred until several frames have been examined. The MHT algorithm maintains several correspondence hypotheses for each object at each time frame. The final track of the object is the most likely set of correspondences over the time period of its observation. The algorithm has the ability to create new tracks for objects entering the FOV and terminate tracks for objects exiting the FOV. It can also handle occlusions, that is, continuation of a track even if some of the measurements from an object are missing. MHT is an iterative algorithm whereby each iteration begins with a set of track hypotheses. Each hypothesis is a collection of disjoint tracks. For each hypothesis, prediction of each object's position is made for the next frame, and then the predictions are compared with actual measurements by evaluating a distance measure. A set of correspondences (associations) are established for each hypothesis based on the distance measure, which introduces new hypotheses for the next iteration. Each new hypothesis represents a new set of tracks based on the current measurements. Note that each measurement can belong to a new object entering the field of view, a previously tracked object, or a spurious measurement. Moreover, a measurement may not be assigned to an object because the object may have exited the field of view, or a measurement corresponding to an object may not be obtained [47]. The latter happens when the object is occluded or it is not detected due to noise. The MHT algorithm is computationally exponential both in memory and time and algorithms have been developed by Cox and Hingorani to reduce this computational cost [48].

To summarise point tracking technique, point trackers are suitable for tracking small objects which can be represented by a single point (single point representation). Multiple points are needed to track larger objects. Points are detected in consecutive frames and they are made to correspond to each other by two main categories, namely, deterministic and statistical methods. The deterministic method involves the minimising a cost function formulated as a combinatorial optimisation problem which associates each object in frame $t - 1$ to a single object in frame t using a set of motion constraints. Statistical methods use the state space approach to model the object properties such as position, velocity and acceleration and they also take uncertainties into account. The Kalman filter is the most common algorithm used in approaching the state-space model solution but for certain situations, other predictive filters such as the EKF, UKF and the particle filter are more suitable.

2.4.3 Silhouette tracking

The final tracking category to be described in this literature review is silhouette tracking. This method has been developed to deal with objects that may have complex shapes, for example, hands, head, and shoulders that cannot be well described by simple geometric shapes. Tracking is performed by estimating the object region in each frame and the information encoded inside the object region is used. This information can be in the form of appearance density and shape models which are usually in the form of edge maps. Given the object models, silhouettes are tracked by a method called contour evolution which can be considered as object segmentation applied in the temporal domain using the object models generated from the previous frames. The goal of a contour based object tracker is to find the boundary between the object and the background in each frame, such that the object region is tightly enclosed within the contour. There are two main ways of achieving this; first is by using state space models to evolve the contour and the second is by performing direct minimisation of a contour energy function.

In the probabilistic state space approach, the object's state can be defined in terms of the shape and the motion parameters of the contour. The state is updated at each time instant such that the contour's *a posteriori* probability is maximised. This probability depends on the prior state and the current likelihood, which is usually defined in terms of the distance of the contour from observed edges. An example is the one developed by M. Isard whereby they define the object state in terms of spline shape parameters and affine motion parameters [49]. The measurements consist of image edges computed in the normal direction to the contour and the state is updated using a particle filter. In order to obtain initial samples for the filter, the state variables are computed from the contours extracted in consecutive frames during a training phase. During the testing phase, the current state variables are estimated through particle filtering based on the edge observations along normal lines at the control points on the contour. Another example is proposed by Chen et al. [50]. In this implementation, the contour is parameterised as an ellipse and each contour node has an associated Hidden Markov Model (HMM). An HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols [51]. The state of each HMM is defined by the points lying on the lines normal to the contour control point, with the hidden states of the HMM being the true contour points of each normal line. The observation likelihood of the contour depends on the background and the foreground partitions defined by the edge along the normal line on contour control points. The state transition probabilities of the HMM are estimated using the Joint Probability Data Association Filtering [50].

These two examples represent the contours using explicit representation, e.g. a spline, but explicit representations do not allow topology changes such as region split or merge. A contour tracking method based on direct minimisation of the energy functional can, for its part, use the implicit representations and allow topology changes. In this method, algorithms evolve the contour onto the object region by minimising the energy functional and the contour energy is computed using temporal information in the form of the temporal gradient which is obtained by calculating the optical flow. There are some similarities with the active contour technique described previously in section 2.3 and more information about the technicalities of the current algorithms is given by Mansouri et al. [52] and Bertalmio et al. [53].

To summarise the silhouette tracking category, one can say that the most important advantage of a contour tracker is that it can model a large variety of object shapes. Contours are represented by explicit (control points and splines) or implicit (level sets) representations with the use of these representations depends on the context of the application. Contour trackers are employed when tracking the complete region of an object is required. Qualitatively, the contour based methods can be compared on the basis of requirement of training and occlusion handling. Moreover some algorithms only use information on the contour boundary for evolution while others use the complete region. Occlusion handling is another important aspect of silhouette tracking methods. Usually methods do not address the occlusion problem explicitly. A common approach is to assume constant motion or constant acceleration where, during occlusion, the object silhouette from the previous frame is translated to its hypothetical new position. Another important issue related to silhouette trackers is their capability for dealing with object split and merge. For instance, while tracking a silhouette of a person carrying an object, when the person leaves an object, a part of the person's contour will be placed on the left object (region split). These topology changes of region split or merge can be handled well by implicit contour representations [27].

2.4.4 Small object tracking techniques

In addition to these three categories of tracking techniques, another 'category' is introduced next which deals with the detection of very small objects, i.e. objects occupying less than a pixel to a few pixels on an image plane. It is not included as a proper tracking category in this literature review in the same breath as the three described so far because several techniques are used with the aim to satisfy the single objective of detecting a small object or target. This type of detection has certain specific challenges such as the difficulty to identify objects by their shape and spatial detail, and the increased impact of noise. The detection of small objects is important in some applications, e.g. biomedical experiments, vehicle tracking for transportation systems or target tracking for military purposes. As an example, one of the most difficult goals of Automatic Target Recognition (ATR) is to spot incoming targets at long range where the motion is small and signal to noise is poor, and to be able to track such targets long enough to identify whether the target is approaching in a dangerous manner or not [54].

One approach is to exploit the useful properties of wavelet filters to provide detection of the motion of these small, low-contrast objects [54]. Wavelet filters have been shown to be superior than Fourier methods for detecting localised high frequency behaviour in an otherwise predominantly low frequency signal [55]. A wavelet transform is a mathematical function used to divide a given function, e.g. an image, into different frequency components called wavelets which are of limited duration [15]. The approach that is adopted in this ATR system performs a *temporal* filtering of the image sequence using a wavelet filter rather than a spatial filtering across the image. By combining evidence over several frames coherent motion stands out above the noise, providing a much higher signal to noise ratio. For each filtered image, the absolute values of the wavelet coefficients are used to estimate potential positions of objects. This is then thresholded and passed through a morphological opening process (erosion followed by dilation) which removes isolated noise regions [15], leaving just the targets and noise regions. Depending on the pixel connectivity criterion used [15], a set of connected regions can be obtained and is then passed to a Kalman filter for tracking [54]. One main drawback of this method is that the targets can become smeared over a larger region of the images if the wavelet filters used are not chosen correctly.

In the biomedical field, single particle tracking is the use of computer analysis of video images to follow the sub-micron motion of micro-organisms under microscopic observation [56]. Several techniques have been developed to track these very small elements which very often have dimensions less than the resolution of the microscope. One such technique is the direct fitting of Gaussian curves to intensity profiles [57]. As the fluorescent images of these particles are diffraction-limited spots with a Gaussian intensity distribution, fluorescent spots in each image can be located by an image analysis routine which identifies peaks having approximately the diffraction limited width. One disadvantage of this technique is that it performs poorly when the spots to be detected are closely spaced [57]. Another technique is one based on correlation [58]. This method compares an image *I* to a kernel *K* which contains the object being tracked. *K* is shifted relative to *I* in one-pixel increments. For each increment, a correlation value is calculated that describes how well the values in *K* match those of the underlying image, *I*. At the relative shift where *K* and *I* are most similar, one finds a maximum in the correlation matrix. However, correlation tends to match the brightest regions of two images rather than the best topographical fit, resulting in errors in some cases, and it is also more computationally intensive than the Gaussian fitting technique [56]. Another example based on correlation is the Sum-Absolute Difference (SAD) technique [59]. SAD determines the translation of *I* relative to *K* that minimises the sum of absolute differences between the overlapping pixels. It has been used in tracking of speckles and it is also simple to implement in digital hardware as it requires only a single difference operation [60]. Other techniques that deal with low resolution imagery can also be adapted to deal with tracking of small objects. One very good example is the super-resolution technique which uses many low resolution frames in an image sequence to increase the general quality of the images and hence make object detection easier [61].

2.5 Behaviour recognition

Monitoring systems usually come with some kind of human behaviour recognition facility. Most of these systems work on human motion recognition in the context of walking recognition or simple activity detection in limited known spaces. There are two main approaches that are commonly used for this purpose. These are the template matching and the state-space techniques respectively [62].

In template matching, human behaviours are characterised by patterns in the first place. Then, the features of a person are extracted from image sequences and matched with the pre-defined patterns to recognise the person's behaviour. For example, Bobick & Davis [63] generate motion-energy images (MEI) and motion-history images (MHI) from an image sequence to interpret motion. The MEI represents the changed pixels in the image sequence. The motion images in a sequence are calculated by differencing successive frames and then thresholding into binary values. These motion images are accumulated in time to form the MEI, which are binary images containing motion blobs. The MHI is computed from the MEI where moving pixels have higher intensities. The template of each behaviour consists of MEI and MHI derived from training examples of the behaviour viewed from different angles. The behaviour of a person is detected by matching the MEI and MHI of the image sequence to the behaviour templates. Although template matching techniques are computationally inexpensive, they are sensitive to the variance in the movement duration and in the various patterns of the same activity that exist [62].

The state-space approach on the other hand defines each static posture as a state and motion sequence as a composition of these states [62]. The states together with the transitions between the states form part of a deterministic model called a Finite State Machine [64]. Under such a scenario, duration of motion is no longer an issue because each state can visit itself repeatedly. This approach defines a set of states for a single person or multiple people. Each behaviour is represented by a model, which consists of some states and the probabilities of transitioning from one state to another. From the video sequence, a sequence of states is extracted. Then, this sequence is matched with the predefined behaviour models to detect the corresponding behaviour of single or multiple people. In the system developed by Ayers et al. [64], an accurate description of the layout of the scene is needed. It uses prior knowledge about the layout of the room such as entrance and exit points, about the location of objects of interest such as telephones and beds, and about how certain objects are used and what sorts of actions human beings can do. This knowledge allows the system to use context to help make decisions about which actions are occurring in a room and hence reduce unnecessary computation. For more complex behaviours, the current deterministic model has its limitations as it cannot account for uncertainties. To deal with these, probabilistic models in human behaviour recognition can be used [62].

2.6 Summary

This chapter is an attempt to provide a survey on the various tracking techniques that exist and examples of where they are being applied are also given. However one has to appreciate that this is not an exhaustive list and various other combinations of object detection and tracking exist. The first stage of a monitoring or surveillance system is to detect the object/s by using various segmentation techniques. Several detection techniques have been discussed with each one used because the requirements are different. One detection technique that is not introduced in this chapter but which is of great importance to this thesis is one used in hyper-spectral imagery obtained from satellite remote sensing images to detect small objects. This technique is given more attention in the next chapter 3 with a view to adapting it for detection in indoor environments. Another stage of a monitoring system that is of upmost importance to this project is the actual tracking of an object of interest. Several techniques were described here with special attention given to the State-Space model and Kalman filters. A state-space model usually consists of two sets of equations, the system equations and the observation equations, and predictive filters such as the Kalman filter are very powerful tools in solving these equations. This is implemented and demonstrated in later chapters. The final section in this chapter talked briefly about the contribution of human behaviour recognition in the development of a visual monitoring system. The next chapter gives more information about the other parts of the monitoring system such as the image sensors to be used, the colour spaces to be chosen and the detection mechanisms to be implemented.

Related information on processing techniques

CHAPTER 3

3. Related information on processing techniques

3.1 Introduction

The previous chapter gave a description of various object detection and tracking techniques that are currently used in monitoring systems. In this chapter, information more relevant to this thesis is given starting from how an image sequence is captured from a low resolution colour sensor, followed by an examination of the theory behind colour spaces and ending with the use of pixel un-mixing techniques on low resolution images to detect the presence of objects. The main types of sensors that need to be investigated to fit in the generic monitoring system diagram (Fig 2-1) are digital image sensors, and particularly those that allow colour to be sensed. Several low resolution sensor technologies exist and each one has its own advantages and disadvantages depending on the manufacturing process and the intended use. The ability to sense colour is usually incorporated by selecting three specific light wavebands from the visible electromagnetic spectrum and mixing them together. The selection can be done by using filters. After a colour image is recorded, there are several ways to represent that colour. These representations, known as colour spaces, have been developed over the years to address specific requirements. In this thesis, one of the requirements is to try to achieve colour constancy, which is the ability of a vision system to diminish or in the ideal case remove, the effect of a changing illumination. The colour distribution of an object in a scene differs under different lighting conditions and even under the same lighting conditions, background colours and shadows may influence colour values that are recorded. Furthermore, if an object is moving, the apparent colour values change as the object's position relative to the camera or light changes. Another requirement before selecting the colour space is to make sure it is compatible with the un-mixing technique to detect small objects as described in sections 3.4 and 3.5.

3.2 Image sensors

Sensors differ by their modes of operation, operating frequencies, nature of output signals, and resolutions. There are two basic modes of operation: (1) Active sensors, e.g. microwave radars, provide their own source of target illumination, and they are equipped with both a transmitting and a receiving system. (2) Passive sensors do not provide target illumination and depend on the surrounding environment for detection. Passive sensors, e.g. photographic cameras that use reflected light energy as their source of 'target' illumination, consist of only a receiver system. Optical and infrared sensors can also be used to produce multi-spectral images using various light wavelengths (or frequencies) to produce these images.

An image sensing device can be defined as being one that converts an optical image to an electric signal when incident light hits it. Figure 3-1(a) shows the components of a single sensor with a light sensitive material called a photodiode or photosite. The filter is used to select and allow certain wavebands of light only to reach the photodiode. The latter records the intensity or brightness of the light that falls on it by accumulating a charge; the more light, the higher the charge and hence the output voltage waveform obtained is generally proportional to the amount light entering the diode [15]. In order to capture a 2-D image, each of these individual sensors can be arranged in the form of a 2-D array (or rows and columns if the matrix notation is used) as shown in Figure 3-1(b). The brightness recorded by each photosite is then stored as a set of numbers that can then be used to set the brightness of dots on a screen or density of ink on a printed page to reconstruct the image. Each photosite can represent one pixel in the captured image, and the higher the resolution, as specified by the number of pixels, the sharper the images will be. An image sensor is typically a charge-coupled device (CCD) or a complementary metal–oxide–semiconductor (CMOS) active-pixel sensor. Before going on to describe the functioning and benefits of these two types of image sensors, it is good to know what characteristics make a sensor good. Some of the most important characteristics of a sensor are its linearity, sensitivity, signal-to-noise level, wavelength response, charge transfer efficiency and size [66].

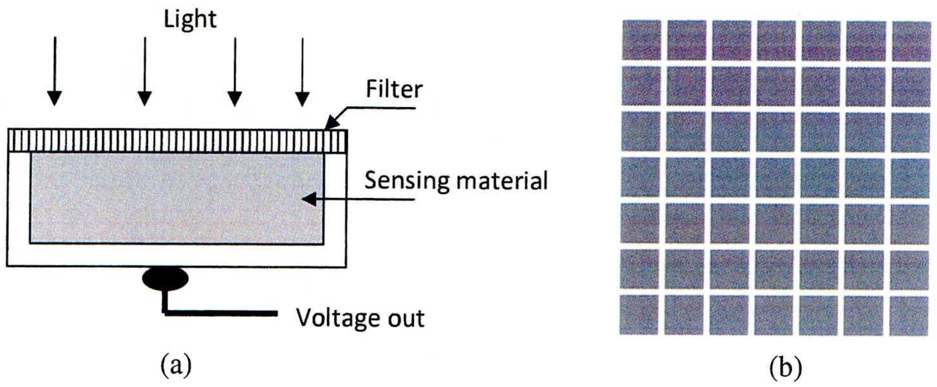


Figure 3-1 : (a) Basic single imaging sensor arrangement (b) Layout of photo-sensitive sites as an array (or 'matrix') of sensors.

Linearity or linear response refers to the sensor's reaction to varying levels of radiation intensity. This response is usually called *gamma* to describe the slope of a response curve when the sensor response is plotted against the radiation intensity [66]. A gamma of 1 corresponds to a sensor with a linear response to radiation. A gamma of less than 1 corresponds to a sensor that compresses the dark end of the range, while a gamma greater than 1 corresponds to a sensor that compresses the bright end.

Another characteristic of a sensor that needs to be taken into consideration is its sensitivity. A sensor requires a discrete amount of time in which to accumulate enough photons to generate a strong signal. Sensitivity is the measure of a sensor's dynamic response to scene brightness [66], and the response curve for a light sensitive sensor can be divided into three parts: the dark area, the linear area and the saturation area. The dark area of the response curve is concerned with the sensor's response to very low light. The output of the sensor in the dark area is very low, is noisy and is unpredictable. As the light falling on a sensor is increased gradually, the sensor's response becomes almost linear whereby the output of the sensor begins to increase predictably as the amount of light increases. The area of linearity is also called the dynamic range of a sensor [66], which is a measure of the maximum and minimum intensities that can be simultaneously detected in the same field of view. The output remains linear until a stage called the saturation point. An increase the light intensity beyond this point results in a non-linear increase in the output of the sensor, as shown in Figure 3-2. If some pixels of an image sensor are in the saturated zone, the charge caused by any additional photons (light) will overflow and have no effect on the pixel values, resulting in clipped or overexposed pixels. Blooming occurs when this charge flows over to surrounding pixels, brightening or overexposing them in the process [66].

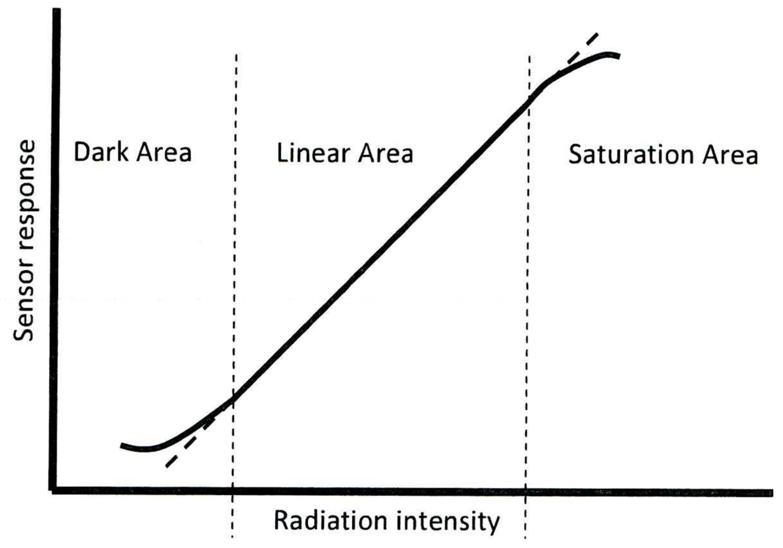


Figure 3-2 : Sensor response curve

The signal-to-noise ratio (commonly abbreviated S/N or SNR) of a sensor is also an important feature. Several factors contribute to sensor noise: the non-linear characteristics of the analogue-to-digital converter, electronic noise from other components and high-frequency clocks, degradation of the amplifier circuitry [66]. SNR is the comparison measurement of the incoming light signal versus the various inherent or generated noise levels and is a measure of the variation of a signal that indicates the confidence with which the magnitude of the signal can be estimated. In general, the larger the SNR, the better a sensor is. Sometimes in images with high background signals, the contrast signal to noise ratio (CSNR), which measures the ratio of the contrast information level of distorted signal to the contrast level of the error signal, is a more relevant measure [67]. Two other characteristics that apply to image sensors are spatial resolution and frame rate. Intuitively, spatial resolution can be considered to be a measure of the smallest discernible detail in an image [15]. Digital image sensors have finite minimum regions of detection (known as pixels) that set a limit on their spatial resolution. Spatial resolution is also affected by other factors such as the quality of the lens or imaging system. Contrast is an important factor in resolution as high contrast objects (e.g. black and white lines) are more readily discerned than low contrast objects (e.g. adjacent grey lines). On the other hand, the frame rate of a digital image sensor is the fastest rate at which subsequent images can be recorded and saved. This is usually measured in frames per second (fps) and it can have a range of values depending on the intended use, for example an image sensor system monitoring the growth of a plant needs far less frames per second than one used in the guidance of an autonomous vehicle. There is often a trade-off between the rate of measurement of an image sensor and its spatial precision if the amount of processing power available is limited.

3.2.1 Examples of digital image sensors

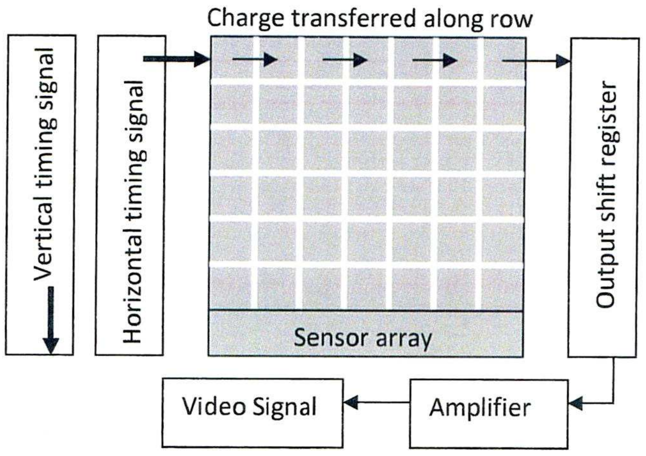


Figure 3-3 : CCD transports the charge across the chip and reads it at corner of the array [68].

The charge coupled device (CCD) and the complementary metal oxide semiconductor (CMOS) active pixel sensor are two very common technologies used in image sensors. Both image sensors convert light into electrical charges in almost the same way. The main differences occur in the manipulation of the charges. Upon receiving a timing signal, each sensitive element in the CCD transfers its contents to the adjacent element in the same row and the charge is then read at one corner of the array as shown in Figure 3-3. An *analogue-to-digital converter* (ADC) then turns each pixel's value into a digital value by measuring the amount of charge at each photosite and converting that measurement to binary form. The more advanced functions, such as the clock drivers, timing logic, and signal processing are normally put on separate chips which mean CCD sensors contain several chips.

A CMOS sensor, on the other hand, includes transistors at each photosite, and every pixel can be read individually, much like a computer's random access memory (RAM) chip. It is not necessary to sweep all the pixels to one location, and, unlike CCD sensors, with which all their information is processed externally to the sensor, each CMOS pixel can be processed individually and immediately. That allows the sensor to respond to specific lighting conditions faster [69]. In other words, some image processing can be done within the CMOS sensor itself, something that is impossible with CCD devices. The circuitry found in a CMOS sensor is similar to that in standard chips such as RAM and hence CMOS sensors can be produced using the same equipment and production lines, in contrast to CCD chips which require special fabrication methods. As a result CMOS sensors can be relatively inexpensive compared to CCD on a pixel-by-pixel basis. CMOS image sensors can incorporate other circuits on the same chip, eliminating the many separate chips required for a CCD. However CMOS sensors perform badly in low light conditions [69]. Their sensitivity to low light is decreased because part of each photosite is covered with the circuitry responsible for the basic image processing mentioned before. The percentage of a pixel devoted to collecting light is called the pixel's *fill factor*. CCDs have a 100% fill factor but CMOS have much less. The lower the fill factor, the less sensitive the sensor is and the longer exposure times must be. CMOS is the technology of choice for high-volume, space-constrained applications where high image quality requirements are not necessary, e.g. security cameras and bar-code scanners. CCD offers superior image quality and flexibility at the expense of system size, and it is the most suitable technology for high-end imaging applications, such as digital photography, broadcast television, and most scientific and medical applications. However several manufacturers are working on improving CMOS sensors and the gap in image quality between CMOS and CCD is expected to decrease [69].

The basic arrangement of photosites as shown in Figure 3-1 (b) cannot capture colour information as it is. It only keeps track of the total intensity of the light that strikes its surface. In order to get a full colour image, most sensors use filtering to record light in its three primary colours, i.e. red, green and blue. This collection of intensity values can be modelled as layers that when combined will contain the necessary information to produce a colour image. Colour theory is explained in the next section.

3.3 Colour spaces

One important step in the development of this monitoring system is to choose the right colour space. Although colour images are acquired by combining signals coming from wavelengths of light representing roughly red, green, and blue, there are many ways in which this colour information can be used. For human beings, colour is a subjective perceptual experience of light in the visible region of the electromagnetic spectrum. Colour can also be used to define the characteristic of a visible radiant energy itself as defined by Wyszecki and Stiles [70]: "*Colour is that characteristic of visible radiant energy by which an observer may distinguish differences between two structure-free fields of view of the same size and shape, such as may be caused by differences in the spectral composition of the radiant energy concerned in the observation*". The eye contains sensors called cone cells that are responsible for colour vision. Experimental evidence has established that the 6 to 7 million cones in the human eye can be divided into three principal sensing categories corresponding roughly to red, green and blue, which are usually referred to as the primary colours [15]. Colour perception is a phenomenon that is not only dependent on the eye but also higher-level processes in the human brain. The International Commission on Illumination, mostly referred to as CIE for its French translation 'Commission Internationale de l'Eclairage', set up a specification, called the CIE XYZ space, as an attempt to parameterise this very complex nature of colour in 1931. Conceptually, the CIE experiments involved getting a certain number of people to match spectral colours (monochromatic light) using a colour made by adding varying proportions of red, green and blue 'primaries', or the *tristimulus* values *X, Y*, and *Z* as they are more formally described by CIE. A particular colour is then specified by its trichromatic coefficients *x, y*, and *z* which are the normalised tristimulus values, i.e.

$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad \text{and} \quad z = \frac{Z}{X+Y+Z} \qquad (3\text{-}1)$$

As such $x + y + z = 1$ and hence only two values are needed to specify a particular colour. This gave rise to the CIE chromaticity diagram shown in Figure 3-4. The positions of the various spectrum pure colours – from violet at 380 nm to red at 780 nm – are indicated around the boundary of the tongue-shaped diagram. Any point within the boundary represents a colour that consists of a mixture of pure spectrum colours. The point of equal energy, E, corresponds to equal fractions of the three primary colours and represent the CIE standard for white light. A point located on the boundary is fully saturated and as a point leaves the boundary to move towards E, more white light is added to the colour. It becomes less and less saturated until it reaches zero saturation at point E [15].

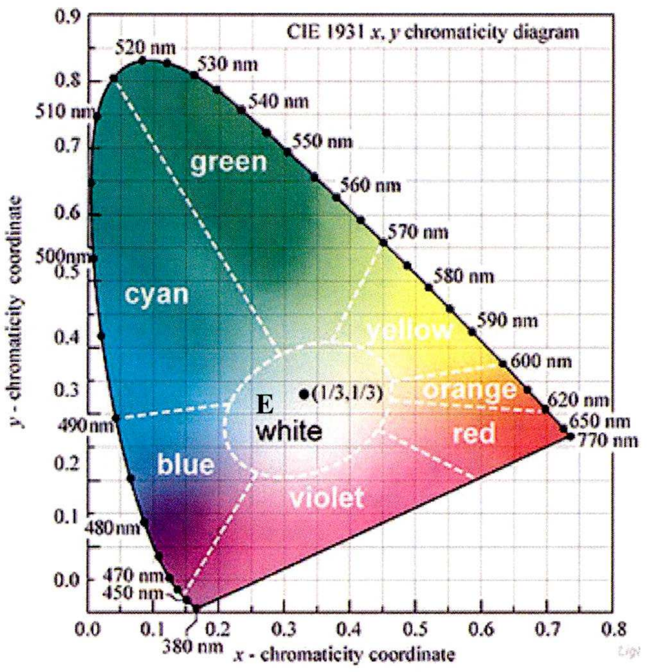


Figure 3-4 : CIE 1931 Chromaticity diagram. The equal-energy point E is located at the centre and has coordinates $(x,y) = (1/3, 1/3)$ [71].

Based on the CIE standard chromaticity diagram, several colour spaces have been developed to facilitate the specification of colours in a generally accepted way for digital images. As humans, we may define a colour by its attributes of brightness, hue and saturation but a computer screen will produce a colour picture in terms of the excitations of red, green and blue phosphors on a CRT faceplate or LEDs in newer displays. A colour space is an abstract mathematical model describing the way colours can be represented as a sequence of numbers, typically as three or four values of colour components. The colour models are oriented towards hardware (such as monitors) or towards applications where colour manipulation is a goal (such as animation graphics). There exist several colour spaces and the most common ones are discussed next.

3.3.1 The RGB colour space

One of the most common colour spaces is the *Red-Green-Blue* (RGB) colour space. RGB is a linear colour space that formally uses single wavelength primaries, 645.16 nm for Red, 526.32 nm for Green and 444.44 nm for Blue [72]. This model is based on a Cartesian coordinate system which is best shown as a cube with the primary colours at three corners, three secondary colours cyan, magenta and yellow at three other corners, black at the origin and white at the corner furthest from the origin. It is an additive colour model in which red, green, and blue light are added together in various ways to reproduce a broad array of colours. For 8-bit images, each pixel can be represented as having three values of the three primary colours in the range between 0 and 255. The brightness value I at each pixel can also be defined by $I = R + G + B$, where the range of each component's value is [0... 255]. A colour image created and stored in this way is said to have a depth of 24 bits because three image planes of eight bits each are required. The total number of colours that can be represented is hence $(2^8)^3$= 16,777,216 [15].

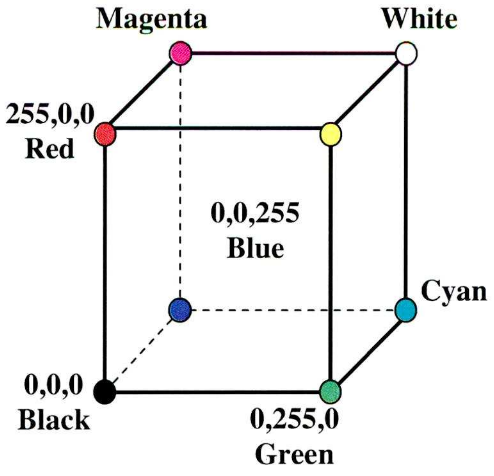


Figure 3-5 : RGB Colour space [15].

This model should be familiar to most people who have used cameras and computer monitors as it is used in many electronic systems involving sensing and displaying of colour in general. It is a hardware-oriented model which means that the resultant colour depends on the equipment and the set-up used to produce it. Each monitor, for example, will display a colour in a slightly different way depending on its age, calibration and materials used. Another drawback of this colour space is that it is not robust to changes in lighting. As the lighting changes, so does the corresponding location of a point in the colour space. RGB cannot describe colour in a constant way when illumination changes. One way to improve the RGB representation is to normalise each colour component value with the brightness value I to give us the Normalised RGB (*NRGB*) space as follows [73]:

$$r = R/I, \quad g = G/I, \quad b = B/I \qquad (3\text{-}2)$$

where $r + g + b = 1$.This transformation reduces the sensitivity of the colour information to the brightness value of a pixel and hence provides a way to achieve colour constancy. Only two of these three normalised variables are needed to specify any colour within the range allowed by the primaries since $r + g + b = 1$. ($b = 1$ is given by the absence of r and g).

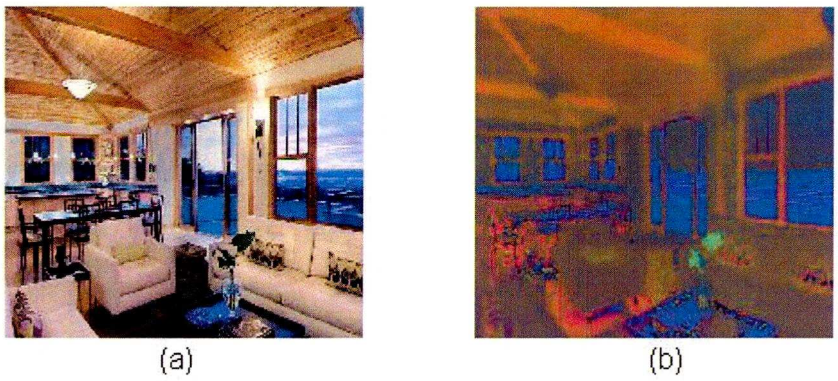


Figure 3-6 : (a) Original RGB image (b) Image shown after colour channels have been normalised.

3.3.2 The HSV colour space

The *Hue-Saturation-Value* (HSV) colour space, which is also device dependent, is a model closer to the way humans tend to perceive colour than RGB. For example, when asked the colour of a random car on the streets, one does not answer by giving percentages of each of the primary colours that compose the final colour because humans do not think of colour images as being composed of single images that combine together to form a final colour image. When people view a coloured object, they tend to described it as pale, deep, or light red for example. In a similar fashion, HSV tries to model colour the same way that humans interpret it. In this representation, *Hue* is a colour attribute that describes a pure colour (e.g. pure red, pure yellow or pure orange) and is represented as an angle which varies from 0 to 360. S*aturation* defines the relative purity or the amount of white light mixed with a hue and is measured from 0 to 1. *Value* refers to the brightness of the image, which is a subjective descriptor. Value is that quality that distinguishes a light colour from a dark one. The HSV model decouples the intensity component from the colour-carrying information (hue and saturation) in a colour image.

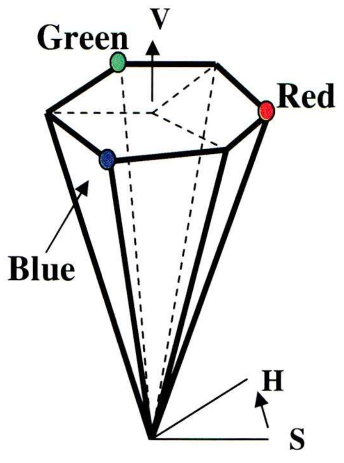


Figure 3-7 : HSV colour space using the hexcone model

It is often represented by the *hexcone* model proposed by A. R. Smith [74] which has the shape of a hexagonal cone. Converting from RGB to HSV is simply a matter of developing the equations to map RGB Cartesian coordinate values to the cone coordinates of the HSV model which are readily available from numerous textbooks such as the one by Gonzalez and Woods [15].

$$H = \begin{cases} \theta & \text{if} \quad B \leq G \\ 360 - \theta & \text{if} \quad B > G \end{cases} \tag{3-3}$$

with $\theta = \cos^{-1}\left\{ \frac{0.5[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}} \right\}$

$$S = 1 - \frac{3}{R+G+B}[\min(R,G,B)] \qquad (3\text{-}4)$$

The HSV model is based on polar coordinates rather than Cartesian with the hue of the primary colours separated by 120°. Decreasing S corresponds to increasing whiteness, and decreasing V corresponds to increasing blackness.

3.3.3 The YCbCr colour space

This is another hardware-oriented model used very frequently in video systems, and unlike the RGB space the luminance is separated from the chrominance data. Chrominance is the property that the average person thinks of as the 'colour' of light or an object, and it is the colour portion of a video signal that is closely related to hue and saturation, requiring luminance to make it visible [4]. The *Y* value represents the luminance (or brightness) component, while the *Cb* and *Cr* values (where C*b* = Blue minus 'black and white', and C*r* = Red minus 'black and white'.) represent the chrominance component of the image.

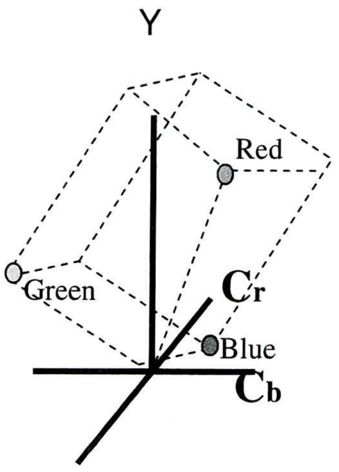


Figure 3-8 : YCbCr Colour Space [75]

This colour model was invented for colour television systems and it had to be compatible with black-and-white (B&W) TV systems. The luminance component already existed as the B&W signal and the colour channels (Cb and Cr) were added on top of the luminance channel to give the YCbCr space. YCbCr is used in common digital video compression algorithms such as MPEG-2 as it allows image compression techniques to take advantage of the fact that the eye is more discriminating of brightness levels than colour [76]. YCbCr is not an absolute colour space in the sense that it is derived from RGB space. It is rather a way of encoding RGB information, and the actual colour displayed depends on the RGB colorants used to display the signal. There are several ways to convert from RGB to YCbCr color space with the most common being the CCIR (International Radio Consultative Committee) Recommendation 601-1, as shown in next equation [77].

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \frac{1}{256}\begin{bmatrix} 65.738 & 129.057 & 25.064 \\ -37.945 & -74.494 & 112.439 \\ 112.439 & -94.154 & -18.285 \end{bmatrix}\begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (3\text{-}5)$$

3.3.4 The CIE L*a*b* colour space

CIE *L*a*b** is a device independent model developed by the *Commission Internationale d'Eclairage* (CIE) and is derived from the CIE *XYZ* colour space (Figure 3-4). The three parameters in the model represent the lightness of the colour (L^*, $L^* = 0$ yields black and $L^* = 100$ indicates white), its position between red and green (a^*, negative values indicate green while positive values indicate red) and its position between yellow and blue (b^*, negative values indicate blue and positive values indicate yellow). Image data can be imagined as being plotted in 3 dimensions along 3 independent (orthogonal) axes, one for brightness and two for colour. The colour axes are based on the fact that a colour cannot be both red and green, or both blue and yellow.

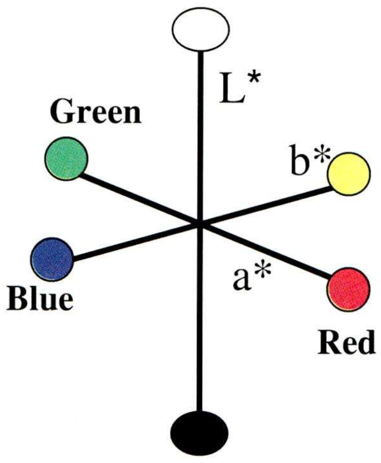


Figure 3-9 : CIE L*a*b* Colour Space

Although not always very intuitive to use, *L*a*b** is designed to approximate human vision with its *L* component closely matching human perception of lightness. The *a** and *b** components contain the required 'colour' information. *L*a*b** is also defined as being a perceptually uniform colour space. A uniform colour space is one in which the distance in coordinate space is a good guide to the significance of the difference between two colours – in such a space, if the distance in coordinate space were below a threshold, a human observer would not be able to tell the colours apart [72]. More information on this colour space and its properties are available in standard reference textbooks such as Wyszecki and Stiles [70]. The quantities *L**, *a**, and *b** are obtained from the tristimulus values according to the following transformations:

$$
\begin{aligned}
L^* &= 116\left(\frac{Y}{Y_n}\right)^{1/3} - 16 \\
a^* &= 500\left[\left(\frac{X}{X_n}\right)^{1/3} - \left(\frac{Y}{Y_n}\right)^{1/3}\right] \\
b^* &= 200\left[\left(\frac{Y}{Y_n}\right)^{1/3} - \left(\frac{Z}{Z_n}\right)^{1/3}\right]
\end{aligned}
\qquad (3\text{-}6)
$$

where X_n, Y_n, and Z_n are the X, Y, and Z values of a reference white patch [72].

Figure 3-10 shows the information contained in each dimension of these four colour spaces. Although each dimension (or channel) has values which are of different scales, it is possible to compare and contrast the layers when they are all mapped to a pseudo-colour for visualisation purposes, e.g. greyscale mapping. The colour spaces mentioned so far are used in various applications based on the requirements. In this thesis, a robust way of modelling true colour in the presence of lighting variations is required. One way to achieve this is to use a colour space that satisfies this condition. By robust here, it is meant that certain features of an image are maintained even when the light changes. An example of a not very robust colour space is shown in Figure 3-12 where the same image taken under different lighting conditions gave different point distributions in the RGB colour space. Other colour spaces such as the HSV tend to preserve the value of coloured elements in its Hue channel if the lighting conditions are not dropped dramatically to near darkness. However given the circular representation of data, this colour space may not be suitable for the triangle wrapping and un-mixing processes that are introduced in the next section 3.4. The Normalised RGB (*NRGB*) colour space although not fully immune to changing lighting conditions tends to preserve the most important information as shown in Figure 3-13. *NRGB* is used as the main colour space for most of the tests in this thesis. This is because of the main detection technique used involves the need to wrap a triangle around a set of points and it was found that the distribution of points of the NRGB colour space when faced with varying levels of lighting conditions change in such a way that the triangle wrapping process adapts itself well to maintain a closely wrapped triangle. This whole concept about triangle wrapping to detect objects is explained in sections 3.4 and 3.5 while the perceived advantage of the NRGB colour space over the other colour spaces in dealing with triangle wrapping for a changing point distribution is shown in the next chapter 4.

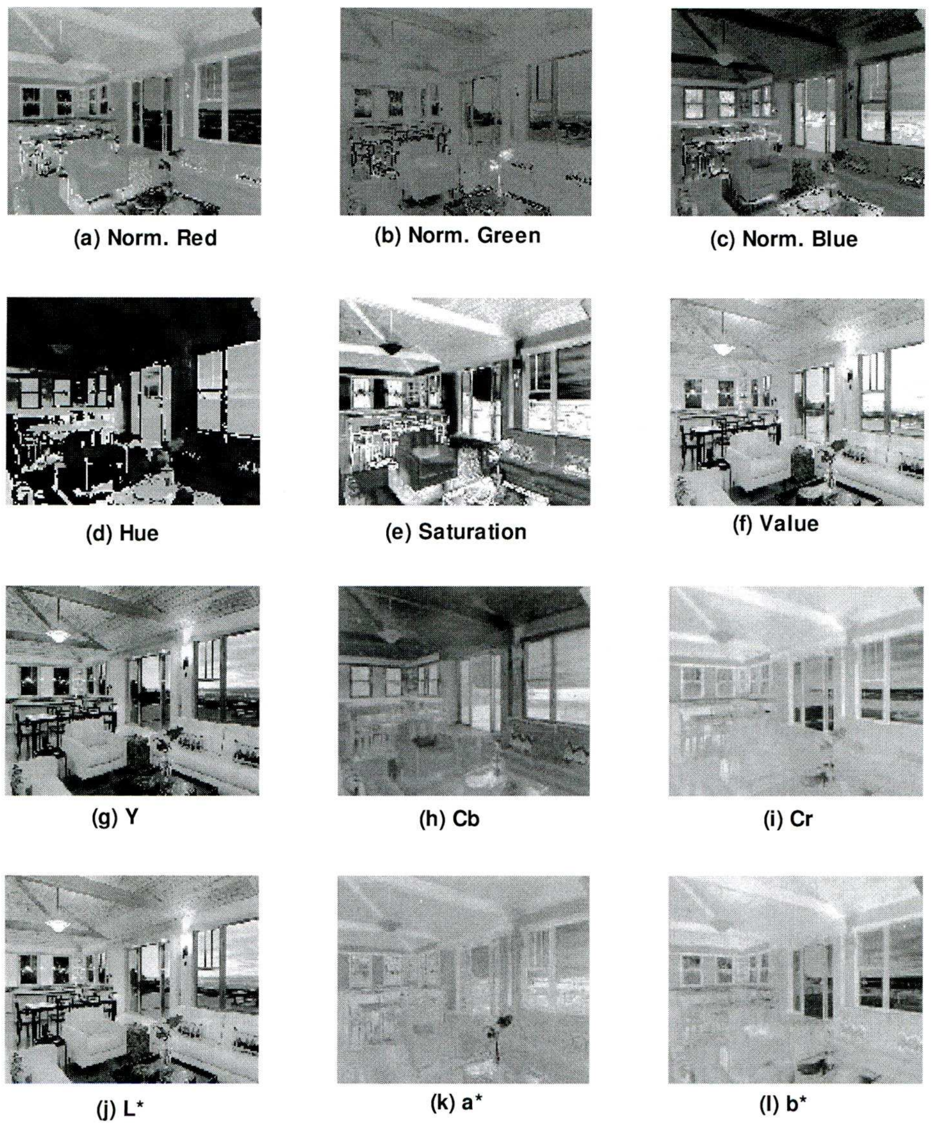


Figure 3-10 : Each row shows the information contained in the 3 dimensions for each colour space. The information has been converted to a greyscale map for visualisation.

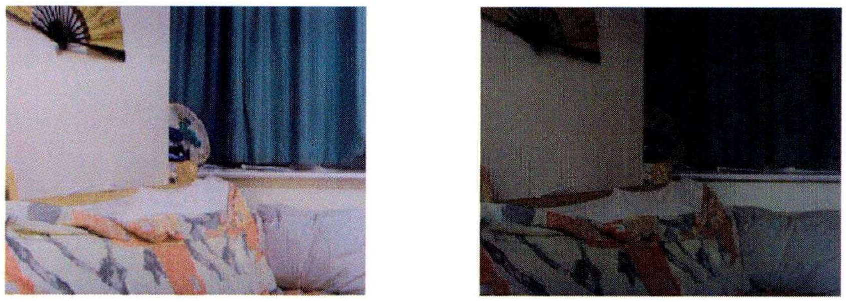


Figure 3-11 : Same image taken under bright and dark light conditions.

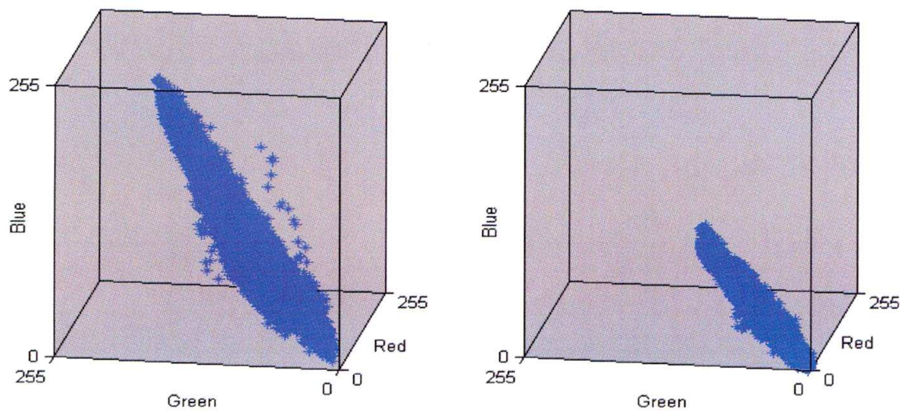


Figure 3-12 : RGB point distribution for bright and dark image.

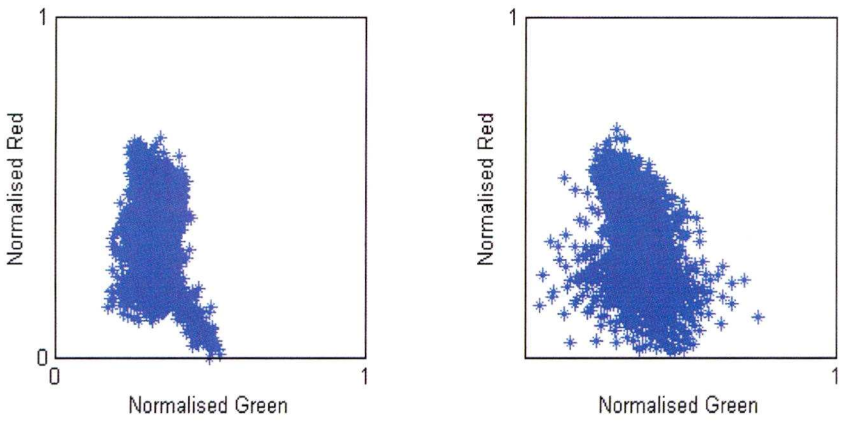


Figure 3-13 : Normalised RGB point distribution for bright and dark image

3.4 Detection algorithms used in spectral imaging applications

In satellite reconnaissance, sub-pixel target detection is very common and many algorithms have been developed to deal with situations where each pixel contains several distinct types of object or background, each with a different colour or spectral signature. In this context, 'colour' is a general term that is related to the intensities in different visible and/or thermal (i.e. infrared) wavebands – the number of wavebands can be significantly larger than the usual three colours that are familiar in the visible band simple sensors. A multi-spectral image consists of a few colour layers with each layer representing an image acquired at a particular wavelength band whereas a hyper-spectral image (HSI) is acquired from many (100 or even more) contiguous and very narrow (about 0.010 μm wide) spectral bands that typically span the visible, near-infrared, and short wave infrared portions of the spectrum (0.4 μm - 2.5 μm) [78]. Spectral imaging techniques exploit the fact that all materials reflect, absorb, and emit electromagnetic energy, at specific wavelengths, in distinctive patterns related to their molecular composition. This enables the construction of an almost continuous radiance spectrum for every pixel in the scene. There is a restriction in the band wavelengths that are used due to the fact that the reflected radiance is attenuated after passing through the atmosphere and this attenuation is wavelength dependent.

The reflectance spectrum of a material is a plot of the fraction of radiation reflected as a function of the incident wavelength and serves as a unique signature, called the spectral signature, for the material [78]. Figure 3-14 shows the spectral signature of different types of surface materials. HSI data exploitation makes possible the remote identification of ground materials-of-interest based on their spectral signatures. If one wants to identify a component of a particular pixel, one can just analyse how the spectral reflectance behaves for each wavelength and then match it to a library of wavelengths. A form of matched filtering can also be performed. In signal processing, matched filtering is used to maximise a signal relative to noise and clutter, and this same concept is used when detecting objects of interest based on their spectral signatures. The response of the desired and known component is maximised and the response of the composite unknown background is suppressed, thus 'matching' the known signature. As the number of components to be identified is limited (e.g. vegetation, soil, water, road etc), the libraries for these elements have already been set up and this provides a rapid means of detecting these specific materials.

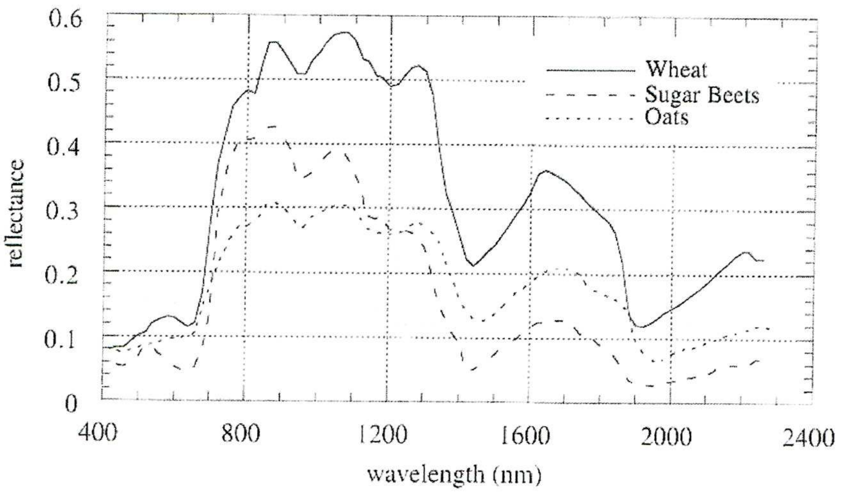


Figure 3-14 : Spectral reflectance curves of different types of surface materials [78].

The main problem that arises when the spatial resolution is not high enough is that natural surfaces are rarely composed of a single uniform material and hence, many different classes of material may be present in each pixel. This problem is often described as the mixed-pixel classification and many researchers have investigated this field called spectral mixing in general [79]. There are two main approaches to classify mixing within pixels: Linear and non-linear approaches. Several researchers have investigated mixing scales and linearity. Singer and McCord [80] found that if the scale of the mixing is large (macroscopic), mixing occurs in a linear fashion while for microscopic or intimate mixtures, the mixing is generally nonlinear. In the linear model, the mixing of the light happens in the sensor, as the rays from the different patches on the ground are reflected in the field of view of the sensor element. In the intimate mixture model, the incident light suffers many multiple reflections from the different components before it ends up in the sensor element as shown in Figure 3-15. The spatial scale of the mixing and the physical distribution of the materials govern the degree of non-linearity.

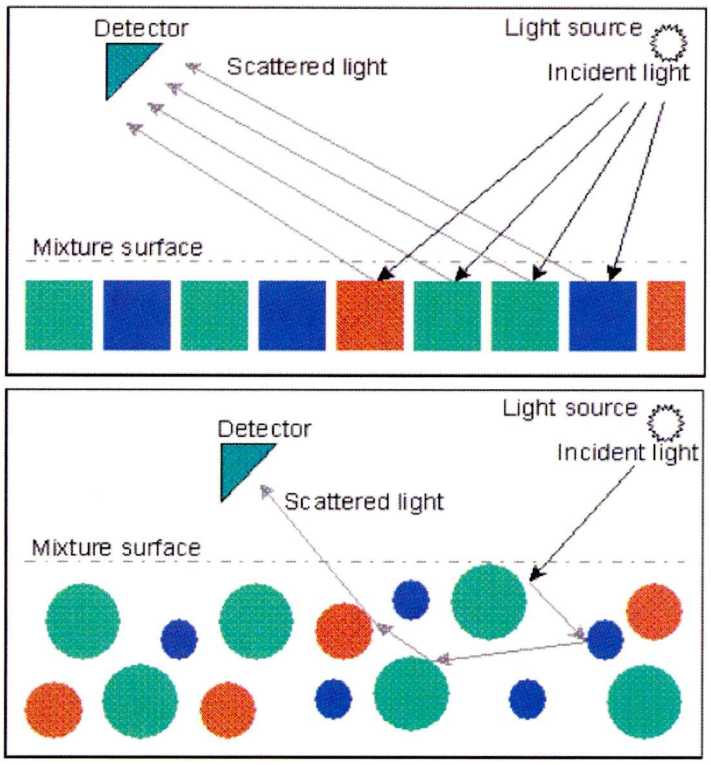


Figure 3-15 : Top: Linear mixing model with mixing of light occurs at the sensor. Bottom: Non-linear with mixing happening on the ground. [79]

The linear approach to mixed pixel classification is best described by the Linear Mixing Model (LMM) [81]. LMM assumes that the spectrum of a mixed pixel is a linear combination of the spectra of the constituent pure classes and that the spectral proportions of the pure classes reflect the area covered by the proportions on the ground. In other words, each pixel in a given image contains a proportion of one or more definite colours (or spectra) and that each mixed pixel may be decomposed into a linear combination of the individual colours or pure classes, also known as *endmembers* [78, 81]. LMM is described mathematically as a linear vector-matrix equation,

$$P_{xy} = E\alpha_{xy} + \varepsilon \tag{3-7}$$

where α_{xy} is the $L\times1$ vector of L endmember fractions for the pixel xy, and E is the $K\times L$ endmember signature matrix, with each column containing one of the spectral vectors. P_{xy} is the K-dimensional spectral vector at pixel xy and ε represents noise. The α's represent the amount of each constituent in a given pixel, and are often referred to as the abundance coefficients with the following 'additivity' and positivity constraints applied to them.

$$\sum_{l=1}^{L} \alpha_l = 1 \quad \text{and} \quad \alpha_l \geq 0 \tag{3-8}$$

Solving the LMM equation is called un-mixing. If the spectra of the pure classes are known, the apparent fractional abundance of each endmember material in each pixel can be easily deduced. These known endmembers can be drawn from the data (averages of regions picked using previous knowledge), drawn from a library of pure materials by interactively browsing through the imaging spectrometer data to determine what pure materials exist in the image, or determined using expert systems to identify materials. The mixing endmember matrix is made up of spectra from the image or a reference library. The problem can be thought of in terms of an over-determined linear least squares problem [78]. The mixing matrix is inverted and multiplied by the observed spectra to get least-squares estimates of the unknown endmember abundance fractions. Constraints can be placed on the solutions to give positive fractions that sum to unity. Shade and shadow are included either implicitly (fractions sum to 1 or less) or explicitly as an endmember (fractions sum to 1) [79].

The un-mixing method described so far assumes that the spectral libraries are available. However it can happen that they are not available. There is another technique called the 'full un-mixing' that uses the imaging spectrometer data themselves to 'derive' the mixing endmembers [82]. This technique can be summarised as follows:

- A linear 'sub-space' that spans the entire signal in the data is derived.
- The data are projected onto this subspace. This lowers the dimensionality of the un-mixing.
- The data are 'shrink-wrapped' by a simplex of *n*-dimensions. Examples of simplexes in different dimensions are a line, a triangle and a tetrahedron for n= 1, n = 2, and n = 3 respectively.
- The simplex is used to derive abundance estimates of the pure endmembers. The estimates that are obtained are positive fractions that sum to unity.

Mixed pixels are visualised as points in n-dimensional scatter-plot space (spectral space), where n is the number of bands. In two-dimensional feature space, if only two endmembers mix, then the mixed pixels will fall in a line and the pure endmembers will fall at the two ends of the mixing line. If three endmembers mix, then the mixed pixels will fall inside a triangle. If these endmembers are 'pure' and if they are an exhaustive basis for all spectral vectors in the image, the spectral vector for any pixel must lie within the convex hull defined by the envelope around the endmembers. In other words, mixtures of endmembers 'fill in' between the endmembers.

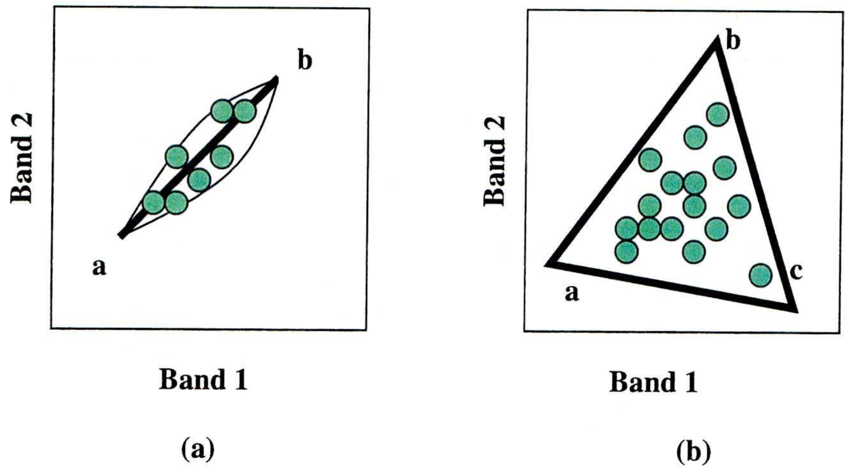


Figure 3-16 : (a) Two endmembers (b) Three endmembers

In Figure 3-16 (b), the corners of the triangle represented by a, b and c could represent soil, rocks and vegetation for example. Un-mixing can then be applied to estimate the proportion of each class member for each pixel vector. Assuming that there is no noise at first, each element of equation 3-6 can be written out as follows:

$$\begin{bmatrix} P_{\text{Band1}} \\ P_{\text{Band2}} \end{bmatrix} = \begin{bmatrix} E_{soil1} & E_{rocks1} & E_{vegetation1} \\ E_{soil2} & E_{rocks2} & E_{vegetation2} \end{bmatrix} \begin{bmatrix} \alpha_{soil} \\ \alpha_{rocks} \\ \alpha_{vegetation} \end{bmatrix} \qquad (3\text{-}9)$$

Equation 3-8 is underdetermined since there are three unknowns, i.e. the fraction components, but only two equations for the bands 1 and 2 respectively. However the requirement that all linear combinations of the three fraction components add up to unity can be combined with equation 3-9 to form an augmented mixing equation such that the latter can be solved for the fractions as shown in the next equation 3-10 [78].

$$\begin{bmatrix} \alpha_{soil} \\ \alpha_{rocks} \\ \alpha_{vegetation} \end{bmatrix} = \begin{bmatrix} E_{soil1} & E_{rocks1} & E_{vegetation1} \\ E_{soil2} & E_{rocks2} & E_{vegetation2} \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} p_{\text{Band1}} \\ p_{\text{Band2}} \\ 1 \end{bmatrix} \qquad (3\text{-}10)$$

The task of identifying the endmembers in a scene is a difficult task and two main approaches exist to deal with it, namely interactive and automated endmember extraction techniques. The pixel purity index (PPI) method [83] is the most representative interactive approach. Its operation first involves randomly generating lines in the chosen n-dimensional space. All of the data points in the space are then projected onto the lines after a maximum noise fraction (MNF) transform has been applied to them to reduce their dimensionality [83]. Those pixel values that fall at the extremes of the lines are counted. After many repeated projections to different lines, those pixels with a count above a certain threshold are declared 'pure' or an endmember. However this method usually gives many redundant spectra in the pure pixel list and the actual endmember spectra can only be selected after a combination of intelligent review of the spectra themselves and through n-dimensional visualisation. An alternative is the N-FINDR method [84] which is an automated approach that finds the set of pixels that define the simplex with the maximum volume, potentially inscribed within the dataset. Randomly selected pixels qualify as endmembers, and a trial volume is calculated. In order to improve the initial volume estimate, a trial volume is calculated for every pixel in each endmember position by replacing that endmember and recalculating the volume. If the replacement results in a volume increase, the pixel replaces the endmember. This procedure, which does not require any input parameters, is repeated until there are no replacements of endmembers left. Once the endmembers are found, their spectra can be used to un-mix the original image using equation 3-10. This produces a set of images, each of which shows the fractional abundance of an endmember in each pixel. Other automatic endmember extraction techniques are introduced in section 3.5. It should be noted that both PPI and N-FINDR rely on spectral properties of the data alone, neglecting the information related to the spatial arrangement of pixels in the scene.

The other approach to model mixing within pixels is the non-linear one. Nonlinear mixed pixel analysis involves a detailed knowledge of multiple scattering effects that may arise due to the intimate association of components residing inside each pixel and it is described extensively by Hapke [85]. However it has been observed that linear un-mixing techniques, while at best an approximation, appear to work well in many circumstances involving non-linear mixtures with the effects of multiple scattering in the majority of applications assumed to be negligible if a linear model is used [86]. Moreover indoor environments have relatively few light sources and they contain few shiny surfaces. These objects do not scatter and reflect light in a way serious enough way to model mixing in the non-linear manner. The un-mixing of the LMM is the process that is adapted to detect objects in low resolution videos for this project and this adaptation is expounded both in the next section and in chapter 4.

3.5 Un-mixing process to detect objects and estimate their sizes

In low resolution images, each pixel can often represent more than just one colour value. The combination of finite pixel sizes and small number of pixels available results in mixing within individual pixels and the LMM (together with its un-mixing) is an approach that can be used to separate the colours present in a pixel. In a two-dimensional feature space (for example Normalised Red v/s Normalised Green), a distribution similar to Figure 3-17(a) might be expected for an image with three main colours. If the end-members are 'pure', one can expect the spectral vector for any pixel in the image to lie within the triangular envelope around the endmembers. Pixels containing only one endmember will be found near the vertices of the simplex, those with two endmembers will be located near the facet of the simplex opposite the vertex associated with the missing endmember as shown in Figure 3-17(b), and the pixels with all three endmembers will tend to be found near the centre of the triangle. The position of a point within the triangle gives information about the proportion of an endmember contained in that pixel. An example of a real low resolution image is shown in Figure 3-18. It can be seen how the data points representing each colour are based around three corners and then other points fill in the rest of the space in between the corners.

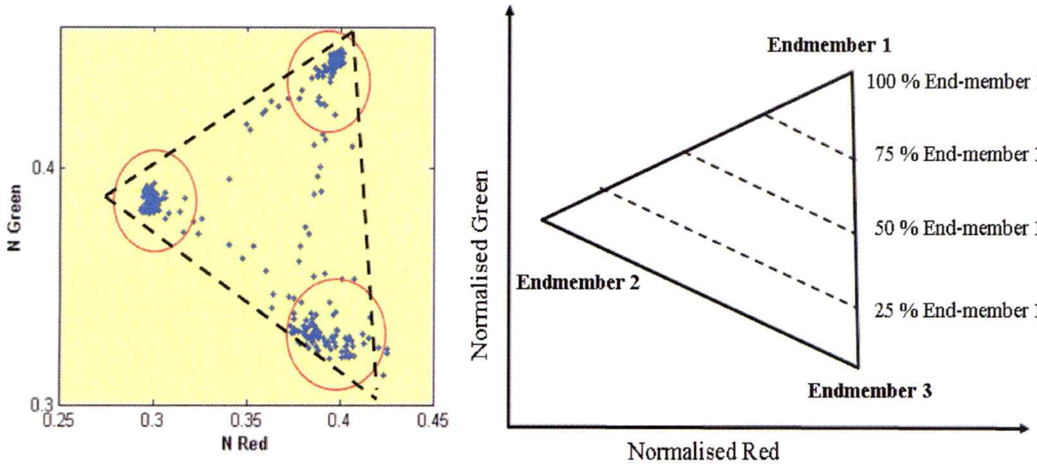


Figure 3-17(a) : Endmembers are encircled. Triangular envelope wraps around the data values and endmembers, (b) Parallel contours of equal abundance of Endmember 1[75].

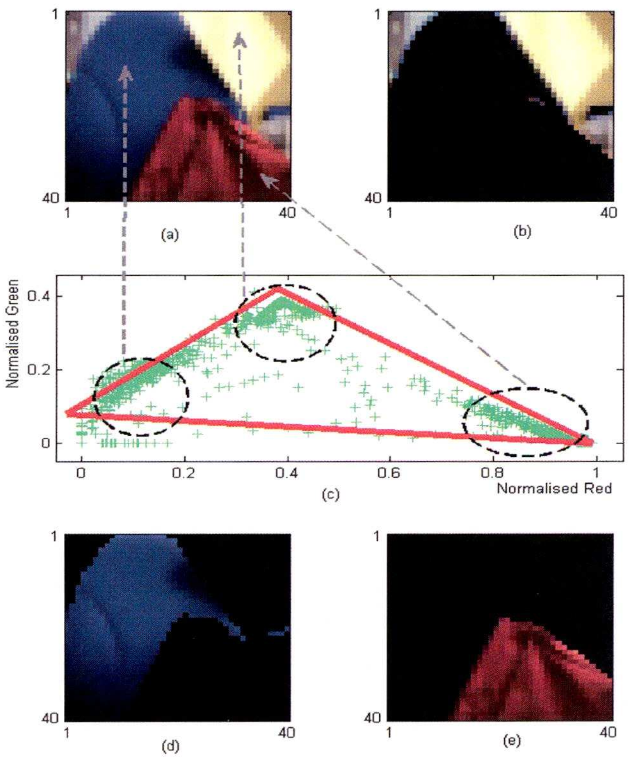


Figure 3-18 : (a) Real low resolution image. (c) Triangular envelope wraps around data values and endmember zones are circled. Fig. 1(b), (d) and (e) show the portions of the images that represent these three endmembers.

Knowing the point distribution on its own is not however very useful. To be able to perform the un-mixing process, a triangle wrapped around the data points as closely as possible is required and this process of obtaining the triangle automatically is explained in the next chapter 4.2. When a triangle is available around the data points, the un-mixing process can be performed and this is done to estimate fractions of each endmember component from a given pixel as explained in equation 3-10. The 2-D explanation example shown in Figure 3-17 above is chosen here because it is easy to visualise. However it must be noted that this approach scales up to higher dimensions. For example, if a fourth endmember is added to the first three, three dimensions will then be required and the mixing triangle will become a tetrahedron to determine if a point is outside or inside the mixing space. This means that for unambiguous colour un-mixing, the dimensionality of the mixed data must usually be one less than the number of independent endmembers. In domestic environments with simple image sensors, k usually represents two bands e.g. normalised red and green intensity values from RGB data capturing and it can be assumed that there are three main colours in such rooms, e.g. background wall-paint, a main furniture and a moving person with clothing containing one major colour that stands out. The E values of equation 3-10 are assigned the coordinate values of the triangle's vertices while P is assigned the normalised intensity values of the pixel being un-mixed. This process is also sometimes called pixel decomposition. One might argue the point here that this un-mixing procedure is very inefficient if it has to be calculated for each pixel present in an image frame. This is somewhat true but at the same time one has to be aware that not all pixels need to be un-mixed as not all pixels are mixed, i.e. not all pixels in a low resolution image represent more than one or part of one object at a time. Mixed pixels tend to be found at object boundaries where they represent areas that partially belong to one object and partially to another one or a background. The reverse process of matching the data points near a particular corner to a pixel location in the image can be done to estimate the position of the 'centre of mass' or centroid of an object, as will be explained in the next section 3.5.1. This method is more appropriate when there are three or four main colours in an image sequence and the object to be tracked occupies a significant number of pixels (at least 3 pixels in a 20×20 image).

3.5.1 Finding the 'centre of mass' of an object

The centroid of a 2-D object is a unique point that balances it on a pinpoint if it were cut out of a rigid, uniform sheet of cardboard. There exist many formulas to find the centroids of regular shapes such as squares, rectangles, etc. In computer images, objects are represented by pixels and are often of irregular shapes. It is hence difficult to apply standard equations from geometry to find the centre of mass of these objects. For irregularly shaped objects, the actual feature shape and location of all the pixels present can be taken into account to enable the centroid to be obtained. One quick way to achieve this is by finding the largest and smallest coordinates in the horizontal and vertical directions first. These limiting coordinates then define a bounding box around the object in the spatial domain and the midpoint of this box can be taken as the centre of mass [68]. This method is not always ideal because it is too easily biased by just a few pixels, e.g. a whisker sticking out of an object will pull the centroid in the direction of the whisker [68]. However the objects of interest in this thesis are expected to be 'solid' ones, e.g. piece of clothing, and therefore this technique is acceptable. A way to derive the centre of mass of an object using bounded-box method augmented with pixel un-mixing is devised. As shown before in Figure 3-17(b), the location of a point relative to a vertex within the un-mixing triangle determines the proportion of that endmember in the pixel being analysed. These proportions can be added to the main central bounded-box (or blob) to give a better estimate of the object's shape and area, and hence its centroid. To understand this concept, consider Figure 3-19. An intial centroid estimate can be obtained in a direct and simple way for the full pixels numbered 6, 7, 10, and 11. The intensity values of these pixels are normally found very close to the triangle vertex representing the object's colour in the data space projection. These points are usually found at a distance of less than one quarter of the length of each of the two sides joining at this vertex along each side. When these intensity value points are projected back to the spatial domain, a bounding box is obtained and its midpoint is the initial centroid estimate. This estimate can then be improved by using pixel decomposition. Pixel decomposition or un-mixing allows us to know the amount of the desired object present in the pixels found at or near the boundaries of the object, and by adding these proportions to the initial centroid estimate, the latter's accuracy can be improved.

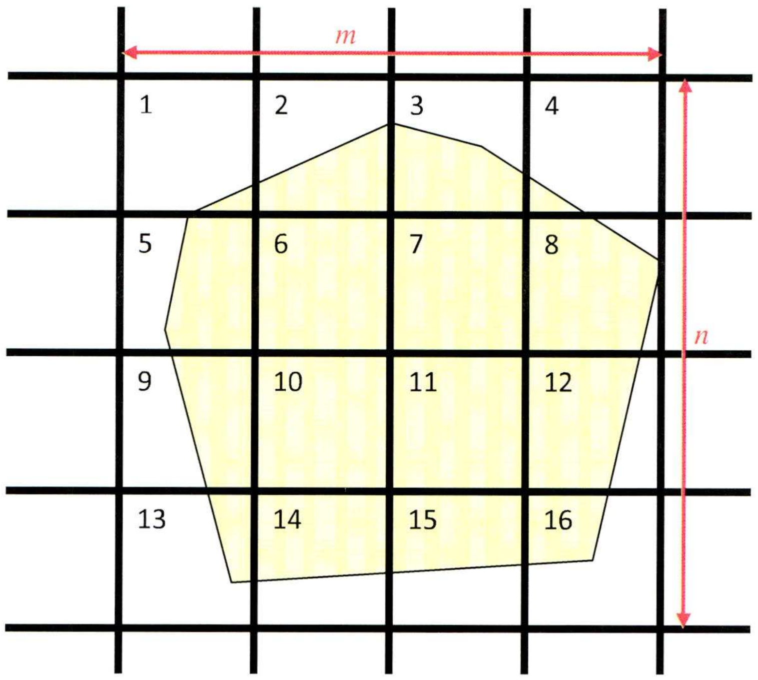


Figure 3-19 : Estimating the 'centre of mass' of an object. Pixels 6, 7, 10 and 11 are used to obtain initial centroid estimate. The remaining pixels are un-mixed to improve the estimate.

Before explaining how this improvement to the centroid's estimate is achieved, it is good to see how the proportion of an object in a particular pixel can be calculated in a real world example. The top row of Figure 3-20 is an example of an image sequence containing a yellow object moving from right to left on a background with two main colours. A pixel (highlighted by a red square) is under observation because it contains different colour proportions over time as the object goes through it. The bottom row of Figure 3-20 shows the intensity values point distribution of each frame when these values are projected into the Normalised RGB colour space. A triangle is wrapped around each data set and the location of the chosen pixel's value is shown with a red cross. As the object moves into the pixel and covers it for a few frames, the cross moves towards the vertex of the triangle representing that object's colour. This means that the proportion of yellow in the pixel is increasing while the contribution of the other endmember colour decreases. Figure 3-21 shows a magnified version of the image sequence at frame 50 and the corresponding pixel value location in the data space projection. It can be seen from Figure 3-21 (b) that the pixel under observation is neither pure yellow nor pure green but a mixture of both. This information is reflected in the data space projection where the cross lies in between the vertices representing these two endmembers. The location of this cross is used to estimate the proportion of a particular endmember in that pixel by using equation 3-10.

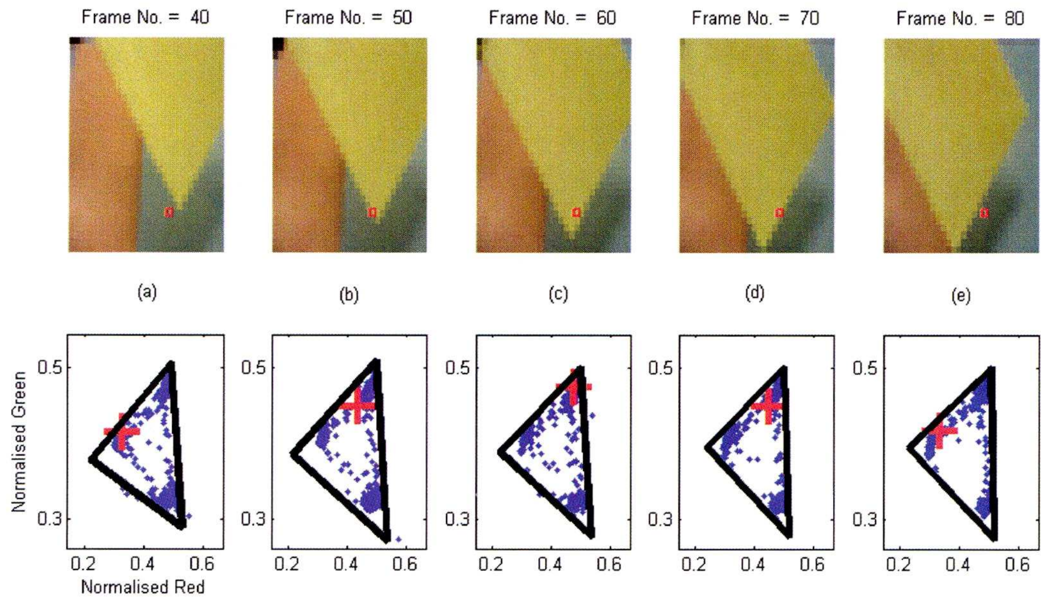


Figure 3-20 : Top row is an image sequence containing a yellow object moving from left to right on a background with two main colours. A pixel (highlighted by a red square) is under observation. Bottom row shows the corresponding data point distributions and a triangle wrapped around them for the un-mixing process.

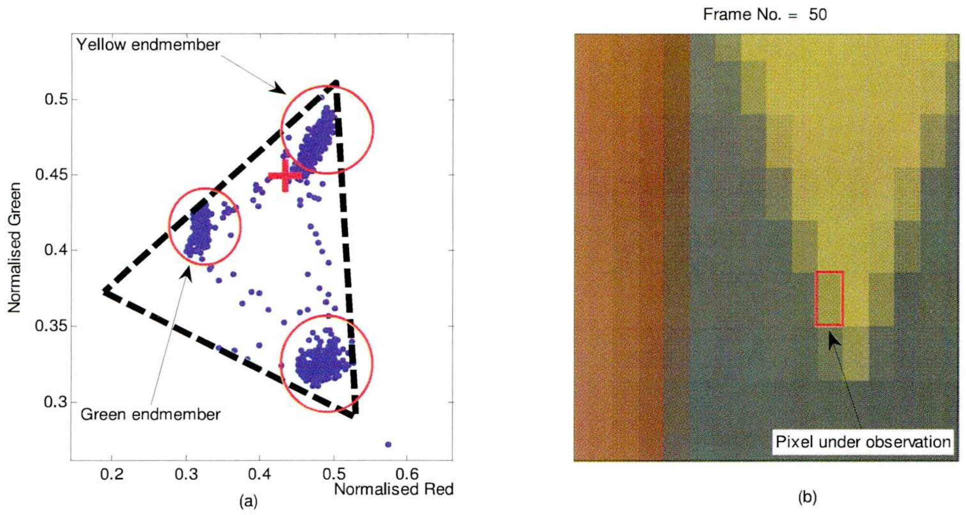


Figure 3-21 : (a) The location of a pixel value in the data space projection, (b) Blow-up of the pixel in the image sequence at frame 50

Figure 3-22 shows the calculated proportions of each endmember in the observed pixel as the object moves through it. The zones between the vertical dotted lines represent the period when mixing within the pixel is occurring. It has to be noted that the proportions add up to unity at any instant in time. However the proportions obtained for each endmember are not always accurate because the triangle used for the un-mixing process is not tightly around the data points. This is addressed in the next chapter where an automatic triangle wrapping algorithm is developed to work on a sequence of data points.

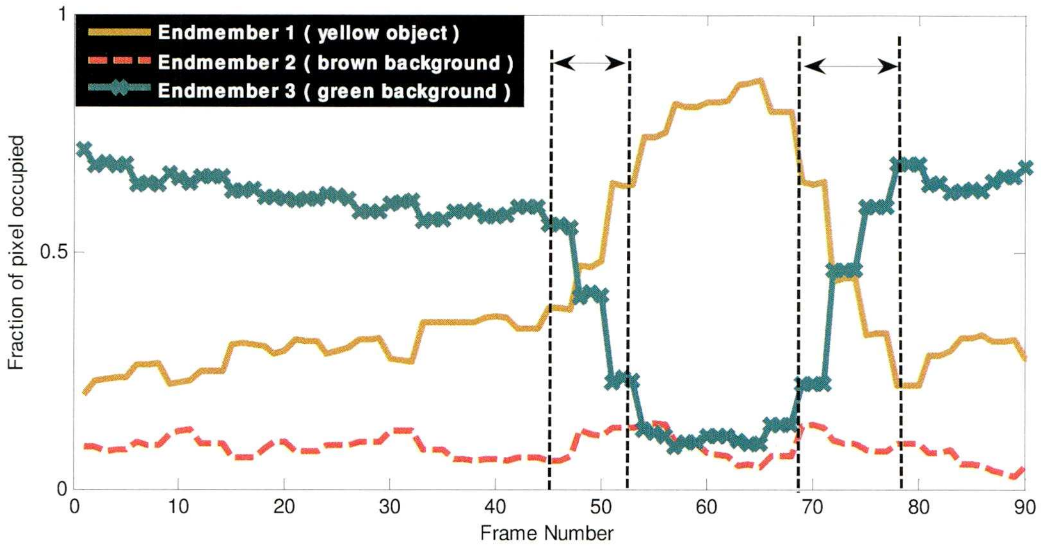


Figure 3-22 : Proportions of each endmember in the observed pixel over time

Being able to calculate the proportion of an endmember in a pixel is just part of the quest to improve the centroid location. The idea is to decompose all the other mixed pixels in Figure 3-19 and then add up their respective proportions of the desired endmember. To achieve this, the mixed-pixels around the central blob that contain more than 50% of the desired endmember are un-mixed. The intensity values of these pixels are found in a band within the un-mixing triangle as shown in Figure 3-23. Those pixels with more than 75% of endmember present are not decomposed because they are already taken into account in the derivation of the initial bounding box or blob and those with less than 50% are simply ignored. Any pixel within this desired band is given the weight equivalent to a pixel containing half the intensity value of the desired endmember. Using this information, an improved estimate of the centroid of the object is obtained by using a weighted centre of mass approach based on the intensities of the pixels as explained in equation 3-11 below. Given each pixel (i,j) with intensity level l_{ij} in the m × n window shown in Figure 3-19, the coordinates (X, Y) of the centroid relative to the local axes of the window are [87]:

$$X = \frac{\sum_{j=1}^{n} \sum_{i=1}^{m} j.l_{ij}}{\sum_{j=1}^{n} \sum_{i=1}^{m} l_{ij}} \quad \text{and} \quad Y = \frac{\sum_{j=1}^{n} \sum_{i=1}^{m} i.l_{ij}}{\sum_{j=1}^{n} \sum_{i=1}^{m} l_{ij}} \qquad (3\text{-}11)$$

The coordinates of the centroid in the image co-ordinate system are $(x_0 + X,\ y_0 + Y)$ where both x_0 and y_0 have a value of 1 in digital image processing applications. While this is still not a fully precise location of the centroid, one has to be aware that the requirements are not as strict as microscopic tracking in biomedical applications for example and a small error is allowed here. Moreover because of the nature of digital image representation in computers, rounding occurs which result in near impossibility to obtain locations of sub-pixel precision. Yet, this method of detecting an object and estimating its location does give good results as will be shown in the next chapters.

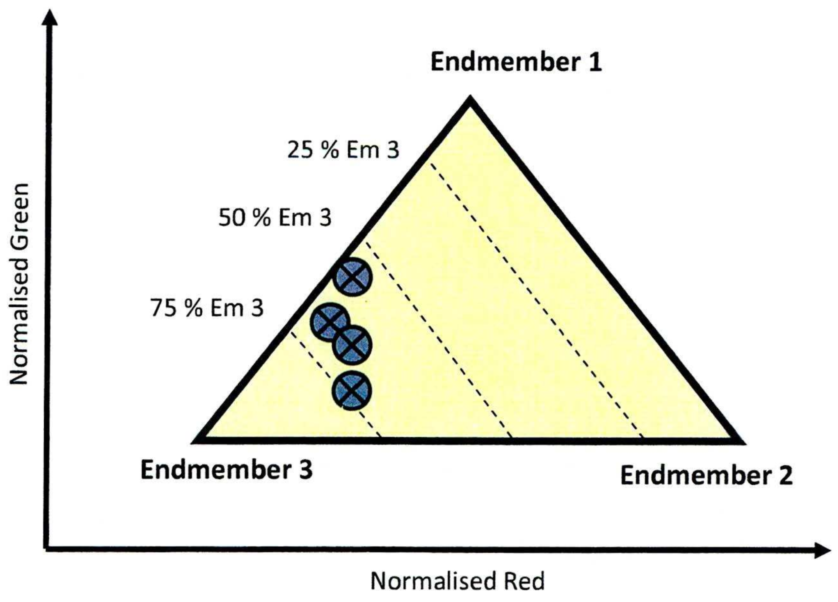


Figure 3-23 : Location of intensity data points of mixed pixels

3.6 Summary

This chapter has given further information about the various parts of this project. The tracking of small objects is an established field and is very useful in certain sectors such as biomedical and military tracking. A different technique, based on satellite reconnaissance applications, to the ones currently used in these fields is proposed to be used in this experimental project. This technique involves using the un-mixing process to estimate the location and the size of an object. However before reaching that stage, the appropriate colour space has to be chosen. A colour space is a model for representing colour in digital images in terms of intensity values. It defines a one-, two-, three-, or four-dimensional space whose dimensions, or components, contain information about the colours. The colour space to be chosen has to be able to minimise the effect of changing ambient lighting on an image by keeping some information about the colours constant whatever the light conditions are. Another requirement for the colour space is that it must be compatible with the un-mixing process that is proposed to be used as a detection mechanism of small objects. The un-mixing process as used in satellite imagery applications involves a step whereby a triangle has to be wrapped around many data points as closely as possible. Algorithms that are able to automatically generate this triangle are explained and developed in the next chapter.

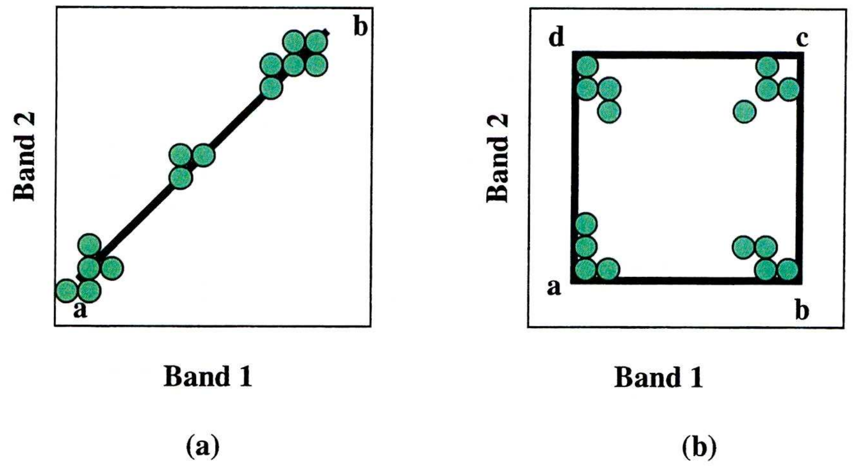


Figure 3-24 : (a) Three colours cannot be unmixed because no triangle is formed (b) four colours can be unmixed in 2-D space.

As mentioned in section 3.2, if three endmembers mix in two-dimensional feature space, then the mixed pixels will fall inside a triangle and the spectral vector for any pixel must lie within the convex hull defined by the envelope around the endmembers. Given only the spectral distribution, pixel un-mixing can then be practised on it to infer the constituency of each pixel depending on the location of its spectral vector within the triangle. There are however some special situations where three colours cannot be separated using this technique, and some where four colours can be separated in 2-D as shown in Figure 3-24. The presence of three main colours with the distribution of the data points enabling a triangle to be formed also does not always mean that the un-mixing process will work well. For example, when the object being investigated is very small, e.g. 1 or 2 pixels, it can happen that the triangle wraps itself around the wrong points. To understand this, consider the un-mixing triangle shown in Figure 3-25.

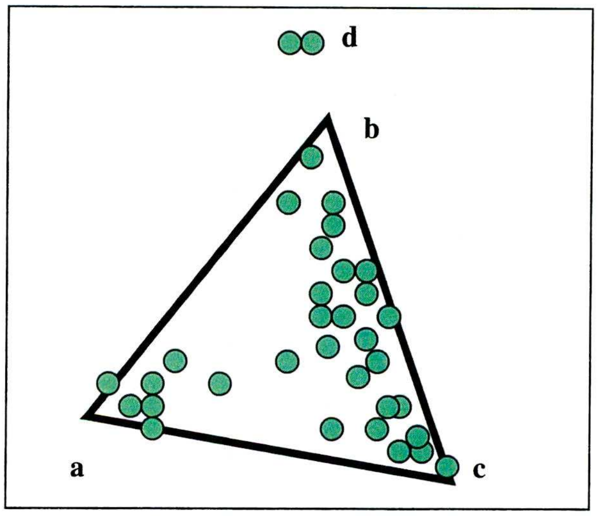


Figure 3-25 : Points a, c, and d represent the three true end-members for an image, however due to a lack of any pure pixels of type d, the selected end-members are a, c, and b (a mixed pixel).

Points a, c, and d represent the three true end-members for an image, however due to a lack of many pure pixels of type d, the selected end-members are a, c, and b (a mixed pixel). This means that the estimates of the proportions of d in a particular pixel will not be exact if d is the object that contains the colour that one is interested in. One way to tackle very small objects (1 pixel or less than a pixel in size) is by making use of the intensity profiles at pixel boundaries when these objects go through the pixels. By using the time derivatives of the intensity at each pixel, and the spatial derivatives of intensity obtained by comparing the intensity of neighbouring pixels, the displacement and even velocity can be estimated. Because of the very nature of these objects that are small, the intensity modulation that they bring about in a pixel is very small and is often heavily affected by noise in an image. A method to filter these values has to be used first to recover the information that is needed to be able to estimate the location of these objects, and this technique for detection sub-pixel sized objects is introduced in the next chapter.

CHAPTER 4

4. Detecting objects

4.1 Introduction

One of the main parts of this thesis is the development of a suite of algorithms to detect small objects with sizes ranging from less than a pixel (sub-pixel) to a few pixels wide. This chapter is mainly concerned with how the un-mixing process is adapted to detect small objects in image sequences of indoor environments. An automated triangle wrapping mechanism is developed and implemented as part of this preliminary and very important step of un-mixing. The search for the optimum triangle can be done by an exhaustive search over an allowable range of parameters. However, the computational cost increases with both the dimension of the parameter space and the dimension of the dataset, and thus an alternative search method using an optimisation technique has to be found. Several optimisation techniques are tested and a complete description of the results obtained in the test experiments is given together with the choice of the optimiser to be used. For very small objects, i.e. those less than a pixel in size, another technique to these objects using intensities at boundaries of adjacent pixels as the objects move through the pixels is described in section 4.3. A third technique involving the detection of human skin in image sequences is also introduced in section 4.4.

4.2 Triangle Wrapping Algorithms and detection of small objects

A very important task in the detection process of small objects as described in section 3.5 is to find an automated way to perform the triangular wrappings. There exist many ways to obtain a triangle wrapped closely around a set of data points. Three commonly used algorithms are: the Simplex Shrink-Wrap [1], the Minimum Volume Transform [88] and the N-FINDR algorithm [89]. The Simplex Shrink-Wrap is a gradient-descent algorithm on an objective function defined on the vertices of the triangle [1]. The objective function is the sum of two terms: first term is the volume of the simplex to be minimised and second term is a penalty term which has the effect of 'pushing' the faces of the simplex away from the data points. The gradient of each of these terms is determined analytically, and used in the gradient descent algorithm. The penalty term includes a multiplicative constant which approaches zero as the gradient descent algorithm progresses, causing the iterates to converge to the vertices of a simplex which fits tightly around the given data points. This process begins with a large initial simplex and shrinks it down around the data cloud. Intuitively, one can imagine the process as the multidimensional analogy of a wrapping a sheet of plastic around a convex set containing the data; hence the term 'Shrink-Wrap'. The problem addressed here is that of finding the simplex which gives the best fit to a given set of data points, where a best fit simplex implies to be one with minimum area, subject to the constraint that all the data vectors lie in the interior [1]. While the first term in the objective function is there to be minimised, the second term is slightly different. It is a penalty function chosen to enforce the constraint that all the data lie in the interior of the simplex. Qualitatively, it means that this penalty function should be small when the simplex is large and its faces are far from the data points, and conversely it should be large, as any simplex face gets near the data points. The behaviour of the Simplex Shrink-Wrap algorithm is illustrated in Figure 4-1. The first panel shows the data cloud and the true endmembers projected into the plane; the second panel shows the initial simplex; the third panel shows the sequence of iterates (sampled every 10 iterations to improve clarity) and the final panel shows the estimated simplex at the last iteration. The manner in which the shrinking simplex rotates and aligns itself with the shape of the data cloud is of particular interest. However the problem addressed in the paper is somewhat idealised and one can imagine the effect that outliers will have on this triangle wrapping algorithm because of its constraint that all the data points to lie within the triangle.

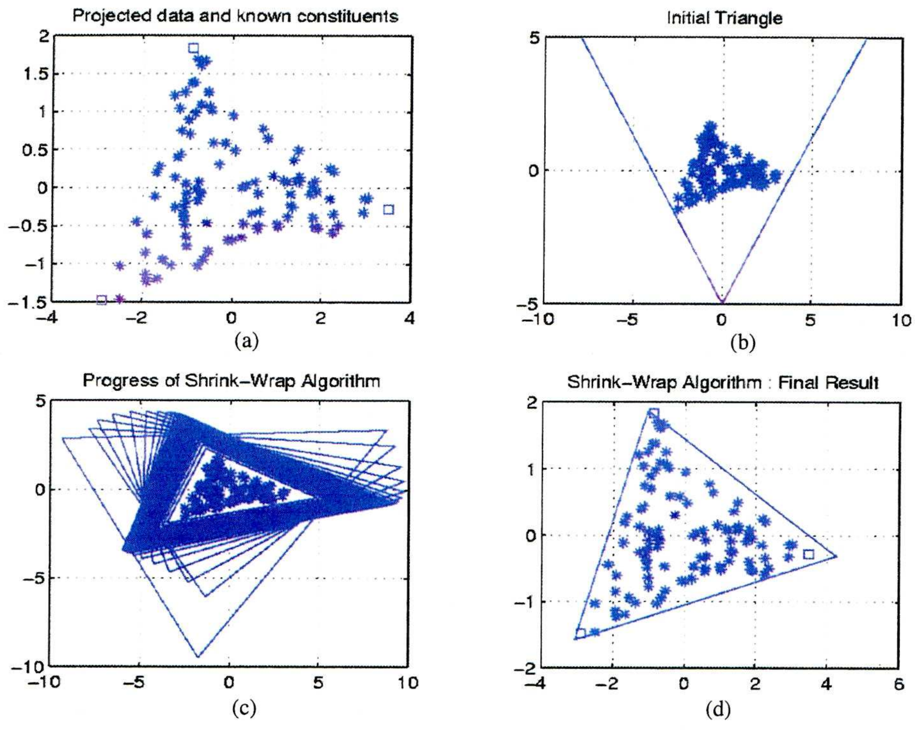


Figure 4-1 : Behaviour of the Simplex Shrink Wrap. (a) Available data points (b) Initial triangle (c) Progress of the wrapping algorithm, triangles superimposed on each other (d) Final triangle Image source [1], pp 505

Another triangle wrapping algorithm popular in the remote sensing community is the Minimum Volume Transform (MVT). The MVT method starts with a large initial triangle and reduces it down to shrink it round the data. It does so by repeatedly varying the orientation of one facet at a time while all the data points are embraced by the simplex all the time until the minimum-area simplex is obtained [88]. The MVT needs at least one vertex, called the 'dark point', to be known beforehand. To achieve this, it applies a linear transformation to move the data points in such a way that they become confined to the corner of two lines joining each other. A third line is then added to complete a triangle with a view to obtain one with a minimum area. This technique might have reasonably been termed a minimum-area transform but the term 'minimum-volume' (MVT) is used because that paper was based on a three-dimensional case. For consistency across all dimensions, one might also think that the term ‘minimum-content’ would be more accurate.

A third algorithm that often comes in the satellite reconnaissance literature is the N-FINDR algorithm [89]. The N-FINDR algorithm works from the 'inside-out', i.e., it takes a set of points in the data set and then 'pushes' the triangle until a maximum area triangle is found. During the optimisation process, the N-FINDR finds those vertices by randomly selecting a set of *P* pixels from the scene as initial endmembers, and calculating the volume of the simplex formed by these initial endmembers. This process is iterated through the following steps to test every pixel in the image as an endmember. First, each of the initial endmembers is replaced one at a time with the pixel being tested. Second, the volumes (or areas for 2-D) of the simplexes formed by each replacement are calculated. Finally, the algorithm evaluates if replacing any of the initial endmembers with the pixel being tested results in a larger simplex volume. If this is the case, the pixel being tested replaces the initial endmember and the process is repeated again until each pixel is evaluated as a potential endmember. This procedure is repeated until there are no more replacements of endmembers that can result in an increase in area. The pixels which remain as endmembers at the end of the process are considered to be the final endmembers. This algorithm’s efficiency is dependent on the number of points contained in the data cloud and if there are too many points, it can take a long time to give the maximum area. Implementing N-FINDR can require very high and expensive computational complexity because of the exhaustive search for optimal *P* endmembers simultaneously among all possible *P*-endmember combinations in the data. When the data sample vectors are huge, the computational cost can be unmanageable and it may take quite long time to converge to a desired set of endmembers. To try to decrease this cost, modifications to the search process can be brought to the N-FINDR algorithm that do not generally conduct a fully exhaustive search, but rather focus on endmembers selected from some feasible regions for iterations. However, in order for such modified algorithms to be also effective, initial endmembers must be representative and cannot be arbitrary. Therefore, a judicious selection of initial endmembers is necessary.

These techniques mentioned so far are designed for static satellite imagery and its accompanying characteristics. In this thesis, another triangle wrapping technique is developed that answers the requirements of noisy and quickly changing data points brought about by changes in room lighting or poor image sensors. Moreover the un-mixing is done without prior knowledge of any endmember location. The optimisation procedure has to be time dependent and deal with highly dynamic situations. The method used here is closest to that of the Simplex Shrink-Wrap because a minimum area triangle is sought and the objective function in the optimisation process is based on the vertices. A multi-objective function based on the area of the triangle and the number of points outside the triangle is defined to be minimised. However, unlike the Simplex Shrink-Wrap, there is no constraint to put all the points inside the triangle. The data points used here are assumed to come from the Normalised Red and Green channels of an image using the Normalised RGB colour space.

The objectives for the triangle wrapping are twofold; to fit a triangle around as many data points as possible ignoring noisy outliers, while at the same time keeping the triangle to small finite proportions. This is a multi-objective optimisation, which can be implemented through a weighted-sum approach. The overall minimising cost function is defined as:

$$f(\vec{x}) = \text{Area}(\vec{x}) + w \cdot [\text{H} \cdot \text{W-Points}(\vec{x})] \tag{4-1}$$

where $\vec{x} = (x_A, x_B, x_C, y_A, y_B, y_C)$ is a vector composed of the triangle's vertices coordinates. Area($\cdot$) is a function calculating the overall area of the defined triangle, while Points($\cdot$) returns the number of points within the triangle. H and W are the width and height of the image plane. The non-negative term $[\text{H} \cdot \text{W-Points}(\vec{x})]$ helps in minimising the number of points outside the triangle. $w > 0$ is a user-defined weight balancing the two objectives which are of different scales and units. In this way, when $F(\cdot)$ is minimised, the smallest triangle that optimally contains most of the data points is found while the effect of outliers is mitigated.

4.2.1 Multi-objective optimisation

Multi-objective optimisation, also known as multi-criteria, is the process of simultaneously optimising two or more conflicting objectives subject to certain constraints [90]. This approach can be found in various fields wherever optimal decisions need to be taken in the presence of trade-offs between two conflicting objectives, e.g. maximising profit and minimising the cost of a product, and minimising weight while maximising the strength of a particular component. The term 'optimise' means finding such a solution which will give the values of all the objective functions acceptable to the decision maker. These objective functions may also be commensurable (measured in the same units) or non-commensurable (measured in different units) [90]. Without loss of generality, all objectives are of the minimisation type - a minimisation type objective can be converted to a maximisation type by multiplying by negative one. A minimisation multi-objective decision problem with M objectives is defined as follows:

$$\left.\begin{array}{lll} \text{Minimise} & \mathrm{f}_m(\mathrm{x}), & m = 1,2,\ldots,M; \\ \text{subject to} & x_i^{(\mathrm{L})} \le x_i \le x_i^{(\mathrm{U})}, & i = 1,2,\ldots,n. \end{array}\right\} \quad (4\text{-}2)$$

where x is a vector of n decision variables, $\mathrm{x} = [x_1, x_2, \ldots, x_n]^T$, each subject to take a value within lower bound $x_i^{(\mathrm{L})}$ and upper bound $x_i^{(\mathrm{U})}$. The idea is to find a vector x* that minimises a given set of M objective functions $\mathrm{f}(\mathrm{x}) = \{f_1(\mathrm{x}), f_2(\mathrm{x}), \ldots, f_M(\mathrm{x})\}$ within a solution space called objective space X. The predominant solution concept in defining solutions for multi-objective optimisation problems is that of Pareto optimality [90]. A solution in the feasible solution space is called Pareto optimal if there is no other feasible solution in the solution space that reduces at least one objective function without increasing another one [90]. A Pareto optimal solution cannot be improved with respect to any objective without worsening at least one other objective. The set of all feasible solutions in X is referred to as the Pareto optimal set, and for a given Pareto optimal set, the corresponding objective function values in the objective space are called the Pareto front. One goal in multi-objective optimisation is to find a set of solutions as close as possible to the Pareto-optimal front.

For many problems, the number of Pareto optimal solutions is enormous (perhaps infinite). To solve a particular multi-objective problem, one approach is to construct a single aggregate objective function (AOF) from the multiple objectives that exist. The basic idea is to combine all of the objective functions into a single functional form, called the AOF [91]. A well-known combination is the weighted linear sum of the objectives. One specifies scalar weights for each objective to be optimised, and then combines them into a single function that can be solved by any single-objective optimiser (examples of such optimisers are given in next sub-section). Clearly, the solution obtained will depend on the values (more precisely, the relative values) of the weights specified. The solutions obtained using the weighted sums are always Pareto optimal, but coming up with meaningful combinations of weights can be challenging. In practice, it can be very difficult to precisely and accurately select these weights, even for someone familiar with the problem domain. Compounding this drawback is that scaling amongst objectives is needed and small perturbations in the weights can sometimes lead to quite different solutions. In equation 4-1, the AOF $\text{Area}(\vec{x}) + w \cdot [\text{H} \cdot \text{W-Points}(\vec{x})]$ with a weight w is obtained from the two functions $\text{Area}(\vec{x})$ and $[\text{H} \cdot \text{W-Points}(\vec{x})]$ respectively. The first function calculates the area of a triangle and this value can be between 0 and 0.5 square units in magnitude if a Normalised RGB colour space is used. The second function calculates the number of points that are outside the triangle and its value can be any value from zero to the number of pixels in the image (e.g. maximum of 900 values in a 30×30 image). Although the value of 900 is very unlikely as this would mean that the triangle is not enclosing any point, a value of anything less than 20 is very likely when there are noisy outliers in the data. Hence to be able to balance the two functions, a 'small' value of less than one is needed and this value can only be obtained by trial and error unfortunately.

Another approach to solve the multi-objective function is the Multi-Objective Evolutionary Algorithm (MOEA) approach. One example of an evolutionary algorithm is the Genetic Algorithm (GA) that was developed by Holland et al. in the 1960s and 1970s [94]. GAs are inspired by the evolution theory explaining the origin of species and are described in deep details later in sub-section 4.2.2.3. Being a population-based approach, GAs are well suited to solve multi-objective optimisation problems [90]. The ability of GAs to simultaneously search different regions of a solution space makes it possible to find a diverse set of solutions for difficult problems with non-convex and/or discontinuous solution spaces. In addition, if enough information is available, multi-objective GAs do not require the user to prioritise, scale, or weigh objectives. However they are more complex and time-consuming to set up than the weighted sum approach.

4.2.2 Optimisation algorithms

With the multi-objective function finalised, an optimisation search technique is required to minimise it. The choice depends on the type of problem under consideration and optimisation algorithms can be divided into two broad categories [93]:

- *Gradient-based algorithms*, such as the Robbins-Monro stochastic approximation algorithm can be considered to be a generalisation of the deterministic steepest descent. It requires that direct measurements of the gradient are available, but these measurements are generally a gradient estimate because the underlying data is usually noisy [93].
- *Gradient-free algorithms* such as the simple random search, the Nelder-Mead method, the Simulated Annealing or the genetic algorithm method. These methods can be useful for a broad search over the domain of the parameters being optimised, and can provide initialisation for a more powerful local search algorithm.

Approaches based on the use of gradient estimations tend to be fast, but are sensitive to the presence of local optima. Additional discussion of these methods with their relative advantages and disadvantages can be found in [94]. In order to minimise Equation 4-1 for the purpose of this thesis, three gradient-free optimisation techniques are tested and - where possible - the following constraints are enforced, namely that the values of the vertices are always within 0 and 1 (as these were the normalised colour-space image boundaries) and secondly that the area of the triangle can not be higher than 0.5 square units (as the maximum area of a triangle within a unit square is 0.5). The three optimisation methods tested are:

(1) stochastic search method

(2) deterministic method

(3) genetic algorithm method.

4.2.2.1 Stochastic search method

The stochastic method used is a directed random search algorithm. Stochastic search methods for optimisation are based on exploring the domain Θ in a random manner to find a point that minimises $f = f(\vec{x})$, where f is the cost function. The advantages of this method include relative ease of coding in software, the need to only obtain F measurements (versus gradients or other ancillary information), reasonable computational efficiency (especially for those direct search algorithms that make use of some local information in their search), broad applicability to non-trivial loss functions and/or to $\vec{x}$ that may be continuous, discrete, or some hybrid form, and a strong theoretical foundation [95]. The simplest random search method is the 'blind random search' and it can be described as in the following algorithm:

Step 0 (initialisation) Choose an initial value of $\vec{x} = \vec{x}_0$ inside of Θ. Set $k = 0$.

Step 1 (candidate value) Generate a new independent value $\vec{x}_{\text{new}}(k+1) \in \Theta$, according to the chosen probability distribution. If $F(\vec{x}_{\text{new}}(k+1)) <$ $F(\vec{x}_k)$, set $\vec{x}_{k+1} = \vec{x}_{new}(k+1)$. Else take $\vec{x}_{k+1} = \vec{x}_k$

Step 2 (return or stop) Stop if maximum number of F evaluations has been reached or user is otherwise satisfied with the current estimate for $\vec{x}$; else, return to step 1 with the new k set to the former $k + 1$.

This algorithm is unique among all general stochastic optimisation algorithms in that it is the only one without any adjustable algorithm coefficients that need to be "tuned" to the problem at hand. It also converges almost surely (a.s.) to the optimum $\vec{x}^*$ under very general conditions [94]. While blind random search is a reasonable algorithm when $\vec{x}$ is low dimensional, it can be shown that the method is generally a very slow algorithm for even moderately dimensioned $\vec{x}$. This is a direct consequence of the exponential increase in the size of the search space as the problem dimension p increases [94]. It also does not adapt the current sampling strategy to information that has been garnered in the search process.

An improvement on this method is called the 'Enhanced Localised Random Search' method whereby the search is more localised in the neighbourhood of an estimate, allowing for a better exploitation of information that has previously been obtained about the shape of the objective function. The following is a simplified version of the full version of the algorithm as described by Solis and Wets [96]:

Step 0	(initialization) Choose an initial value of $\vec{x} = \vec{x}_0$ inside of Θ. Set $k = 0$
Step 1	(candidate value) Generate a random d_k and a bias term b_k. Check if $\vec{x}_k + b_k + d_k \in \Theta$. If not, generate new d_k or move $\vec{x}_k + b_k + d_k$ to nearest valid point. Let $\vec{x}_{new}(k+1)$ be $\vec{x}_k + d_k + b_k$ or the modified point.
Step 2	(check for improvement) If $F(\vec{x}_{\text{new}}(k+1)) < F(\vec{x}_{\text{k}})$, set $\vec{x}_{k+1} = \vec{x}_{new}(k+1)$ and $b_{k+1} = 0.2b_k + 0.4d_k$ and go to Step 5. Otherwise, go to Step 3.
Step 3	(candidate value) Use the random d_k and a bias term b_k of Step 1. Let $\vec{x}'_{new}(k+1) = \vec{x}_k + b_k - d_k$ or its nearest valid point within Θ. If $F(\vec{x}'_{\text{new}}(k+1)) < F(\vec{x}_{\text{k}})$, set $\vec{x}_{k+1} = \vec{x}'_{new}(k+1)$ and $b_{k+1} = b_k - 0.4d_k$ and go to step 5. Otherwise, go to Step 4.
Step 4	(update bias) Set $\vec{x}_{k+1} = \vec{x}_k$ and $b_{k+1} = 0.5b_k$. Go to Step 5.
Step 5	(return or stop) Stop if maximum number of F evaluations has been reached or if user satisfied with current estimate; else, return to step 1 with new k set to former $k + 1$.

This algorithm allows one to focus the search more tightly as evidence is accrued on the location of the solution. It also exploits the knowledge on 'good' or 'bad' directions. For example, if a move in one direction produces a decrease in loss, bias is added to the next iteration to allow the algorithm to continue moving in 'good' direction or similarly if a move in one direction produces an increase in loss, bias is added to the next iteration to move the algorithm in the opposite way. This algorithm is chosen to be the stochastic search method implemented in this thesis.

4.2.2.2 Deterministic method

The deterministic method used is the gradient free Nelder-Mead nonlinear algorithm [97]. It is based on the concept of a simplex, a geometric object (convex hull) of $p + 1$

vertices enclosing all the $p + 1$ points in p dimensions, e.g. a line segment on a line, a triangle on a plane, a tetrahedron in three-dimensional space and so forth. This algorithm is the baseline gradient-free multivariate optimisation technique in MATLAB (function fminsearch [98]) and is one of the most popular optimisation techniques when comparison of objective function values only is concerned. The basic idea in this method is that at each iteration a new point is generated in or near the current simplex. The new point then replaces one of the current simplex vertices, yielding a new simplex. This new point is generated by certain transformations such as reflection, expansion and contraction as detailed in the algorithm description in Figure 4-2.

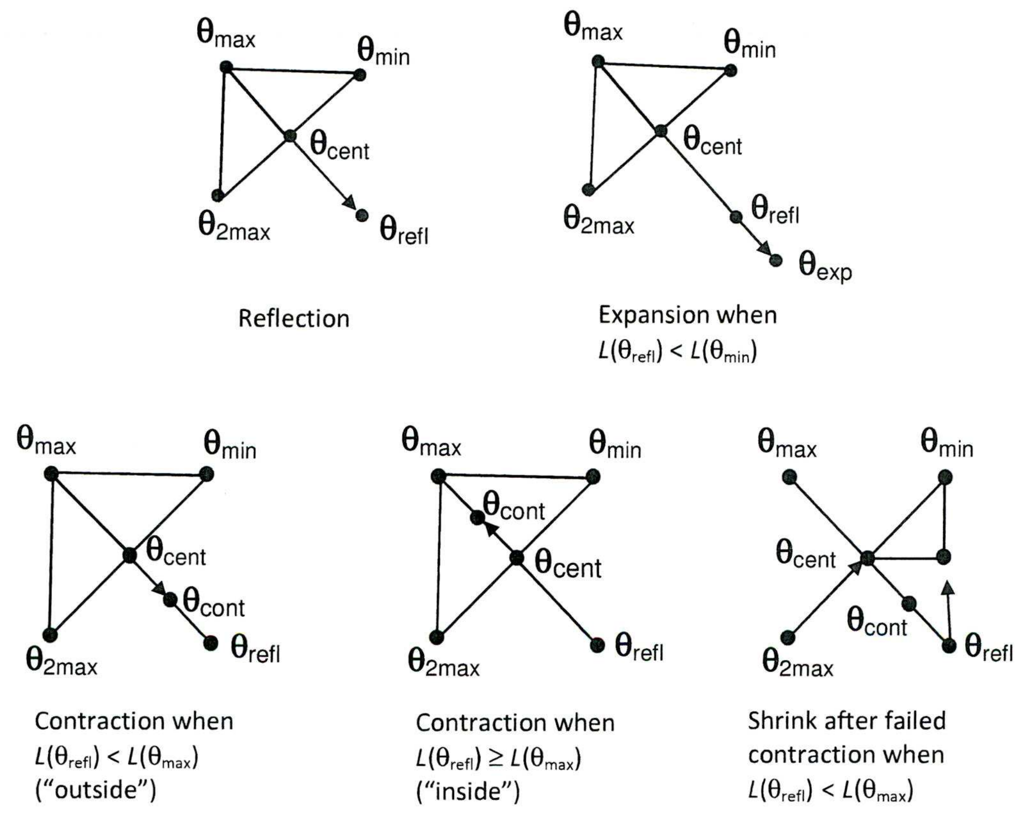


Figure 4-2 : Nonlinear simplex algorithm for p = 2, where θ is the parameter vector (adapted from [99])

The algorithm steps shown here are identical to those in MATLAB fminsearch function. These steps differ slightly from the original Nelder-Mead algorithm but they have become widely accepted because performance has been improved. A full description of the algorithm is available in Spall's book [94, p. 57]. To put it simply, the algorithm searches for $\vec{x}*$ by moving the convex hull within Θ. If the algorithm works properly, the convex hull shrinks or collapses onto $\vec{x}*$. One has to also note that there is no randomness injected in this technique. This method is very often effective in both noise free and noisy function measurements, but there is no general convergence theory and there are many numerical counter-examples where it is shown that it does not converge.

4.2.2.3 Genetic algorithm method

The third method to be evaluated is a standard genetic algorithm (GA) which is a good method for solving both constrained and unconstrained optimisation problems. GAs are the most popular evolutionary computation (EC) algorithms and are based on principles of natural selection and survival of the fittest whereby at each step individuals are selected probabilistically from the current population to be parents and then are used to produce the new solutions (offspring) for the next generation [100]. In GA terminology, a cost function such as, $F(\vec{x})$, is often referred to as the fitness function to emphasise the evolutionary concept of the fittest of a species having a greater likelihood of surviving and passing on its genetic material [94].

A fundamental difference between GAs and the previous two methods is that GAs work on a population of encoded solutions. A GA can simultaneously consider multiple candidate solutions to a minimisation of F and iterate by moving this population of candidate solutions towards a global optimum as shown in Figure 4-3. This is motivated by a hope, that the new population will be better than the old one. Solutions which are selected to form new solutions (offsprings) are selected according to their fitness - the more suitable they are the more chances they have to reproduce. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

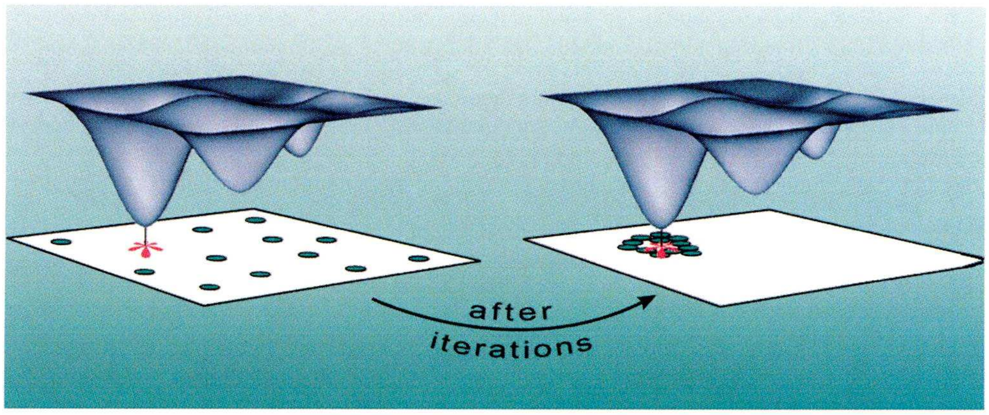


Figure 4-3 : Minimisation of a cost function. Successful operations of a GA with a population of 12 candidate solutions clustering around the global minimum after a number of iterations or generations [94]

The basic outline of this algorithm can be described as below [99]:

Step 0 [Start] Generate random population of *n* chromosomes (suitable solutions for the problem)

Step 1 [Fitness] Evaluate the fitness *F(x)* of each chromosome *x* in the population

Step 2 [New population] Create a new population by repeating following steps until the new population is complete

- **A.** [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
- **B.** [Crossover] With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
- **C.** [Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).
- **D.** [Accepting] Place new offspring in a new population

Step 3 [Replace] Use new generated population for a further run of algorithm

Step 4 [Test] If the end condition is satisfied, stop, and return the best solution in current population

Step 5 [Loop] Go to step 1

GAs are relatively new optimisation techniques and so far they have been shown to be particularly useful in solving timetabling and scheduling problems [101], engineering problems and global optimisation problems. They are also used in the fields of computer science and artificial intelligence where their ability to solve complicated optimisation problems together with the mystique surrounding them have contributed to make them very popular. However one must be careful not to make exaggerated claims concerning GAs as there appears to be no formal evidence of consistently superior performance relative to other stochastic algorithms. As disadvantage, the use of a population versus a single solution affects the range of practical problems that can be considered. For example, a GA is not generally suited to working with real-time physical experiments. GAs usually require more function evaluations than the other two techniques to reach the same cost function values, i.e. convergence may be slow. In some cases, GAs may have a tendency to converge towards local optima or even arbitrary points rather than the global optimum of the problem. The likelihood of this occurring depends on the shape of the fitness landscape and this problem may be alleviated by using a different fitness function, increasing the rate of mutation, or by using selection techniques that maintain a diverse population of solutions.

4.2.3 Triangle optimisation on static data

The next step is to test the optimisation algorithms on static artificial data that resemble a triangular distribution using the multi-objective cost function shown in equation 4-1. All three techniques obtain their starting triangle vertex values by identifying the centre coordinates of three regions of high point density after using a k-means clustering algorithm [102]. However it can happen that three distinct regions of high point density are not available due to high levels of noise or the fact that the object of interest has not yet entered the field of view of a camera monitoring a room with two main background colours. To get round this, permutations of the maximum and minimum values of the data plots are used to obtain candidate starting triangles, and then a choice is made depending on which triangle is enclosing the highest number of points as shown in flowchart in Figure 4-4. Although all the optimisation techniques should in theory be able to find a triangle to wrap closely around the data points given enough function evaluations, this informed initialisation help a lot in accelerating the search. After the starting points are obtained, the triangle optimisation procedure can then be started. However before going on to discuss the results of the optimisation techniques, it is worth noting that the cost function itself, i.e. $f(\vec{x}) = \text{Area}(\vec{x}) + w \cdot [\text{H} \cdot \text{W} - \text{Points}(\vec{x})]$, has to tested first to see if it is a well defined function for this problem situation.

$f(\vec{x})$ contains two main calculations that have to be implemented. These are $\text{Area}(\vec{x})$ and $\text{Points}(\vec{x})$ respectively, i.e. how to find area of a triangle and how to find the number of points that are enclosed by a triangle. The area of a triangle with vertices A, B and C can be obtained using vector products given by:

$$\text{Area} = \frac{1}{2} \times \left| \overrightarrow{AB} \times \overrightarrow{AC} \right| \qquad (4\text{-}3)$$

To find the number of points that are enclosed inside a triangle, two main methods are possible. The first one makes use of the inherent property that a point inside a triangle will divide the triangle into three smaller triangles and that the sum of the areas of these three triangles will be equal to the area of the main triangle as shown in Figure 4-5. The second method involves using the *delaunay* and *tsearch* functions available in MATLAB. Given a set of data points, the Delaunay triangulation is a set of lines or edges connecting each point to its natural neighbours satisfying an 'empty circle' property, i.e. for each edge a circle containing the edge's endpoints can be found but not containing any other points [103]. The MATLAB implementation of Delaunay is based on the QuickHull(qhull) technique of doing shapes with convex shapes [104]. The QuickHull algorithm computes the convex hull of a set of points in two or more dimensions, where a convex hull of a set of points is the smallest convex set that includes the points. Both techniques are implemented in MATLAB and it is found that the second one involving the use of the *delaunay* function (i.e. qhull method) performs a lot better in terms of time spent in calculating the number of points inside a triangle. This method is hence chosen as the preferred technique.

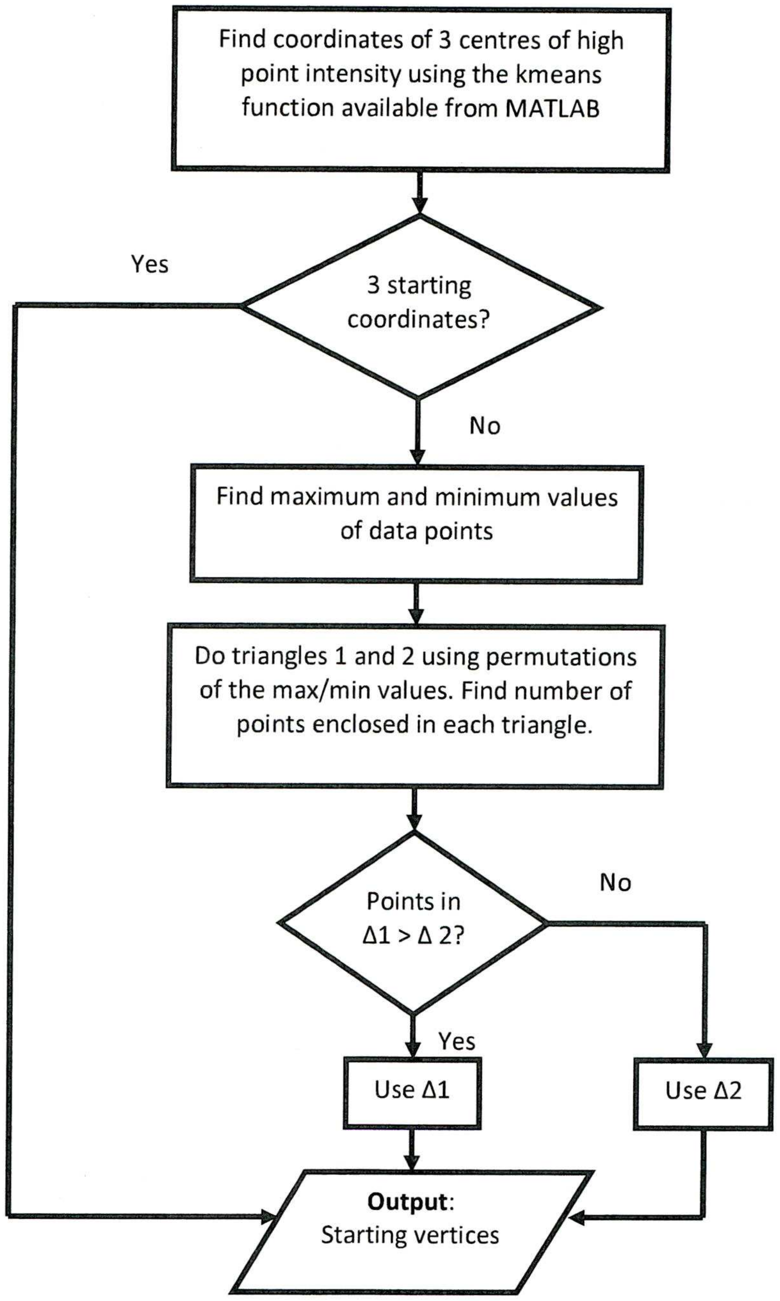

Find coordinates of 3 centres of high point intensity using the kmeans function available from MATLAB
3 starting coordinates?
Yes
No
Find maximum and minimum values of data points
Do triangles 1 and 2 using permutations of the max/min values. Find number of points enclosed in each triangle.
Points in Δ1 > Δ 2?
No
Yes
Use Δ1
Use Δ2
Output: Starting vertices

Figure 4-4 : Flowchart to find the starting vertices of the triangle to be used in the optimisation procedure.

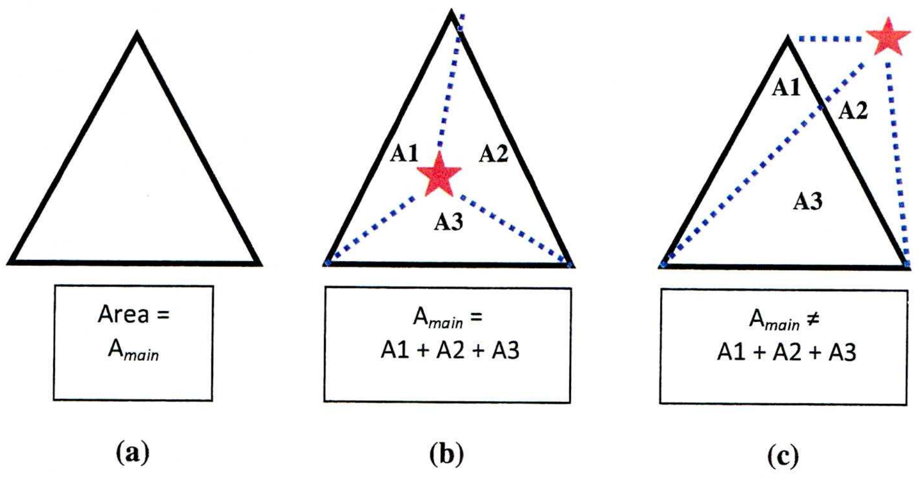


Figure 4-5 : Determining whether a point lies inside a triangle or not by using the sum of the areas of the sub-triangles obtained.

The process of evaluating if the objective function is properly defined can then carried out. This is done by applying the stochastic search optimisation technique on the objective function with synthetic data generated for this purpose. The synthetic data consisted of a certain number of points with 2-D coordinate values ranging between 0 and 1 and the points were distributed around three imaginary corners of an ideal triangle. The results are shown in Figure 4-6 and Figure 4-7. One can see how the optimiser is trying to fit a triangle around as many data points as possible after each iteration process while at the same time decreasing the error in the distance from the ideal vertices that will enclose all the data points with a minimum area. The objective function value also follows a decreasing trend and hence its validity is confirmed.

With the objective function's validity established, the three optimisation techniques mentioned in the previous section are then evaluated. A small experiment is devised in which each technique is given a set of data points that roughly forms a triangular shape when plotted on 2-D space and then each technique's performance is evaluated after running the experiment 100 times. A weight w of small magnitude 0.001 and a maximum number of iterations (generations for the GA) of 20 are used for all the three techniques used to minimise the cost function $f(x)$.

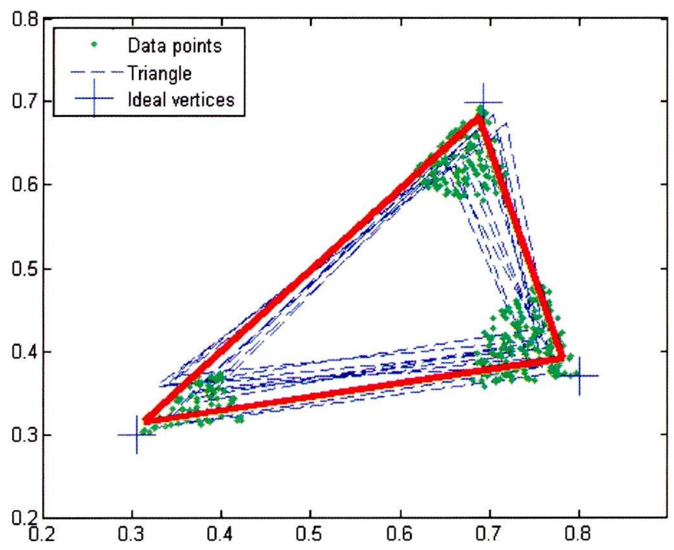


Figure 4-6 : Triangle obtained (dotted-line sides) at each step of the stochastic optimisation iteration process. Final triangle obtained after 20 iterations shown in thick lines.

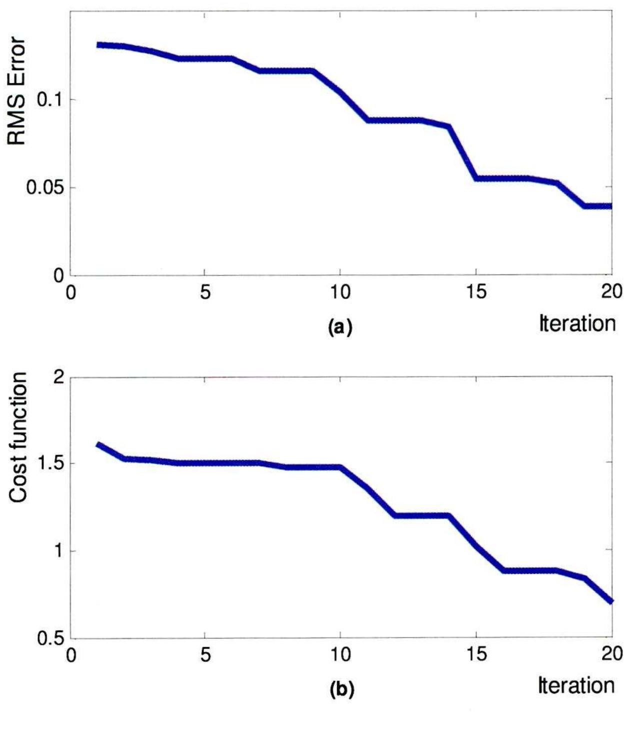


Figure 4-7 : (a) The error in the distance from the ideal vertices' position and (b) Cost function values as the stochastic search optimisation process evolves.

The stochastic optimiser is implemented using the algorithm described in section 4.2.2.1 with an explanatory flowchart shown in Figure 4-8 while the deterministic and GA optimisers are available directly from the MATLAB Optimisation Toolbox as *fminsearch* and *ga* functions respectively. The *fminsearch* function can be used as in the following syntax: `[vertices, fval] = fminsearch(@CostFunction, vertices, options);` This particular command starts at the point `vertices` and attempts to find a local minimum of the function described in `CostFunction.` It minimises the cost function with the optimisation options such as the termination tolerances on the function value and the number of iterations allowed specified. The final output consists of the optimised value of the vertices and the cost function. Repeating the Nelder-Mead optimisation procedure on the same data points would result in the same final triangle because of its deterministic nature and so to test for the stability of this particular method, a random number is added to the starting vertices for each run. Concerning the genetic algorithm implementation, the *ga* function can be executed from the MATLAB command line as with the following syntax: `[vertices fval] = ga(@fitnessfun, nvars, options);` where `nvars` is the number of independent variables for the fitness function (here 6 values for the x and y coordinate values of each vertex) and `options` contains certain settings such as population size, crossover fraction, migration direction and tolerance values. An initial population of 30 vector values close to the starting vertices is used as initial candidate solutions for the GA implementation. The implementation for these three methods is available in appendix A1.

The results for both a random run and the average of multiple runs are shown in Figure 4-9. It can be observed that all the techniques on average decrease the value of the objective function in a smooth way as the number of iterations increase. The random optimiser converge less quickly than the other two, while between the random optimiser and the deterministic method, the latter is found to give smaller objective function values. On the other hand, the Genetic Algorithm gives the best fitness function values and most points inside triangle. It has to be pointed out that each algorithm performs a different number of function evaluations for each iteration and hence the amount of computation involved at each iteration cannot be directly compared.

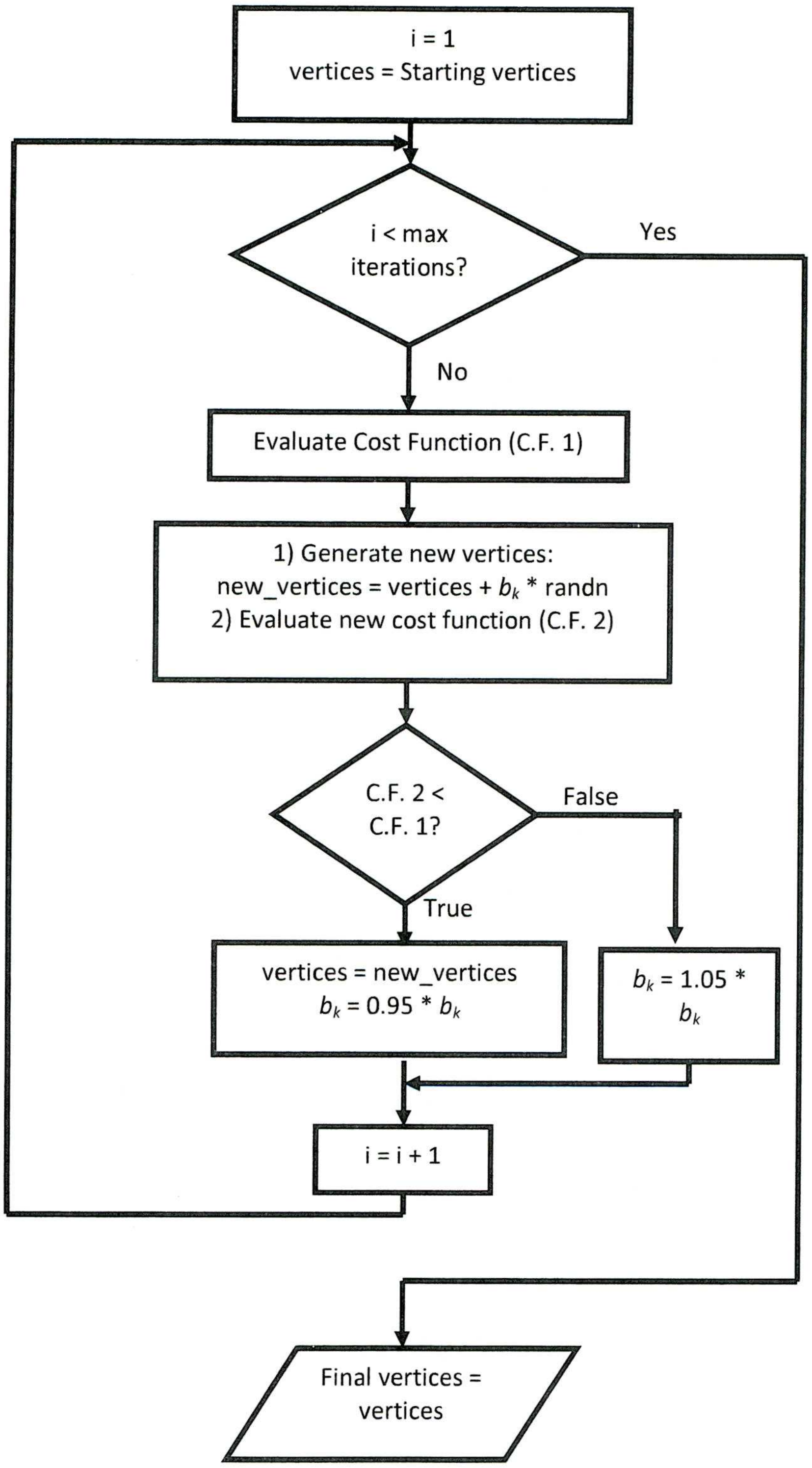

i = 1
vertices = Starting vertices
i < max iterations?
Yes
No
Evaluate Cost Function (C.F. 1)
1) Generate new vertices:
new_vertices = vertices + b_k * randn
2) Evaluate new cost function (C.F. 2)
C.F. 2 < C.F. 1?
False
True
vertices = new_vertices
b_k = 0.95 * b_k
b_k = 1.05 * b_k
i = i + 1
Final vertices = vertices

Figure 4-8 : Flowchart showing how the stochastic optimiser was implemented.

These tests also demonstrate the usefulness of the user-defined weight *w* balancing the two objectives which are of different scales and units. In Figure 4-9 (c) and (e), it can be seen how a smaller triangle (e) can contain more points than in one with a larger area shown in (c). In this way, one can suggest that the idea of finding the smallest triangle that optimally contained most of the data points by wrapping around them has been achieved.

While these methods worked fine on this static test data, one has to take into consideration that the application this optimisation is targeted at is one with a dynamic environment where the colour balance is constantly changing. This is a time-varying problem and hence a dynamic programming approach to optimisation can be more suitable to satisfy these conditions. Dynamic programming is an algorithmic technique that computes solutions by solving simpler overlapping sub-problems and it is was developed by the mathematician Richard Bellman who described the way of solving problems where best decisions have to be found one after another [105]. In the forty-odd years since this development, the number of uses and applications of dynamic programming has increased enormously. For this current case, since the optimisation has to be done on each frame with the latter changing ten times every second (10 fps), the optimised vertices of one frame can also be used as *a priori* information for the next frame to be optimised. Section 4.2.4 gives more information about the choice of the ideal optimiser to deal with the dynamic situation. As a word of caution to avoid any confusion, the word 'programming' in 'dynamic programming' has no particular connection to computer programming at all, and instead comes from the term 'mathematical programming', a synonym of optimisation.

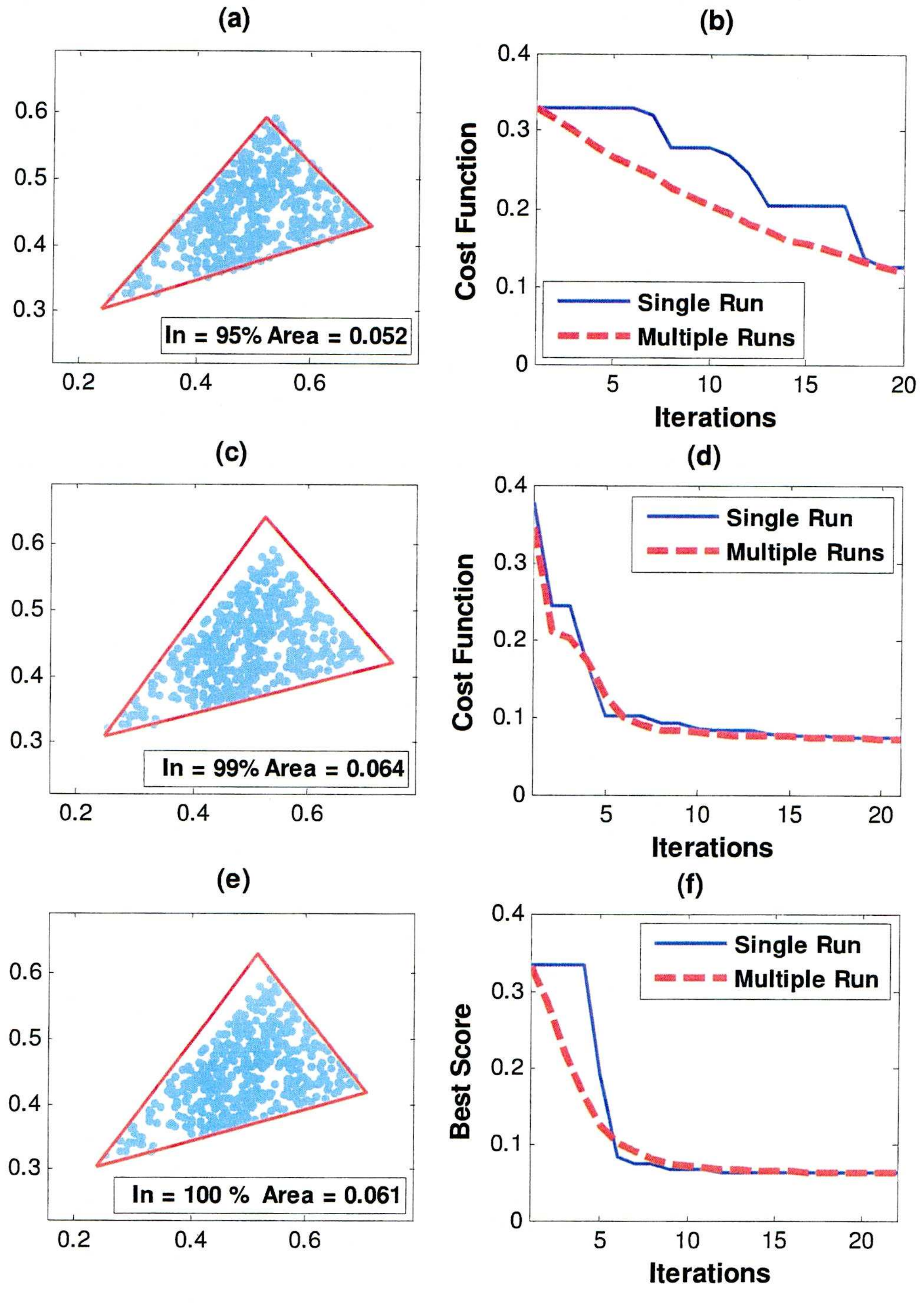


Figure 4-9: (a) and (b) Random Search, (c) and (d) Nelder – Mead, (e) and (f) Genetic algorithm. Evaluating different optimisation techniques

4.2.4 Triangle optimisation and un-mixing on synthetic dynamic data

This section begins with the explanation of how the optimisation process is further modified to adapt to the quickly changing colour balances that occur in dynamic environments. It then describes and discusses the results of the various experiments that are performed. A model of an image sequence that contains a moving object similar to what will be obtained from of a low resolution camera is implemented. A high resolution image matrix is first created and the high resolution image sequence is then down sampled so that the size of the object ranges from a few pixels to less than a pixel in size in the down-sampled low resolution synthetic image (Figure 4-10). Typical values for the example of the model used are 500×500 pixel size high resolution image background and an object of circular or square shape of size of around 20×20 pixels in the beginning. These values are chosen assuming that the field of view of the sensor camera is $5\text{m} \times 5\text{m}$, and that a human head with diameter of 20 cm being tracked. The down-sampling factor is initially of order 25 which results in a series of low resolution frame images of size 20×20, and the moving object being less than 1 pixel of size. A constant random image noise over all the pixels, typical of electronic noise, is also added to make the simulated imagery as close as possible to reality. The background images are also changed during the experiment to evaluate how background colours affect the detection of the small target. The object size and the down-sampling factor are varied, and different object speeds are also tried to see how the detection and tracking process respond at different rates of displacement.

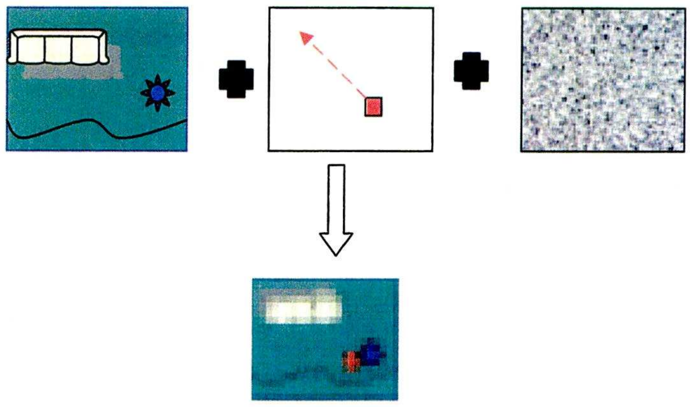


Figure 4-10: High resolution artificial imagery with noise being down-sampled.

To be able to use the un-mixing technique, the triangle optimisation procedure described in the previous section has to be evaluated on a sequence of frames rather than single images. To carry out this evaluation, a moving coloured object on a background with two major colours is first generated together with some image noise for a total of 250 frames at a rate of 10 frames per second. The changing data point distribution thus obtained is then subject to the three optimisation techniques on each frame. It is found that the Nelder-Mead (NM) approach fails when the starting points are not close to being ideal values, i.e. starting coordinates not lying near the extremities of the point distribution. In a sequence of frames where the data point distribution is constantly changing resulting in the starting points being rarely ideal, this method cannot perform well as a result. To be able to use the Nelder-Mead method in a dynamic environment, the starting vertices will have to be calculated for each frame and the optimiser working on each frame separately. This approach will however defeat the purpose of dynamic programming. The random search optimiser provides a way of obtaining a triangle wrapped around changing data points with a low computational complexity but this method often fails when the data changes very rapidly and hence result in the vertices of the triangle being far from the actual 'corners' of the data points. The GA is more robust and its robustness is further improved when a few modifications are made to the optimiser to better deal with the changes that occur between frames. As with the static GA optimisation, an initial population of candidate solutions is fed to the optimiser at the first frame but the difference this time being that the best scores obtained at each GA optimisation step are fed back into the optimiser for the next frame to achieve a better frame to frame propagation as part of the modifications as depicted in flowchart of Figure 4-11. The population size of 20 chosen here is completely a random choice and one can increase or decrease that size at the expense or gain in timing efficiency. A reasonable amount was found to be any value between 20 and 40. Too small a value will not allow the frame to frame propagation process to actually make its presence felt while too large a value will cause the optimisation to last longer than the time taken for a frame to change and hence of no use when it comes to using the population values of a current frame for the next frame.

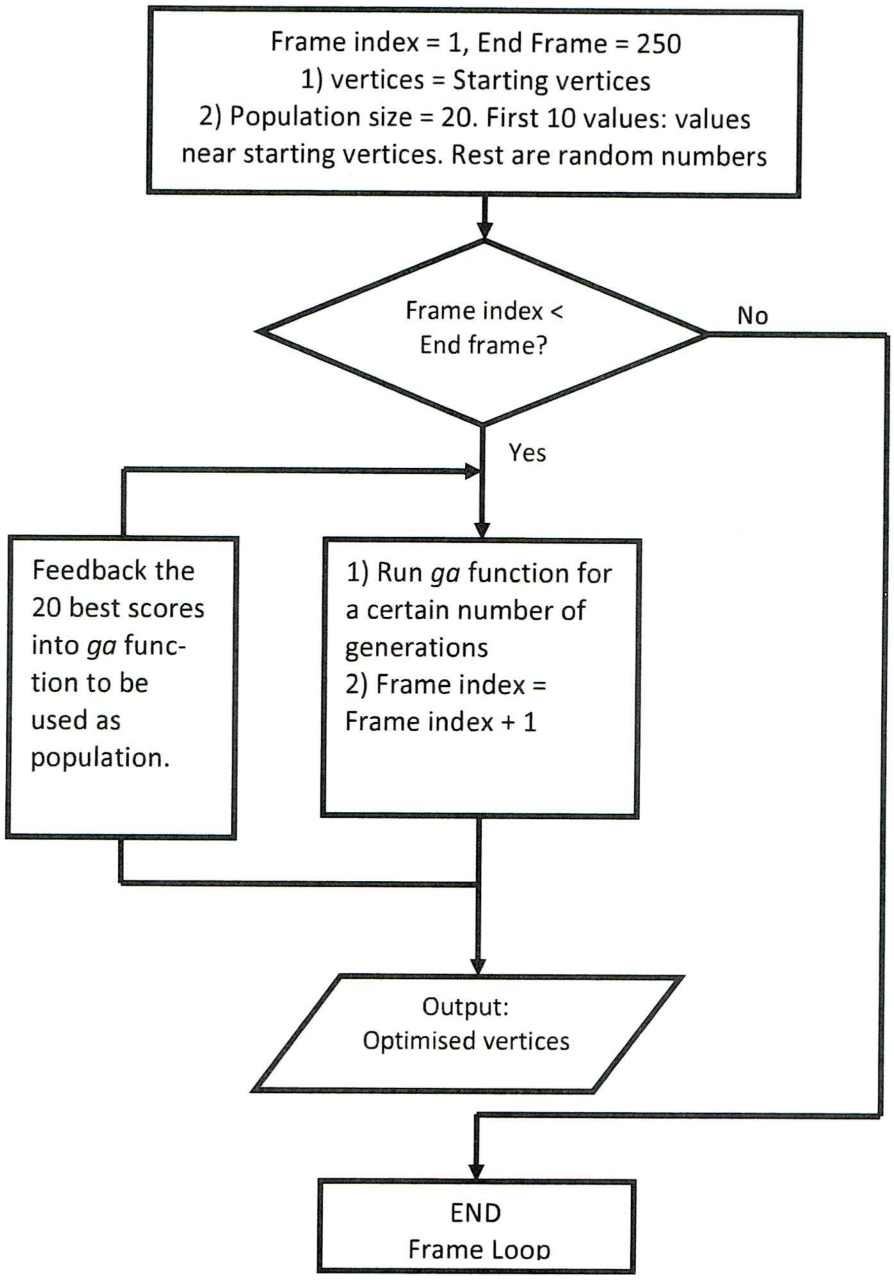

Frame index = 1, End Frame = 250
1) vertices = Starting vertices
2) Population size = 20. First 10 values: values near starting vertices. Rest are random numbers
Frame index < End frame?
No
Yes
Feedback the 20 best scores into *ga* function to be used as population.
1) Run *ga* function for a certain number of generations
2) Frame index = Frame index + 1
Output: Optimised vertices
END Frame Loop

Figure 4-11 : Adapting the GA optimiser for dynamic programming. Population information from a frame is fed back into optimisation step of next frame.

The effects of the population feedback to the following frame can be better appreciated in Figure 4-12 where the scores of the GA can be seen decreasing in almost a saw-tooth way in between the frames while at the same time the values of the objective function follow a decreasing trend. Both the dynamic and static optimisation occurring in this process can be clearly observed in this way. Sometimes a sharp increase in the objective function's value (e.g. between the vertical dotted lines in Figure 4-12) can be observed and this sudden increase usually means that there has been a change in the actual distribution of the data points such as when there is a new object entering the field of view or when there is a change in the lighting. This figure also shows how the GA is trying to bring down the cost function value whenever a new frame occurs.

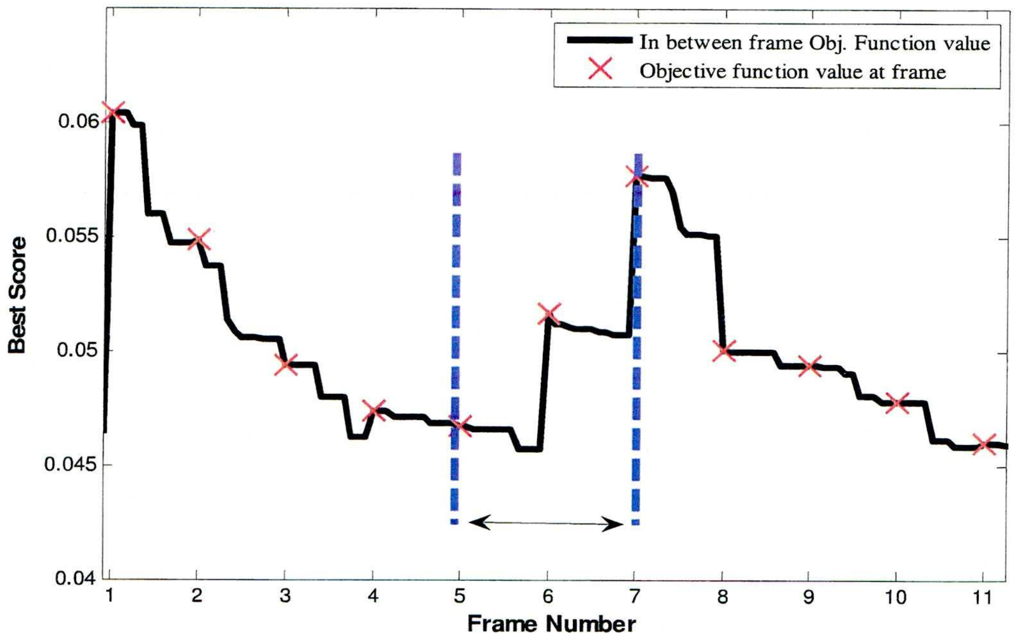


Figure 4-12 : Behaviour of Best Score values of the GA. Between the frames denoted by the crosses, static optimisation is taking place. Dynamic optimisation happens when the general trend in Best Score values is a decreasing one. A sharp increase within the dotted lines indicates a change in data points and location of vertices.

While the observations on the behaviour of the optimisers are based on visual checks, two similar and related experiments are designed to evaluate quantitatively the optimisers on dynamic data. The first one involves finding the number of objective function evaluations needed to achieve a desired level of accuracy. Data points are generated in such a way to simulate the dynamic nature of data points that occur in a real image sequence. In a real image sequence, the data points obtained from successive frames are not static even if there are no big changes in lighting or there is no new object entering and leaving the scene. This is due mainly to noise present in the sensor. To simulate this behaviour, data points that lie in a general triangular shape are first generated. These points are created by filling an ideal triangle with 500 points. This ideal triangle is represented by the vector $\vec{x}$ which contains the vertices. A random perturbation of magnitude ξ is then added to the starting vertices to give a new set of vertices $\vec{x}_{new}$ and a new data point distribution p. ξ is a uniformly distributed random number between $-\sigma$ and $+\sigma$, with $\sigma = 0.1$ initially. Different small random values are then added to each data point of p and this addition is repeated on ten copies of p to give p_j, where $j = 1,2,\ldots,10$. Each data point distribution in p_j is then 'displayed' successively during one second. This has the effect of simulating a data point distribution coming from an image sequence with a frame rate of 10 fps. At the end of this process, the value of σ is increased to 0.2 and the whole process repeated again for $\sigma = 0.1, 0.2, 0.3, 0.4$. This can be summarised by the equations 4-4 and 4-5,

$$\vec{x}_{new} = \vec{x} + \xi \qquad (4\text{-}4)$$

$$p_j = p + \xi_2 \quad \text{for } j = 1,2,\ldots,10. \qquad (4\text{-}5)$$

where ξ is a uniformly distributed random number between $-\sigma$ and $+\sigma$ for $\sigma = 0.1, 0.2, 0.3, 0.4$ and ξ_2 is a uniformly distributed random number between -0.1 and $+0.1$.

For each set of vertices $\vec{x}_{new}$, the three optimisers (GA, Random and Nelder-Mead) are used to obtain the best possible fit triangle with all of them using the same number (50) of cost function evaluations. The optimisers are only given starting vertices at the beginning of the simulation. The experiment is repeated 25 times and the average rms errors of the triangles' vertices from the known ideal vertices are calculated. The results are shown in Figure 4-13. All three optimisers give almost the same error values at low perturbation values with all the error values increasing in different proportions as the perturbation is increased. However the rate of increase for the GA is smaller than the other two. The difference in average error of the GA from the other two is also more considerable when the perturbation factor is at its highest value. Hence, given the same number of maximum allowed function evaluations, it is experimentally shown that the GA achieves much better error rates with increasing noise level.

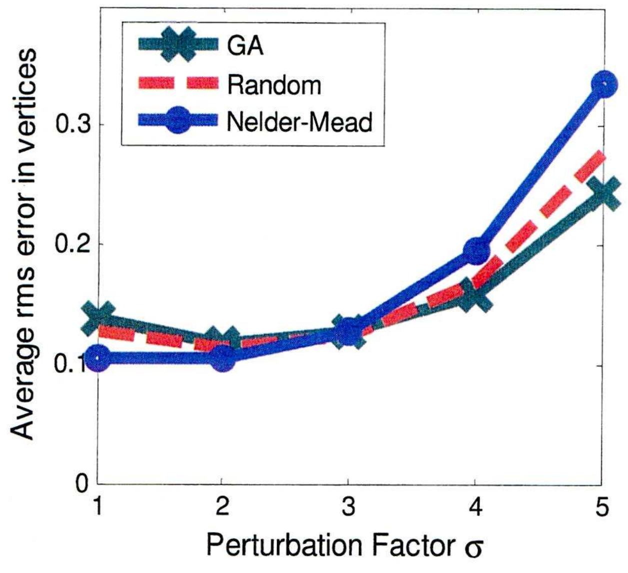


Figure 4-13 : Behaviour of the rms error of triangle vertices from ideal vertices for the three chosen optimisation techniques as the perturbation in the data points is increased.

The second experiment that is carried out has as intention to evaluate how fast an optimiser can react to a change in data point distribution before it reaches a certain predefined accuracy. This experiment is similar to the previous one apart from the fact that each optimiser is allowed any number of function evaluations until it reaches a certain low value of the cost function (0.5) and the time needed to reach that value is recorded. The experiment is run on an Intel Core2 Duo 2.13 GHz processor and it is repeated 25 times. The results obtained are shown in Figure 4-14. It can be seen that the random optimiser takes the longest time in almost all cases while the Nelder-Mead (NM) approach performs best at low values of σ but its ability to react to changes decreases when the starting points are not close to being ideal. It is also observed that at higher perturbation values, both the Random and the NM optimisers take more than 0.1 sec to achieve the set accuracy value, i.e. more than the amount of time that the frame is available in a 10 fps image sequence. This second experiment shows how the GA is able to obtain lower cost function values faster than the others especially when the data points change quickly. Although NM achieves similar or even better results than the GA at low perturbation, it is expected that the higher perturbation factors (noise levels) are closer to real-world imagery and hence this technique is dropped as a possible optimiser choice.

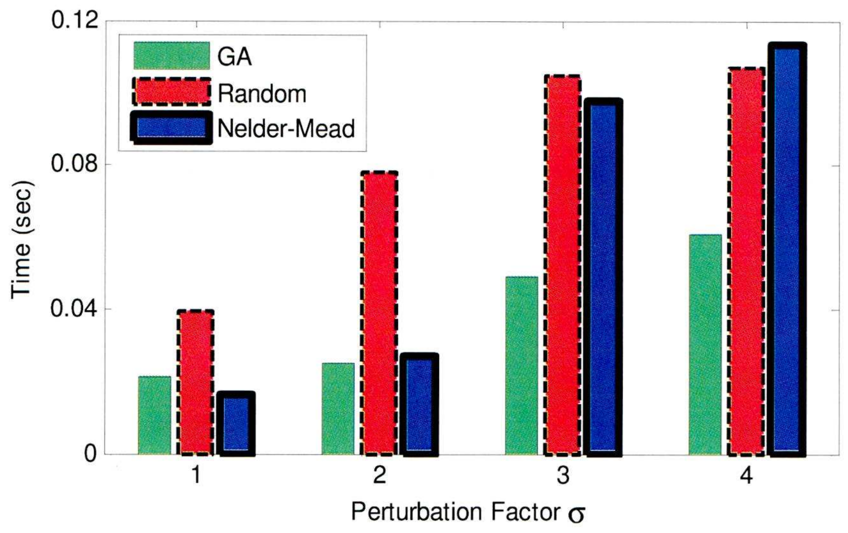


Figure 4-14 : Bar chart showing the time required to reach a predefined cost function value as perturbation is increased.

One could argue that the random optimiser is similar to NM for high perturbation values but it is not discarded yet as it is a method that is easily implemented and can offer a quick and computationally cheap alternative to the optimisers when data points are not too noisy. With the Nelder-Mead optimiser out of the running, the other two methods are then subjected to other tests in dynamic scenes to see how they performed. These tests all involve finding the 'centre of mass' of an object, obtained from weightings determined from spectral un-mixing as explained in section 3.5 (see Figure 3-19), and changing other parameters such as object size and image noise for each test. Figure 4-15 shows the average error when using the GA and Random optimisers in the un-mixing process as the diameter of object is increased for several paths. It can be observed that the position error generally decreases with the increasing size of the object. This is what one would expect as the number of data-points representing the object in the un-mixing triangle increases with increasing object size. The high position error value when the object is just one pixel wide shows that one data-point is not always enough to obtain a proper simplex for the un-mixing process. As the objects get bigger, the position errors tend to settle to a value of slightly less than a pixel, which is an accurate position estimate.

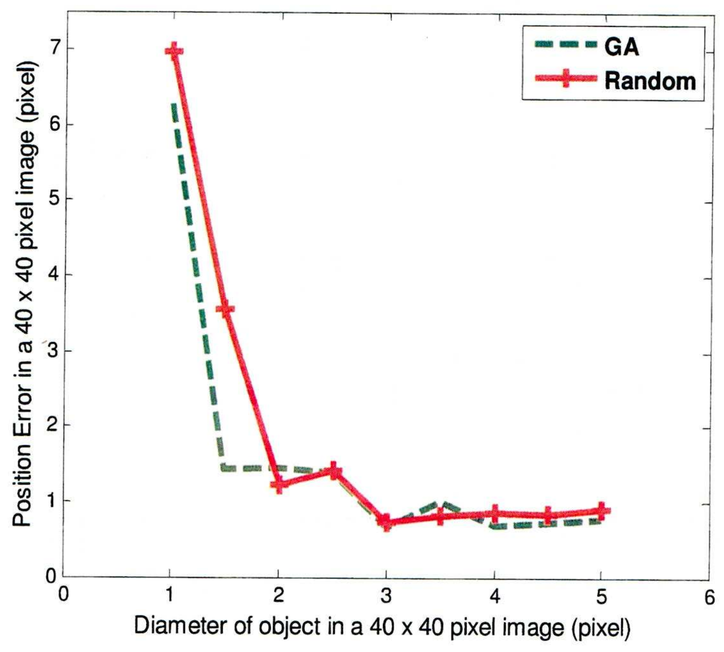


Figure 4-15 : Error in finding the centre of object as the size of object increases.

Another experiment is to find how the optimisers would react when subjected to noisy images. Images obtained from low resolution sensors usually contain electronic noise and a random image noise is a good way to model this type of noise. A slight improvement that can be brought to the model in the future is to add noise that increases with the signal level, resulting in the bright areas being noisier than the dark ones. This type of noise is common in CCD cameras whereby noise is a Poisson process and increases as $\sqrt{N}$, N being the number of detected photoelectrons on the CCD. For the purpose of this experiment, the variance of the random noise is increased to see how the optimisers deal with the constantly changing data distribution with increased number of outliers present. The results on several paths taken by an object moving in noisy environment are shown in Figure 4-16. The GA optimiser's error increases with increasing noise but it still managed to keep the position error to less than one pixel. The random optimiser on the other hand follows a quite unusual trend with a sharp increase in error at first followed by a decrease and then eventually increasing again. It was difficult to understand the reason behind this but after careful analysis it was deduced that the initial high peak (at noise variance = 1) is perfectly correct. The random optimiser finds it difficult to adapt to the noisy data around the three main regions of high data point concentrations. However as the noise increases, the data distribution tend to fill the gaps between the three regions of high concentrations and the optimiser is able to use these points to maintain a steady triangular wrap around the distribution. As the noise increases to higher variances, the position errors start to increase for both optimisers, which is a perfectly understandable situation, and one can predict even higher errors if the noise is increased further.

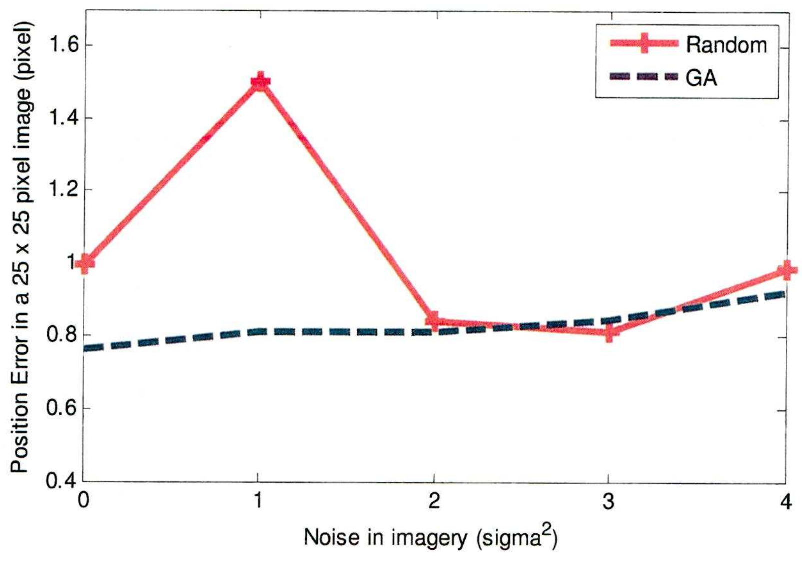


Figure 4-16 : Error in finding the centre of an object as image noise increases

Although both the Random and the GA optimisers are performing well, the random optimiser has the disadvantage of not being able to adapt well to quickly changing point distributions or rapid changes in pixel intensity. The values of the weighting w and the bias term b_k (see Figure 4-8) are also seen to play a bigger role in the random optimiser than the GA when faced with highly noisy data containing many outliers. Having to adapt and fine tune the random optimiser for each type of noisy video input means that it cannot perform in a generic way on any data and hence its use is compromised. The GA method is more stable to increasingly noisy images and is thus chosen to be the optimiser in the un-mixing process for real videos in dynamic environments. With the position of the centre of object estimated in this way, a Kalman filter can then be used to generate a continuous track for the object as will be shown in Chapter 5.

4.3 Sub-pixel sized object detection

The un-mixing method used to detect small objects of a few pixels wide cannot be applied to objects that are smaller than a pixel. This is shown in Figure 4-15 where the position error is high for one-pixel sized objects. As one of the aims of this project is to develop a suite of detection methods to measure small objects whose sizes range from a few pixels wide down to minuscule objects that are even less than a pixel, another method had to be devised to cover detection of the sub-pixel sized objects. There are several ways to track sub-pixel sized objects. In biomedical microscopy, it is very common to track the motion of very small particles using algorithms such as cross-correlation, centroid, and direct Gaussian fit [56]. In correlation-based processing of velocimetry data, the Gaussian character of the spots gives rise to a Gaussian correlation profile that is used to determine sub-pixel displacement [107]. In this project, the modulation of intensity at pixel boundaries is used as a way to detect and measure motion for sub-pixel sized objects. An object of size smaller than a pixel modulates the intensity profile of a pixel in a certain way when it passes through the particular pixel. These changes in intensity can be a valuable source of information to enable the detection of sub-pixel displacement, and hence motion. As shown in Figure 4-17, even when the square translates less than a pixel, its motion modulates pixel intensities.

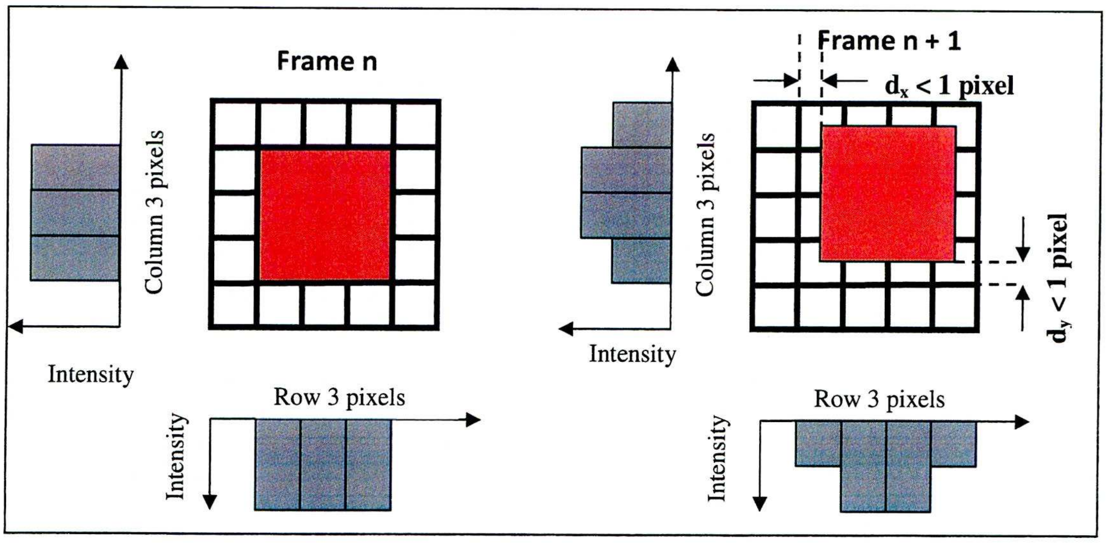


Figure 4-17: Sub-pixel displacement. Even when square object translates by less than a full pixel, there is a change in intensity level of the pixels it is entering and leaving

By using the time derivatives of the intensity at each pixel, and the spatial derivatives of intensity obtained by comparing the intensity of neighbouring pixels, the displacement and even velocity can be estimated as is explained next. Figure 4-18 shows an example of what one would expect to see in the intensity distribution of a pixel if a square object of less than one pixel in size passes through it. Assuming the object is moving at constant velocity and the size one pixel represents in real world is known, the speed of the object and even its size can be estimated. Time t_3 is the time taken for the leading edge of an object to cover a distance X_{pix} equal to the size of the pixel and t_1 is the transit time across the boundary of a pixel. The speed V_{obj} of the object is obtained by dividing the distance X_{pix} by t_3 while the transit time t_1 information is used in determining the object size X_{obj} as explained in equations 4-6 and 4-7 below [9].

$$V_{obj} = \frac{X_{pix}}{t_3} \qquad (4\text{-}6)$$

$$X_{obj} = \left(\frac{t_1}{t_3}\right) \times X_{pix} \qquad (4\text{-}7)$$

It has to be noted that the graph in Figure 4-18 shows the absolute value of the change in intensity values and not how the values of intensity themselves change, i.e. even if a red moving object is less intense than a predominantly red background, the change will always be a positive change of the similar shape as below. The only assumption that is made is that the object is not starting from a background of exactly equal colour values.

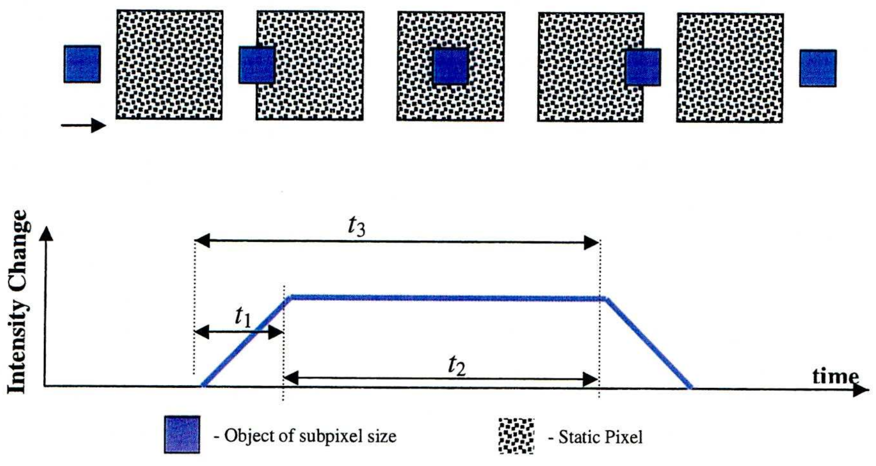


Figure 4-18: Model for finding speed of motion and size of object [9]

With this model in mind an experiment is devised to see how the intensity at a pixel behaves as objects of different sub-pixel sizes passes through it. A simulation of a square object moving horizontally on a black background is created and the intensity change at one particular pixel is analysed as the object goes through it. The size of the square is changed at each run and it ranges from 0.2 pixel to 1 pixel in length. The pixel intensity change is shown in Figure 4-19. The implementation code is available in Appendix A2.

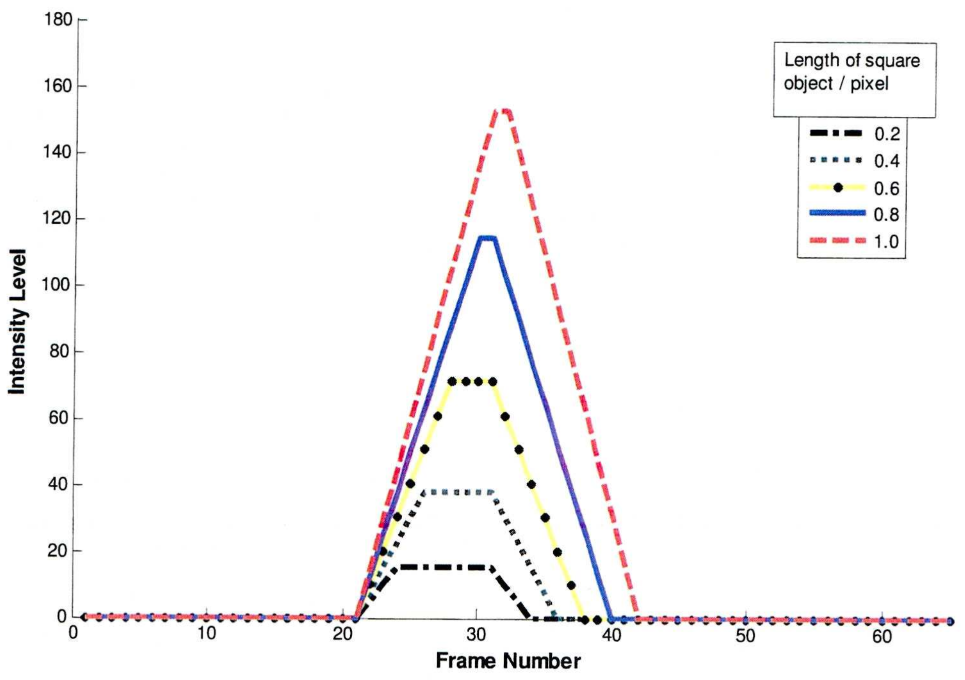
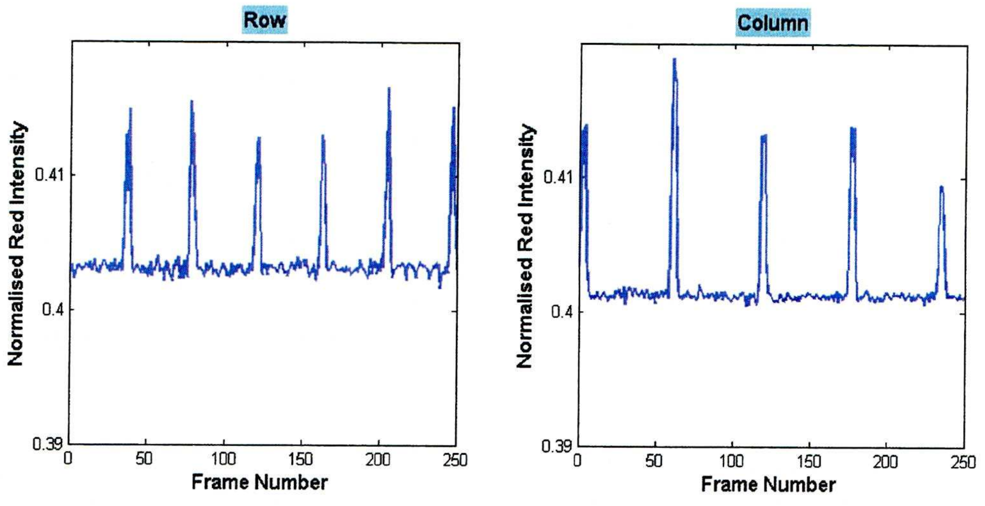


Figure 4-19 : Intensity change in a pixel as a square object of increasing sub-pixel sizes goes through it.

The behaviour of the pixel intensity as a sub-pixel sized object goes through it is found to be similar to the model explained in Figure 4-18 and it can hence be deduced that it is also be possible to estimate the speed and size of the object using equations 4-6 and 4-7 respectively. The next stage of the experiment is to devise a way of detecting automatically when the edge of an object enters and leaves a pixel for all the pixels present in a particular image frame to be able to determine the object's position. While Figure 4-18 and Figure 4-19 show ideal models of intensity change at a pixel, in real life the intensity profile is more likely to be corrupted with sensor noise and other imperfections such as change in lighting. In the noisy artificial imagery that is generated according to the method described in section 4.2.4 containing a moving sub-pixel sized object, a random row and column of pixels give profiles in the Normalised Red channel like those in Figure 4-20. One of the tasks here is to filter the intensity values to obtain the 'zero-crossings' as these points give information about when the leading edge and trailing edge of an object enters and leaves a pixel respectively.



(a) (b)

Figure 4-20 : Intensity change in a random row and column of pixels.

Filtering can be achieved by:

1) Finding the average and variance of the first five values.

2) Subtracting the average from the next values to remove the constant offset.

3) Using the variance to set a threshold value to detect the zero-crossings.

In some cases such as in Figure 4-20(b) where the object of interest starts its motion at the pixel being investigated, the average and standard deviation obtained for the initial five frames are not useful in removing the constant offset. To get round this problem, one can use the fact that the variance or standard deviation is very large whenever the intensity values are showing characteristics that they are peaking or have just had their peak value. If the standard deviation is too high, the average and standard deviation of the next five frames is then calculated and so forth until the standard deviation get to a small value. The filtering algorithm is explained in the flowchart Figure 4-21 and implementation code with offline processing of a dataset can be found in the appended CD.

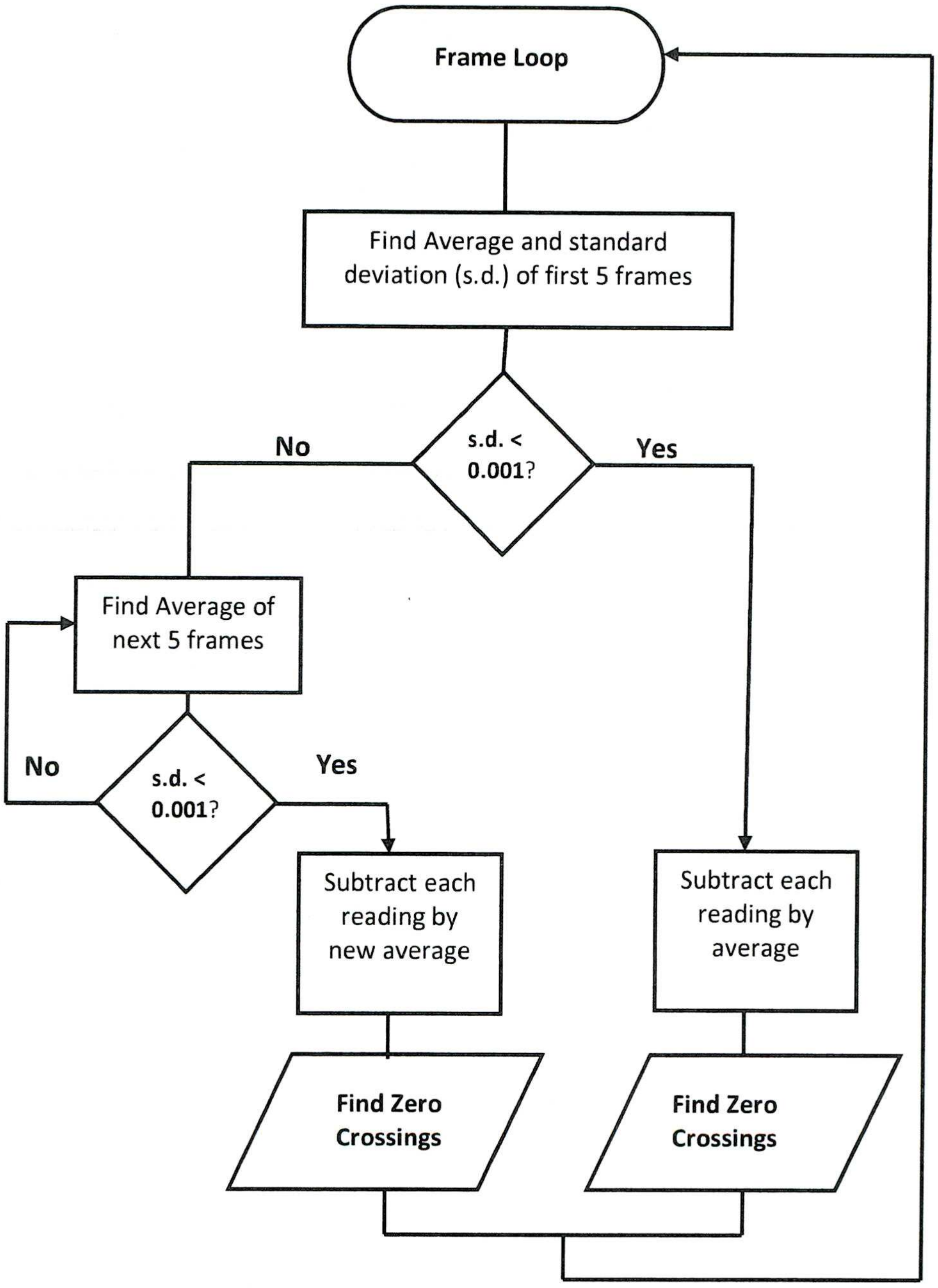

Frame Loop
Find Average and standard deviation (s.d.) of first 5 frames
s.d. < 0.001?
No
Yes
Find Average of next 5 frames
s.d. < 0.001?
No
Yes
Subtract each reading by new average
Subtract each reading by average
Find Zero Crossings
Find Zero Crossings

Figure 4-21 : Flowchart for filtering algorithm

An example result is shown in

Figure 4-22 and Figure 4-23. Figure 4-23 is an example of when the average and standard deviation obtained for the initial five frames are not useful in removing the constant offset in the intensity distribution.

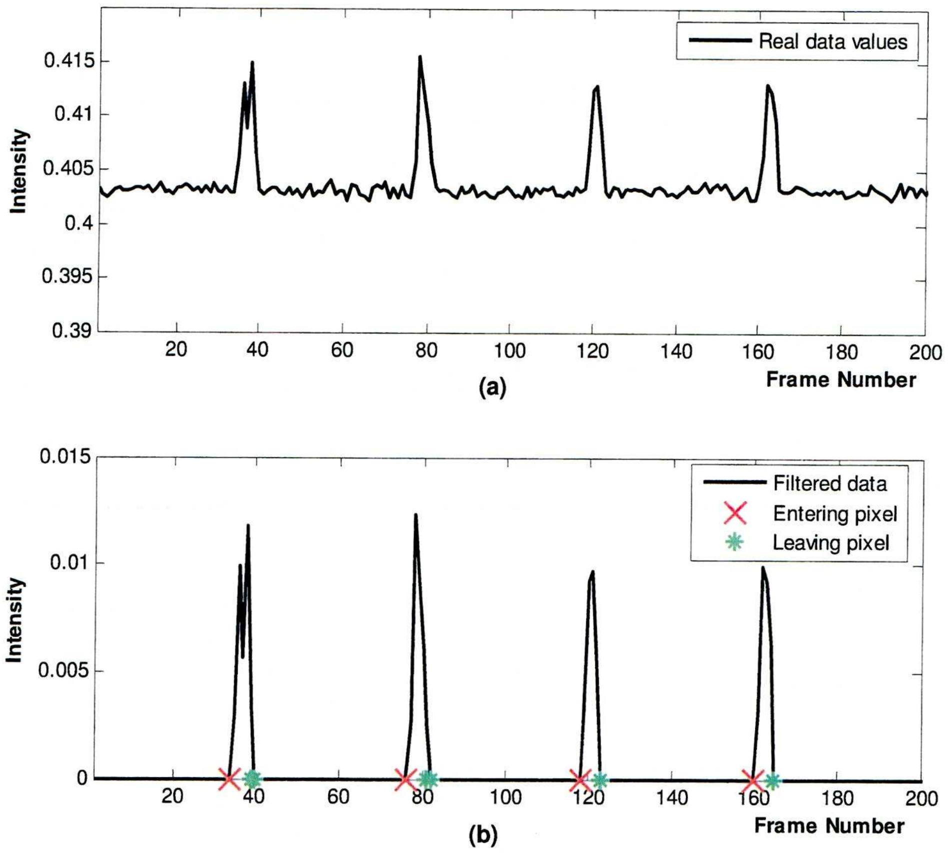


Figure 4-22 : (a) shows the intensity values for a particular pixel of an image as time passes (b) shows the filtered values with indications when the object is entering and leaving the pixel.

Another related use of these intensity profiles is shown in Figure 4-24. In this figure, an object is moving vertically down on an image plane. The object is entering pixel 4 at frame n+1 and entering pixel 5 at frame n+7. The highlighted frames show the time taken t_p for the leading edge of the object to travel across a pixel, and hence the speed of the object can also be estimated from this technique if the distance that this pixel represents in the real world is known. With the location of the sub-pixel sized objects obtained in this way, the next step is then to use predictive filters to track the leading and trailing edges of the object and hence generate a continuous track for the object.

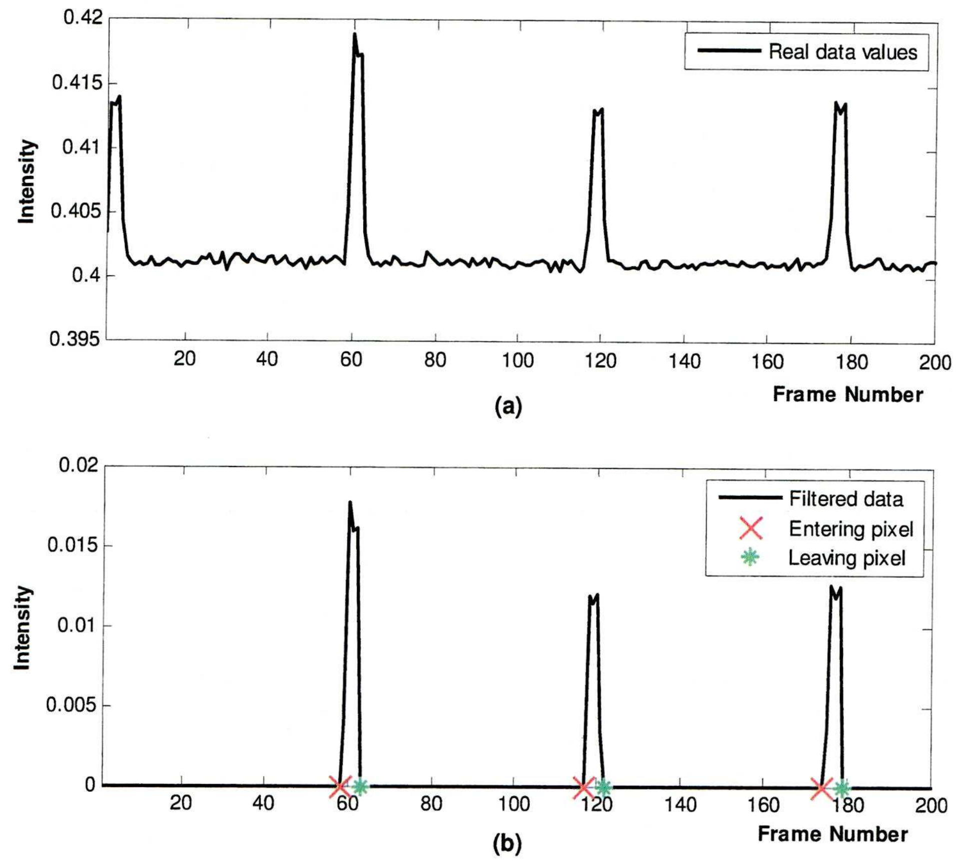
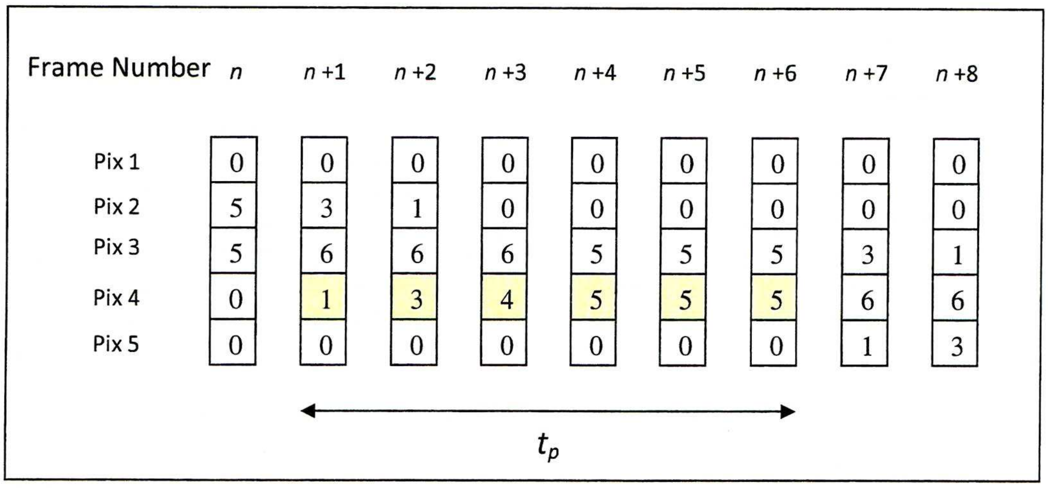


Figure 4-23 : (a) shows the intensity values for a particular pixel of an image as time passes (b) shows the filtered values with indications when the object is entering and leaving the pixel.



Frame Number	n	$n+1$	$n+2$	$n+3$	$n+4$	$n+5$	$n+6$	$n+7$	$n+8$
Pix 1	0	0	0	0	0	0	0	0	0
Pix 2	5	3	1	0	0	0	0	0	0
Pix 3	5	6	6	6	5	5	5	3	1
Pix 4	0	1	3	4	5	5	5	6	6
Pix 5	0	0	0	0	0	0	0	1	3

t_p

Figure 4-24 : Time taken for one edge of object to cross pixel

4.4 Human skin detection

As the objects to be monitored are likely to be human beings, a component that is present in every human being is tested to become an invariant in the detection process. This component is the human skin, particularly its chromatic properties. It is claimed that under certain light conditions and depending on the colour spaces used, skin chrominance varies little between different skin types [111]. This invariance of the skin chrominance can be ideal to assist in the detection of people. In particular, faces and hands tend to remain uncovered in most situations and are therefore good candidates for monitoring human activity. Skin detection already plays an important role in a wide range of image processing applications ranging from face detection, face tracking, gesture analysis, and content-based image retrieval systems. Skin detection methodologies based on the chromaticity information of the skin as a cue is gaining even more attention as it provides computationally effective yet, robust information against rotations, scaling and partial occlusions [108]. Skin detection using colour information can however be a challenging task as the skin appearance in images is affected by various factors such as:

- *Illumination*: A change in the light source distribution and in the illumination level (indoor, outdoor, highlights, shadows, non-white lights) produces a change in the colour of the skin in the image (colour constancy problem). The illumination variation is the most important problem among current skin detection systems that seriously degrades the performance.
- *Camera characteristics*: Even under the same illumination, the skin-colour distribution for the same person differs from one camera to another depending on the camera sensor characteristics. The colour reproduced by a CCD camera is dependent on the spectral reflectance, the prevailing illumination conditions and the camera sensor sensitivities.
- *Ethnicity*: Skin colour also varies from person to person belonging to different ethnic groups.
- *Other factors*: Different factors such as subject appearances (makeup, hairstyle and glasses), background, shadows and motion also influence skin appearance.

Many of the problems encountered in the visible spectrum can be overcome by using non-visual spectrum such as infrared (IR) and spectral imaging as skin colour properties tend to be more robust in these domains [109]. However, it is an expensive process and its use is limited to specific application areas such as biomedical applications. Numerous techniques are available in literature for skin detection using colour in the visible spectrum. In this section of the thesis, a brief review of the various skin modelling and classification strategies based on colour information in the visual spectrum is presented. The review is divided into two main categories. Firstly, the various colour spaces used for skin modelling and detection are evaluated together with an explanation of the different skin modelling and classification approaches. The primary steps for skin detection are (1) to represent the image pixels in a suitable colour space, (2) to model the skin and non-skin pixels using a suitable distribution and (3) to classify the modelled distributions. Secondly, a few approaches that use skin-colour constancy and dynamic adaptation techniques to improve the skin detection performance in dynamically changing illumination and environmental conditions are introduced. The section is concluded by evaluating images of low resolution containing skin samples and testing the suitability of the colour spaces for the un-mixing procedure.

4.4.1 Skin modelling and characterisation

The main problem in using skin tones in image processing is in separating (or segmenting) the skin regions from the other background regions automatically and reliably. Human observers have little difficulty in segmenting an image into regions defined by colour. Automatic segmentation proves to be somewhat more difficult. The perception of colour by humans is a psychological experience as much as it is a physical phenomenon and hence makes the segmentation of a video image into skin-coloured regions and background more complex than a straightforward matching of wavelengths. While the skin colour of a single subject may appear to an observer as being very consistent across an image, there is likely to be wide variation in the wavelengths representing the colour. To deal with this variation, a human skin colour model has to be derived to decide whether a pixel contains skin or not based on a decision rule.

From a classification point of view, this process can be viewed as a two-class problem: skin-pixel vs. non-skin pixel classification [108]. Several techniques on skin-colour model classification, ranging from simple look-up table approaches to complex pattern recognition approaches have been published [110].

In this thesis, a *pixel-based* skin detection method, i.e. one that classifies each pixel as skin or non-skin individually independently from its neighbours as opposed to region-based methods, is used to achieve this skin modelling task. The procedure executed is as follows. Firstly, a large database of photos containing human beings from a wide spectrum of ethnic origins is collected. Care is taken to collect photos which are taken under different lighting conditions and which contain various parts of the human body skin, such as face and arms as shown in Figure 4-25.

Figure 4-25 : Sample of photos containing skin of human beings from various ethnic origins and taken under different lighting conditions.

Secondly, a tedious process of cropping and selecting only the sections of the images containing human skin is initiated. All the skin colour samples are then stored in database *Skin_only* for further processing.

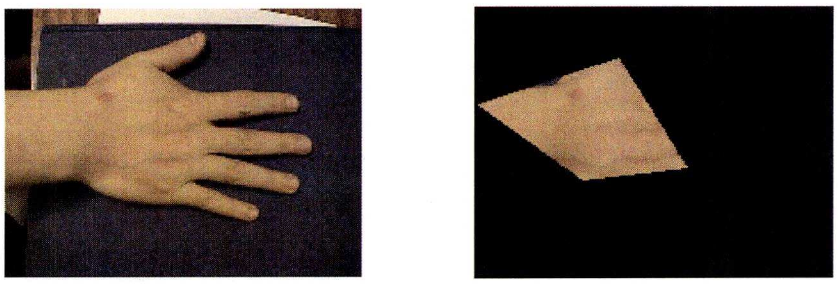


Figure 4-26 : Cropping images to obtain only skin-containing images.

Thirdly, for each image *I* contained in the database *Skin_only*, every pixel's p_i chrominance value is plotted on a 2-D graph with axes dependent on the colour spaces being used as shown in Figure 4-27. This technique is similar to the histogram based approach used by Soriano et al. in their attempt to find a skin locus to characterise skin appearance [111]. The colour space (usually, the chrominance plane only) is first quantised into a number of bins, each corresponding to a particular range of colour component value pairs (in 2D case) or triads (in 3D case). These bins, forming a 2D or 3D histogram, are referred to as the 'lookup table' (LUT). Each bin stores the number of times this particular colour occurs in the training skin images. The final step that remains is to convolve the distribution with a Gaussian to obtain a cloud of points for the skin chrominance model.

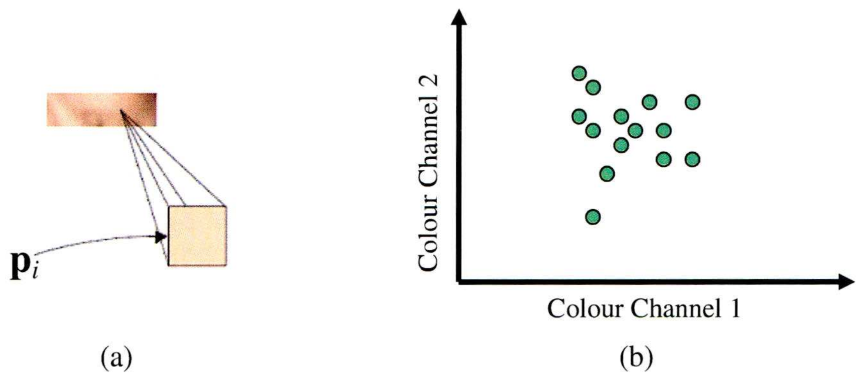


Figure 4-27 : Collecting skin chromaticity values for each pixel present in the image sample containing skin only for a hypothetical colour space

The colour spaces that are used in the experiment to test for consistency in skin colour chrominance are (i) Normalised RGB (*NRGB*), (ii) HSV, (iii) L*a*b* and (iv) YCbCr. For the *NRGB* colour space, only two of the three normalised variables are needed to specify any colour within the range allowed by the primaries as previously explained in section 3.3.1. Only normalised red and normalised green channels are used in this experiment and the results obtained has values that tended to agree with Wang and Yuan [112]. Given an image of size *x* by *y*, the following colour ranges can be used to represent skin chromaticity:

$$O_{(x,y)} = \begin{cases} 1 & \text{if } NRed_{(x,y)} \in [0.35, 0.47] \text{ and } NGreen_{(x,y)} \in [0.28, 0.40] \\ 0 & \text{otherwise} \end{cases} \tag{4-8}$$

As for the Hue-Saturation-Value (HSV) colour space, the representation that most closely resembles the way human beings perceive colours, Sobottka and Pitas derived hue and saturation ranges of 0° to 50° and 0.23 to 0.68 respectively to characterise skin colour [113]. These values are found to agree with the values obtained in the current experiment. However one more condition is added. This new condition is that pixels with Value < 0.1 are discarded because under this threshold the information contained in the Hue is not stable [114]. Given an image of size x by y, the following equation can then be used to filter out the skin:

$$O_{(x,y)} = \begin{cases} 1 & \text{if } H_{(x,y)} \in [0^\circ, 50^\circ] \text{ and } S_{(x,y)} \in [0.23, 0.68] \text{ and } V_{(x,y)} \geq 0.1 \\ 0 & \text{otherwise} \end{cases} \tag{4-9}$$

For the CIE $L^*a^*b^*$ space, the ab chrominance values are used to construct the model. The luminance component L* of the colour representation cannot be a reliable measure for detecting facial regions as the reflected light intensity tends to vary considerably across a human face. The skin samples were found to lie within the ranges of –5 to 35 for a^* and –7 to 40 for b^*. These values are not exactly similar (–10 to 40 for a^* and –10 to 60 for b^*) to those published in literature by Cai and Goshaby [115]. However one has to be aware that the technique employed by Cai is different from the one presented here. Rather than just classifying pixels to skin and non-skin regions, Cai assigns a weight to each pixel, showing the likelihood of the pixel belonging to the skin. The weights are obtained from a chroma chart that is prepared through a training process. For the sake of continuity with the other colour spaces, it is decided to use the results obtained in this project's experiment in the skin filter equation below where O is the output image:

$$O_{(x,y)} = \begin{cases} 1 & \text{if } a^*_{(x,y)} \in [-5, 35] \text{ and } b^*_{(x,y)} \in [-7, 40] \\ 0 & \text{otherwise} \end{cases} \tag{4-10}$$

The final colour space to be tested is the YCbCr. This colour space, like the previous two, allows for an effective use of the chrominance information (Cb and Cr) for modelling human skin. It is also very convenient in the sense that this format is typically used in video coding, and therefore the use of the same, instead of another, format for segmentation can avoid extra computation required in conversion as proposed by Chai and Ngan [116]. The following ranges are found to be the most appropriate to represent skin colours:

$$O_{(x,y)} = \begin{cases} 1 & \text{if } Cb_{(x,y)} \in [77,127] \text{ and } Cr_{(x,y)} \in [133,173] \\ 0 & \text{otherwise} \end{cases} \tag{4-11}$$

The distributions obtained from Figure 4-26's cropped image of a hand is shown below for all the evaluated colour spaces.

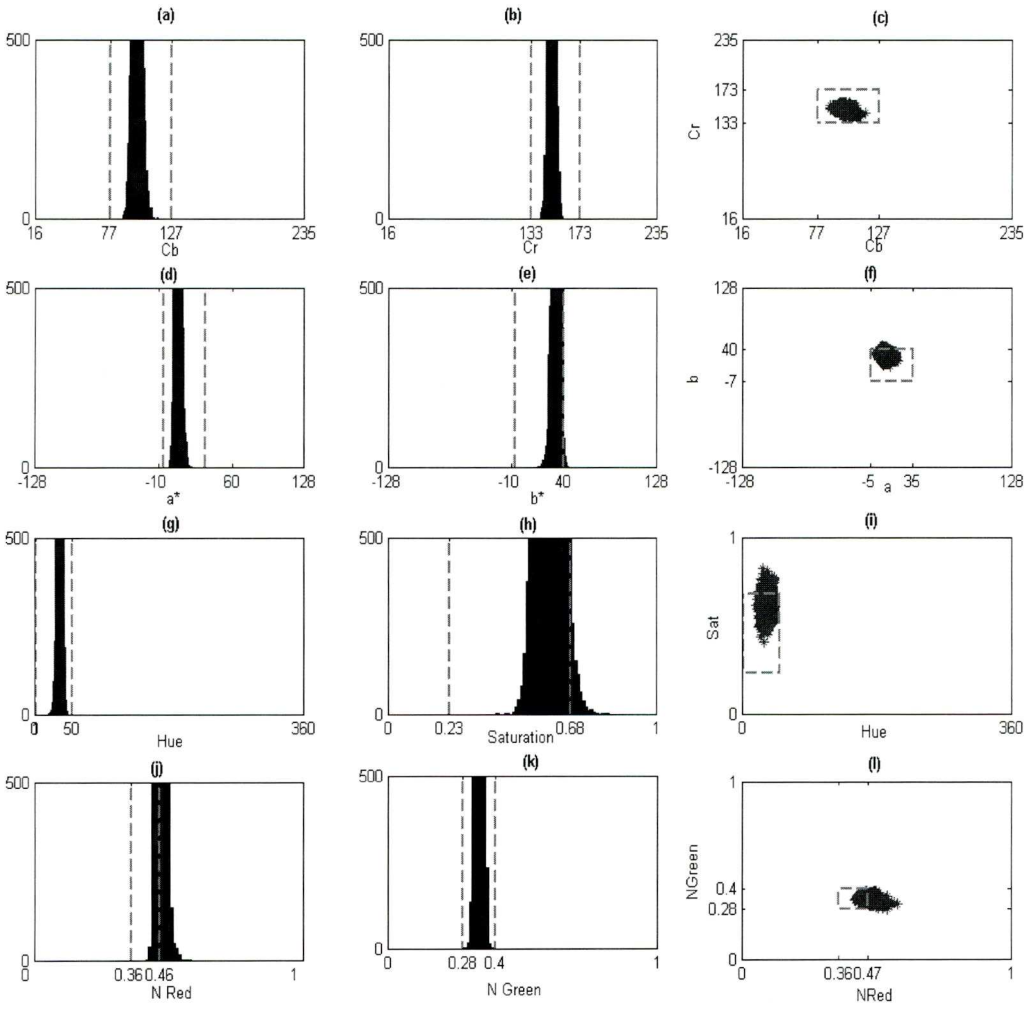


Figure 4-28 : Skin distributions for YCbCr (a) – (c), CIE L*a*b* (d) – (f), HSV (g) – (i), and Normalised RGB (j) - (l). First two columns are histogram distributions.

The ranges of values obtained for the various colour spaces are then tested on random high resolution images containing human skin. An example is shown in Figure 4-29.

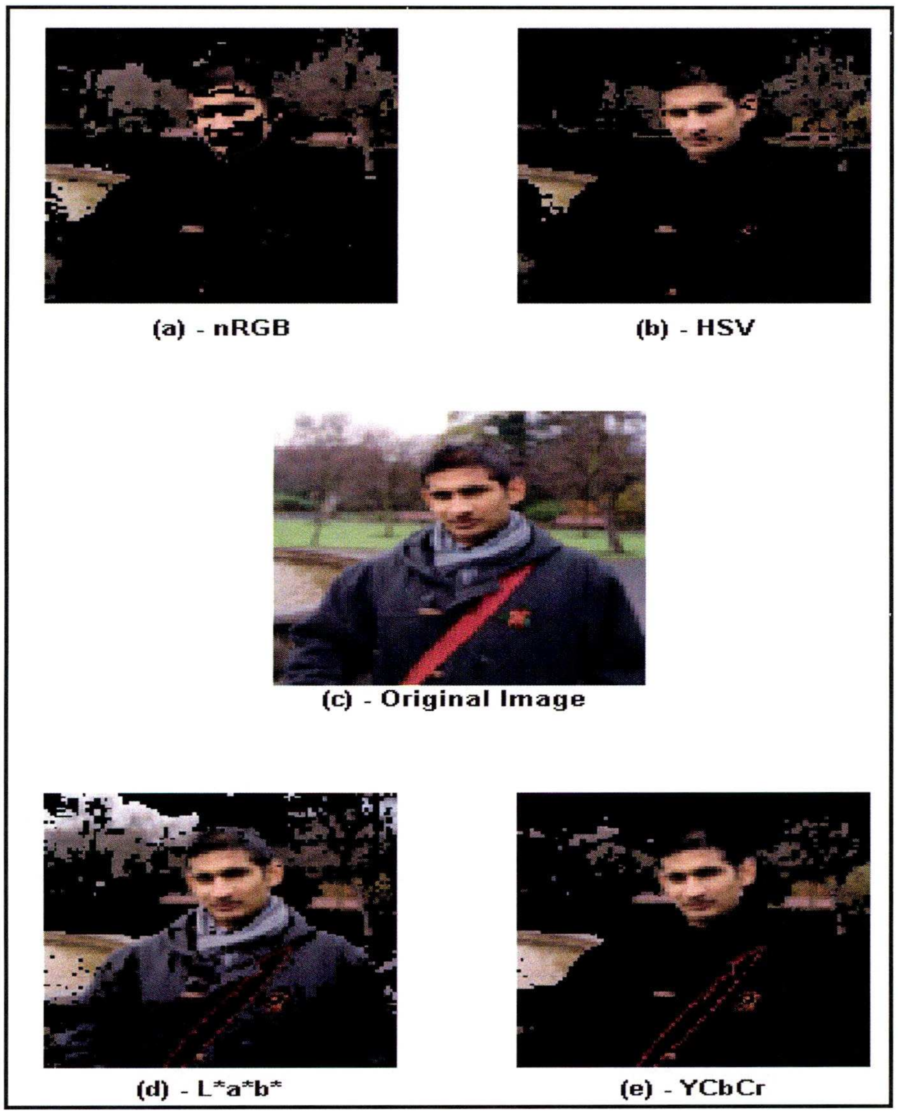


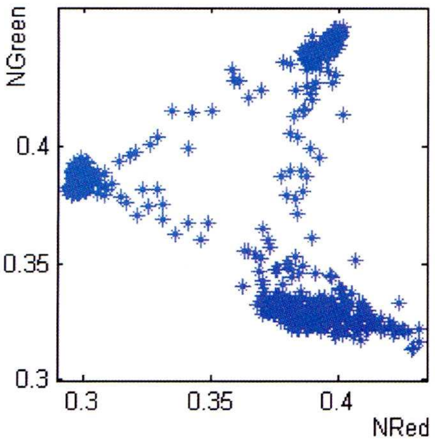
Figure 4-29 : Skin filter using all the four spaces separately applied to an image.

It can be seen that each colour space has its share of false positives and negatives depending on the image being used. The filtering process also does not perform well when the input images have 'bright spots' on the subject's face due to reflection of intense lighting or dark shadows on the face as a result of the use of strong directional lighting that has partially blackened the facial region. Even under the same lighting conditions, background colours and shadows can also influence skin-colour appearance.

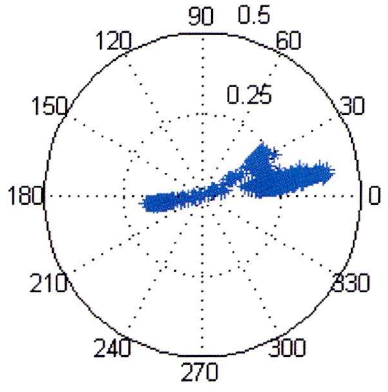
Furthermore, if a person moves, the apparent skin colours change as the person's position relative to the camera or light change. The human visual system can dynamically adapt to the varying lighting conditions and can approximately preserve the actual colour of the object. However, image capturing devices with limited signal processing are not capable of adapting to the rapidly varying illuminations across scenes. And finally, it is observed that very dark and light skins are not detected properly.

4.5 Choice of a colour space for the un-mixing procedure involving low resolution images

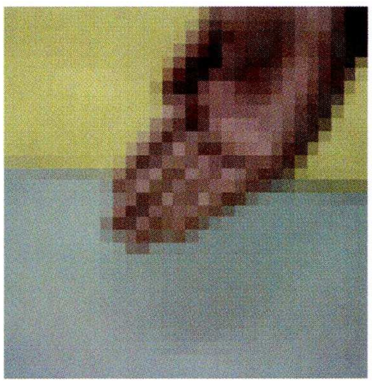
While all these techniques mentioned can help in further enhancing the skin detection process, one has to realise that their main targets are human faces in high resolution images. In this thesis, the images involved are of low spatial resolution with only parts of the human skin visible depending on the camera orientation and the clothing worn by the people being monitored. A second very important requirement to keep in mind is that the colour space to be chosen has to be compatible with the un-mixing procedure described in section 3.5. Figure 4-30 shows the distributions that are obtained for each colour space when a low resolution image containing three main colours was analysed. *NRGB, CIE L**a*b** and YCbCr represented colour values in a Euclidean way and hence very appropriate for spectral un-mixing. The HSV colour space has a circular representation of colour and hence it is not suitable for the simplex wrapping process and the accompanying linear mixing model (LMM). The polar plot of Hue v/s Saturation in Figure 4-30(b) will make it very difficult to apply the linear un-mixing model directly to the data and as a result it is decided to discard this colour space from further use. However, one must be aware of the powerful characteristics of the HSV space such as its invariability to changing light conditions and hence a single channel, e.g. Hue only, could be used in a detection process involving colour cues only. As for the three colour spaces which display un-mixing friendly properties, none of them is an untouchable candidate. It is decided to use the Normalised RGB in this thesis during the experiments on the real videos (shown in Chapter 5) because this colour space is almost readily available from the recording camera. However the other two colour spaces can also be used.



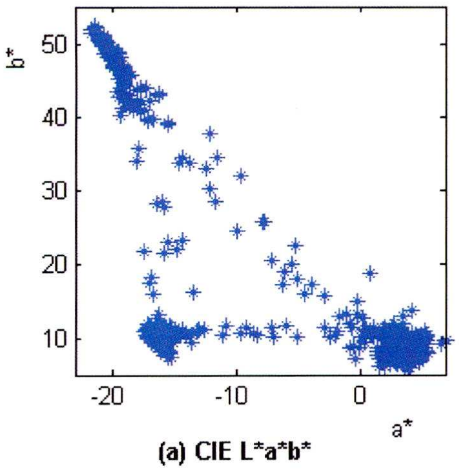
(a) NRGB



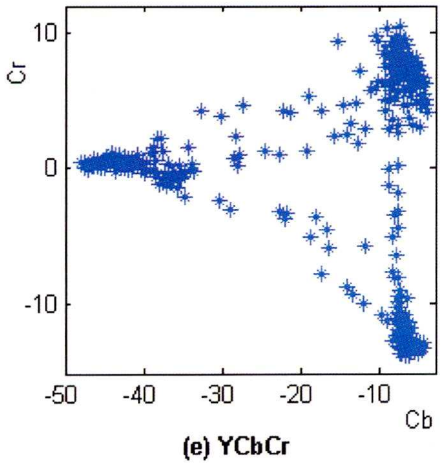
(b) Hue v/s Saturation



(c) Low Resolution Image



(a) CIE L*a*b*



(e) YCbCr

Figure 4-30 : Data points distribution for a low resolution image containing three main colours shown for each colour space

4.6 Summary

This chapter consists of a systematic consideration of the techniques to detect very small objects that are used in this thesis. When objects are a few pixels wide, they can be detected by using the Linear Mixing Model and the accompanying un-mixing process. This model, which assumes that each pixel in a given image contains a proportion of one or more definite colours (or spectra), can be used to decompose each mixed pixel into a linear combination of the individual colours. Once a pixel is decomposed, depending on its location in the un-mixing triangle of an image data values, it can be assigned to a particular object's colour by using a weighting function to connect the pixels that contain proportions of that object. The un-mixing triangle is obtained by using an optimisation function based on the vertices to enclose the data points as tightly as possible. A multi-objective function is derived to achieve this. This multi-objective function uses a weighted sum approach to balance the two objectives present. The values for the weight used during the tests are in the magnitude region of 0.001 to 0.01 but this range cannot be pre-determined unfortunately. It depends on the noise present in the data as outliers need to be taken into consideration. Although this is a disadvantage and other more complicated techniques exist to solve with multi-objective functions in a more generic way, it is not considered to be a major setback as it is found that the weighting value used do not have to be changed so frequently. The weighting value needs to be specified at the beginning of a video sequence depending on the quality of the image available but then it can be left to stay constant as the triangle optimisation process occurs and hence no extra computation is required. The genetic algorithm (GA) optimisation technique is chosen as the optimiser to be used for real video sequences because of its robustness to quickly changing and/or noisy data points. Two other detection techniques are also described. The first one is used to detect objects that are less than a pixel in size and it uses information about pixel intensity profiles to achieve this. The second one is a human skin detector that uses the common characteristics that exist in the skins of people of different ethnic origins to achieve detection. While these two techniques are not the main approaches to detection used in this thesis, they are still considered as they could enhance the main detection technique which is the un-mixing technique. More experimental results are given in Chapter 5.

CHAPTER 5

5. Data collection, experiments and results

5.1 Introduction

This chapter contains all the experiments on real videos that are undertaken to test the various algorithms and ideas proposed in the previous chapters. As mentioned before, this project is most likely to be deployed within a domiciliary care environment. With this in mind, an experimental scenario is devised to collect information about this particular type of environment and how human beings behave in them. Privacy is very important in these types of environments and the uneasiness felt by people being watched may be reduced by ensuring that the image quality is very poor and by not allowing the image data to leave the home at all. For the video testing part, while figures are given with detailed captions, it is often difficult to view the results on paper. Video clips of all the experiments have been stored on the CD available in the appendix for an easier understanding and evaluation of the results. It has to be noted that the frame rate in all the low resolution test videos is at a constant value of 10 fps.

5.2 Data Collection – Experimental Plan

During the data gathering exercise, a high resolution camera is used to record the various videos. The video camera recorder used is a Sony Digital Handycam® Model No. DCR-PC9E and the videos are recorded on high resolution Digital Video (DV) tapes. Each component of the experimental scenario lasts five minutes. Images taken from such type of cameras can vary greatly depending on the settings of the camera such as exposure, brightness, contrast, auto-zoom etc. All the internal camera image processing features that exist in the camera are turned off because in practice, none of these features are available when low resolution sensors are used.

5.2.1 Camera Location

The domestic environment where this project is most likely to be relevant in is homes for elderly people. The rooms in these homes are often cluttered with furniture such as wardrobes, tables and sofas and hence the position and orientation of the camera have to be chosen so that there is minimum occlusion of the person(s) being monitored. Wall-mounted cameras do not satisfy these requirements and a better alternative is to have a wide-angle, central ceiling mounted camera to track one or more persons in a certain area (model in Figure 5-1). This ensures that the maximum area is covered by the field of view of the camera and it also decreases the camera intrusiveness factor since the camera can be hidden behind a glass case or masked to look like a light source.

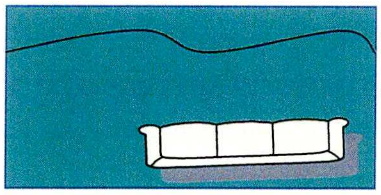


Figure 5-1: Central ceiling mounted camera view.

One drawback of a centrally mounted ceiling camera is that, apart from the head, some very important information such as the colour of the clothing worn and some articulated body movements will not be always visible. Therefore a more appropriate location is in a corner of the ceiling or just above strategic entry/exit places such as doorways as shown in Figure 5-2. These places are described as entry zones and are regions of high activity [117]. Inactivity zones such as sofas and chairs are called so because they involve little motion of the person. Typical use of a room involves entering through entry zones followed by visits to one or more inactivity zones and finally exiting the room. Using a sensor with a wide field of view enables the whole floor surface to be within the line of sight of the camera.

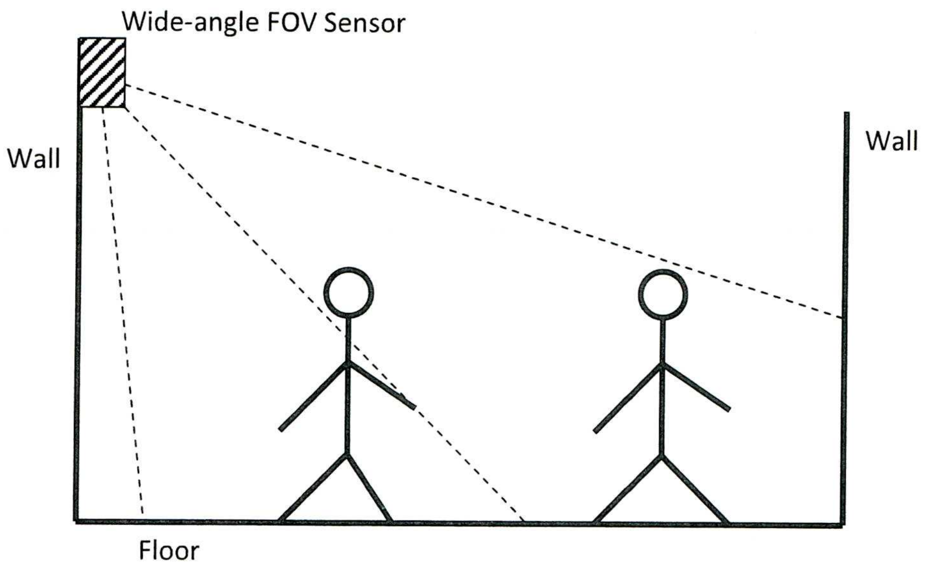


Figure 5-2 : Side view of a sensor arrangement placed at a strategic place in a corner of the ceiling. Sensor has a wide-angle Field of View (FOV).

5.2.2 Lighting conditions

As explained in former chapters, colour cues in general are to be used to obtain robust detection and tracking of people. The image sequences recorded within an indoor environment changes quickly due to both continuous and sudden changes in lighting conditions, e.g. the diurnal variation of daylight and the switching of lights on and off. In order to achieve a certain degree of robustness in the detection process, the intended chromatic processing experimentation has to be as immune to light changes as possible. This is a general problem in colour vision, called colour constancy. The colour appearance depends on the brightness and the colour temperature of the light source. The dependency on the brightness can be resolved by transforming into different chromatic colour spaces such as the Hue-Saturation-Intensity (HSV) or Normalised RGB colour spaces. The information obtained from the usual three visible band colours (Red, Green and Blue) generated by a conventional visible band camera can be transformed to these more stable colour spaces. This transformation allows the detection process to be adaptive to different lighting conditions, where the RGB colours are likely to change more that the hue and saturation components of the HSV values for example. However in this thesis, Normlised RGB colour space only is used in the experimental test cases for the reasons given in section 4.5.

Different sets of room lighting conditions such as bright daylight, directional sunlight, cloudy day, electric bulbs lights and tubes emitting white light are recorded. Colour appearance is often unstable due to changes in both background and foreground lighting and the effect of sudden lighting condition changes such as flicking on/off the room lights is also recorded. For example, one test scenario involves starting with the blinds closed and then opening the blinds really fast. This floods the camera with new light coming from outside. The second test is then to quickly drop the blinds so that the scene grows dark really fast. All the test videos investigating the effect of lighting conditions are recorded and stored according to Table 5-1 below. These videos (available on CD in the Appendix) are the high-resolution ones obtained from the camera.

Lighting Condition	Filename
Bright daylight	bright.wmv
Cloudy day	cloudy.wmv
Directional light	directional.wmv
Electric bulbs	bulbs.wmv
Sudden change of light	sudden_change.wmv

Table 5-1 : Changing lighting conditions

5.2.3 People including clothing and motion

An important factor to be considered when dealing with image monitoring systems depends on what is being monitored in the first place, e.g. cars, luggage, people etc. In this thesis, people are being monitored and hence the different variables that people can exhibit have to be investigated. The first variable to be investigated is the different paths that can be taken by a person within a room. Some of the tracks that are undertaken are shown in Figure 5-3 . Most of the collected data consisted of only one person moving around in the room in different directions and manners. However, a second person is also added to see how the trackers would react when there are multiple targets. The test videos are recorded and saved according to Table 5-2.

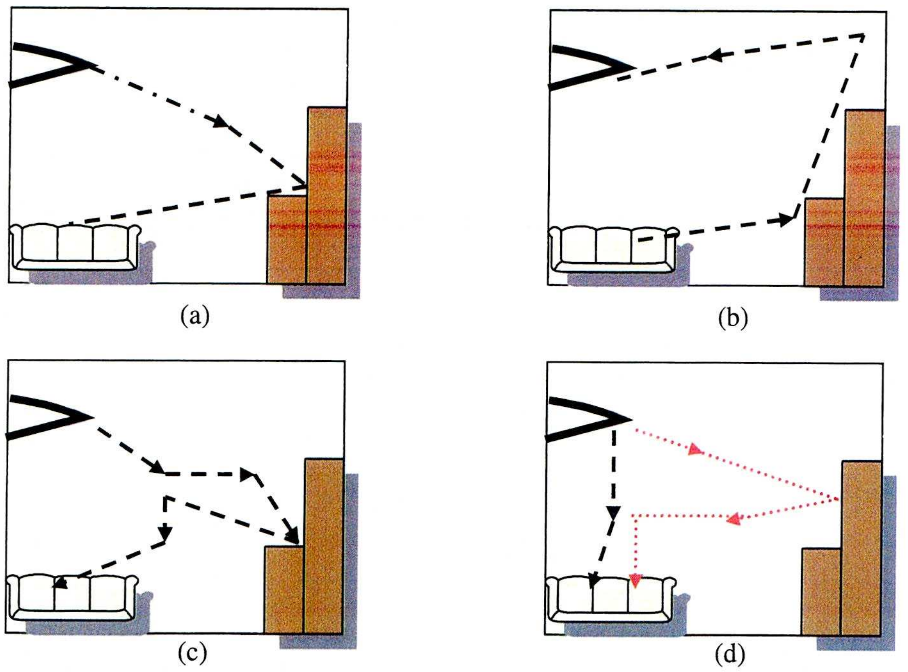


Figure 5-3 : (a) and (b) show simple paths taken by one person, (c) shows zigzag path taken by one person, and (d) shows two persons walking in the room

Track Taken	Filename
Single person, Straight line	oneperson_straightline.wmv
Single person, Going in circle	oneperson_circle.wmv
Single person, occlusion	oneperson_occlusion.wmv
Multiple persons, separate paths	twopersons_separate.wmv
Multiple persons, crossing paths	twopersons_cross.wmv

Table 5-2 : Paths taken by one or more persons.

Another variable investigated is the speed of motion. The detection process of the monitoring system has to be able to cope with different speeds of motion. An average walking speed inside an office environment is usually within the range 0.5 – 1 metre per second (m/s), although this depends heavily on factors such as height, weight, and age. For this data gathering exercise, speeds of motion are classified as slow, normal and fast, where normal is the general walking speed of a fit person in an enclosed space or room. Files are saved according to Table 5-3.

Speed	Filename
Slow	slow.wmv
Normal	normal.wmv
Fast	fast.wmv

Table 5-3 – Speed of motion

The clothing of people is also another important variable that needs to be considered. By clothing, it is meant the colour of the clothes used (intensity of colour, number of colours on a person, dark, bright etc) and the amount of the body skin it covers. It is expected that the detection system based on the un-mixing procedure will work best if the person is wearing clothing with a saturated colour that stands out from the background. However tests have to be made to ascertain how less colourful clothing would fare in this context. Files are saved according to Table 5-4.

Clothing	Filename
Highly saturated pure colour (Red)	red_shirt.wmv
Highly saturated pure colour (Green)	green_shirt.wmv
Pale colour, different background colour	pale_shirt_diff_background.wmv
Pale colour, similar background colour	pale_shirt_similar_shirt.wmv
Multi-coloured clothing	multi_coloured_shirt.wmv

Table 5-4 - Clothing

5.2.4 External Factors

The last variable to be given attention is what can be classified as 'external factors'. This section is concerned with other moving objects that can interfere and influence the detection and tracking system. Examples are curtains or roller blinds being moved by wind, house plants being swayed by wind or an electric table fan rotating while in operation. The motion of background objects is important, e.g. a curtain being blown by the wind could move forward and across pixel boundaries and depending on the method of motion detection being used this can generate a large number of false alarms. These background movements are not expected to impact considerably on the detection algorithms that are developed but their effects have to be considered nevertheless. Table 5-5 below shows the different videos that are recorded.

External Factor	Filename
Curtains moving in background	curtains.wmv
Rotating fan causing light to vary	fan.wmv
Plant moved by wind	plant.wmv

Table 5-5 : External Factors

5.2.5 Video recording and processing

A snapshot of the high resolution videos taken during the experimental scenario described above is shown in Figure 5-4. The captured videos are then reduced to the required sizes by using the bi-cubic interpolation technique [118]. In this technique, each output pixel value is a weighted average of pixels in the nearest 4-by-4 (or higher) neighbourhood. The analysis of these low resolution image sequences, i.e. the detection and tracking processes, is then done on an Intel Core® 2 Duo processor computer running Matlab 2008 with image resolutions ranging from 15×15 pixels to 35×35 pixels at a low frame rate of 10 fps more appropriate of a low quality sensor. A snapshot of the recording sequence in low resolution is shown in Figure 5-5. Please note that the images in the figure have been enlarged to enable them to be visible for printing. Each 'pixel' in these low resolution images printed here is in fact a collection of pixels representing the same sample point.

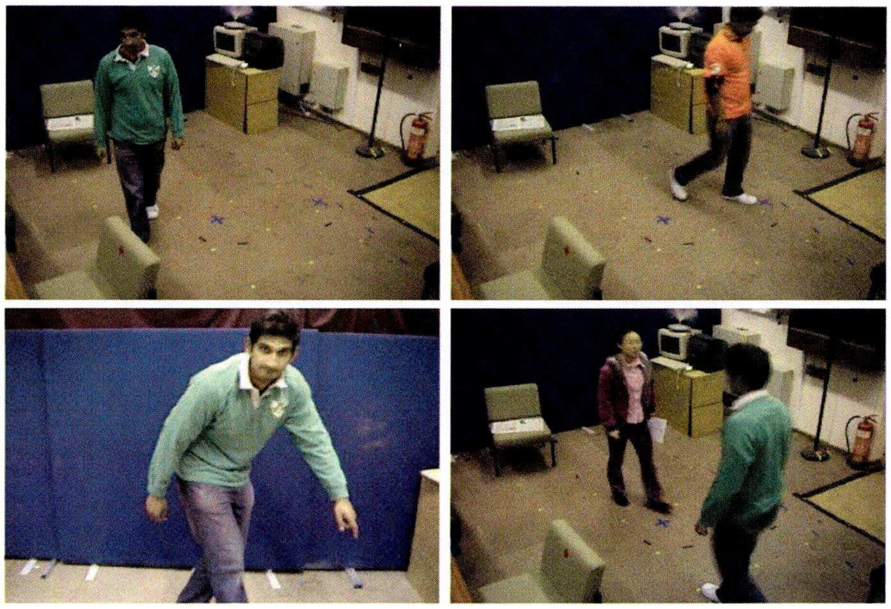


Figure 5-4 : Snapshot of the high resolution data acquisition video recording.

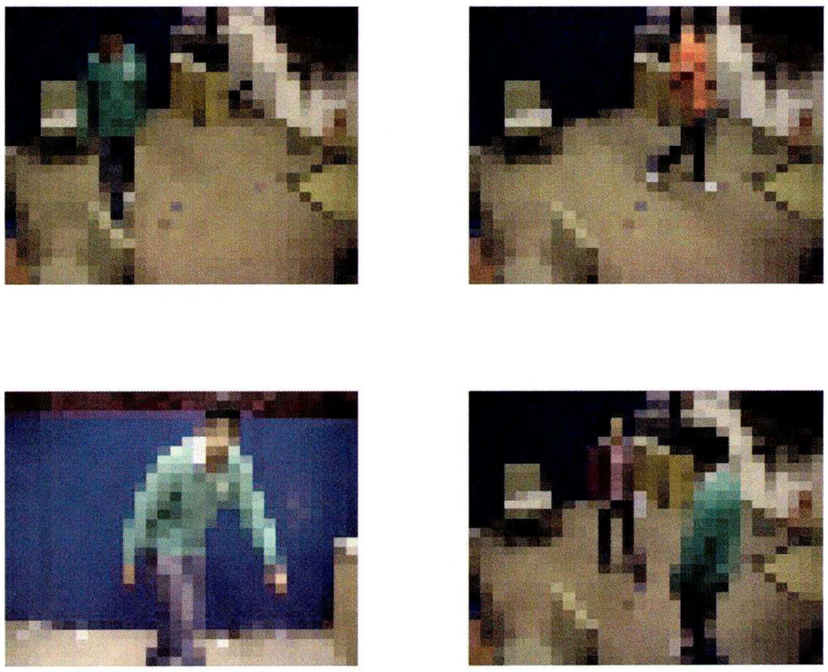


Figure 5-5 : Low resolution equivalent of the snapshots.

5.3 Results

5.3.1 Detecting the object and estimating its position

The first step in the testing and evaluation part of the experiment is to see if the methods of finding the centre of mass of an object using the un-mixing process together with bounding box approach to locate a central blob (see 3.5) can be used effectively on basic real imagery.. The first video to be tested is a 25×25 pixel image sequence with an object (a single person) of a few pixels in size moving in an approximate circular path (see Figure 5-6(a)). The lighting in this first test video is also kept constant.

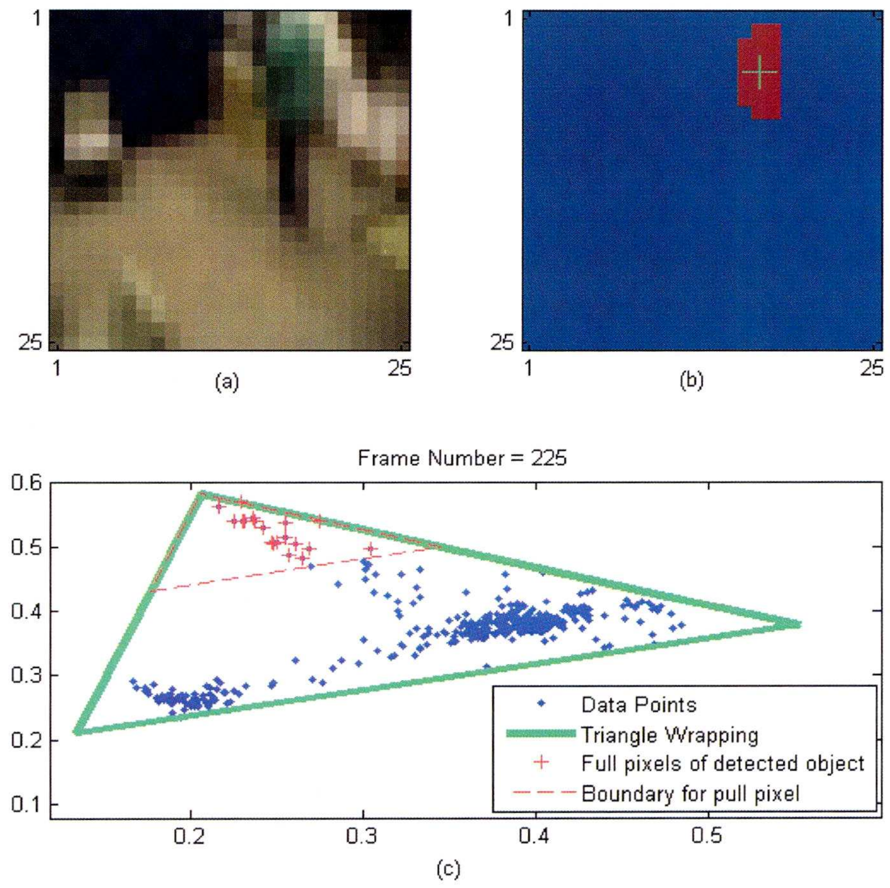


Figure 5-6 : (a) Image sequence containing a moving object (b) Detected centre of mass of object (c) Triangle wrapping around data points, and selected data points in red that represent the object's location in the image plane.

The vertex containing the object (or colour) of interest is chosen by the user by first going through each vertex and seeing what portion of the image it represents (e.g. object, background 1, background 2 etc). This selection by trial and error can be improved in the future by using all the three vertices to un-mix pixels during the initial frames and then checking which one of the three centroid locations obtained is changing. A changing centre of mass position means a moving object while a fixed one means a background. A threshold boundary of 0.25 on each side of the un-mixing triangle is used to allocate certain data points as being full pixels of the detected object, i.e. any points that lie within the fraction 0.25 of the length of each of the two sides meeting at the vertex representing the object (or colour) of interest are considered to be full pixels of that particular colour of interest (See page 3-71 of section 3.5.1 and also Figure 3-23 for detailed explanation). A snapshot of the detection process with the middle-vertex chosen as the un-mixing vertex is shown in Figure 5-6.

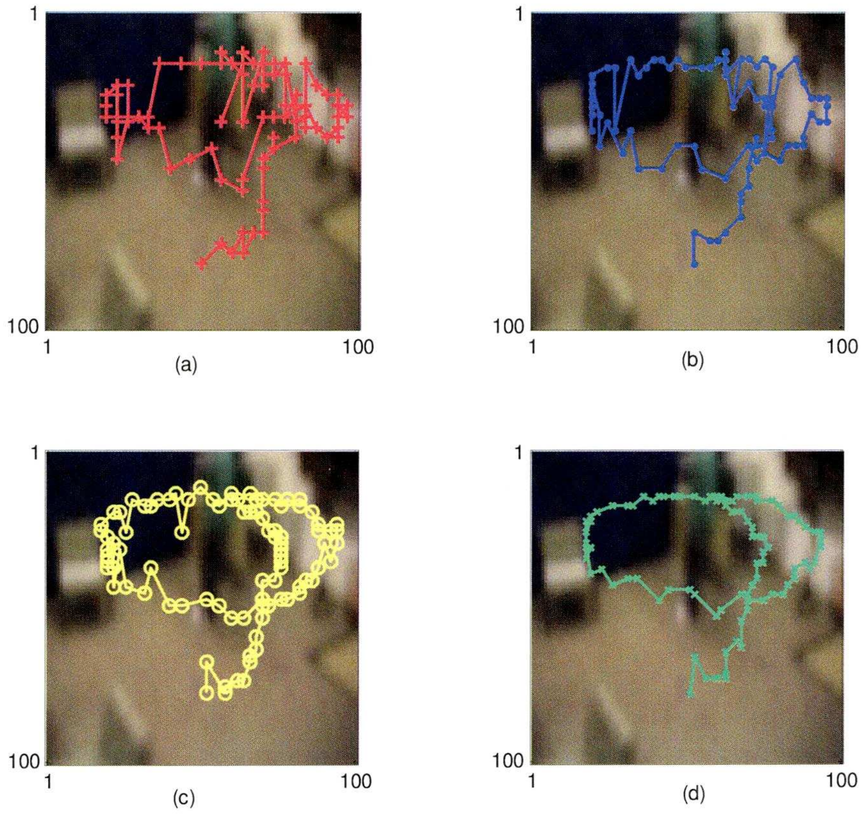


Figure 5-7: Position estimates of the object found when image resolution is (a) 15 × 15 (b) 20 × 20 (c) 25 × 25 (d) 30 × 30 pixels. The estimates have then been scaled up to a resolution of 100 × 100 pixels for comparison purposes.

With the centre of mass detection mechanism seen to be working, the experiment is then repeated for image resolutions ranging from 15×15 pixels to 30×30 pixels. The locations of these centroids for each resolution for the duration of the whole image sequence are shown in Figure 5-7. The values for these locations have been scaled up to an image resolution of 100×100 pixels to allow easier comparison of the results. The path obtained by joining each centroid's position estimate at a resolution of 15×15 consisted mostly of jagged lines but it still gave the impression of a circular trajectory when looking at the overall picture. This is a very good estimate for such a poor quality of image. One can see how the paths drawn by joining each centroid's position estimate get smoother as the resolution increases. These paths are super-imposed on each other in Figure 5-8 to appreciate even further the accuracy of the detected centroids as the image resolution changes. One way to smoothen the paths for the very low resolution image sequences is to use a Kalman filter. Not only can the Kalman filter smoothen the paths but it can also insert a certain dose of prediction at each point detection which is very useful in the overall tracking process. This is explained in the next subsection.

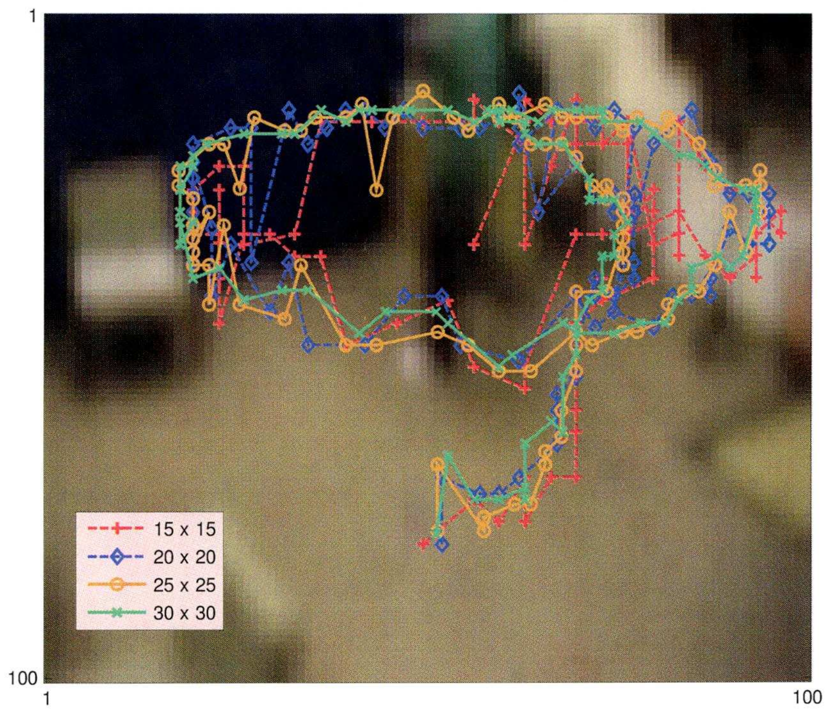


Figure 5-8 : Detected points superimposed on each other

5.3.2 Tracking the object using Kalman filters

With the position of the centre of object estimated in this way, a filter such as the Kalman filter can then be used to initiate a tracking process and improve the object position estimates. A constant velocity model with a state vector containing x and y coordinates whenever a measurement is made and velocity obtained from a change in position as a function of time is implemented using the equations 2-8 to 2-13. Two variables, the measurement noise N_y and the covariance error matrix E_y that are mentioned in Chapter 2, are given initial values as shown in equation 5-1. This particular value of E_y shows that an error of 2 pixels is assumed for both position and velocity estimates at the beginning.

$$N_y = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \text{and} \quad E_y = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \qquad (5\text{-}1)$$

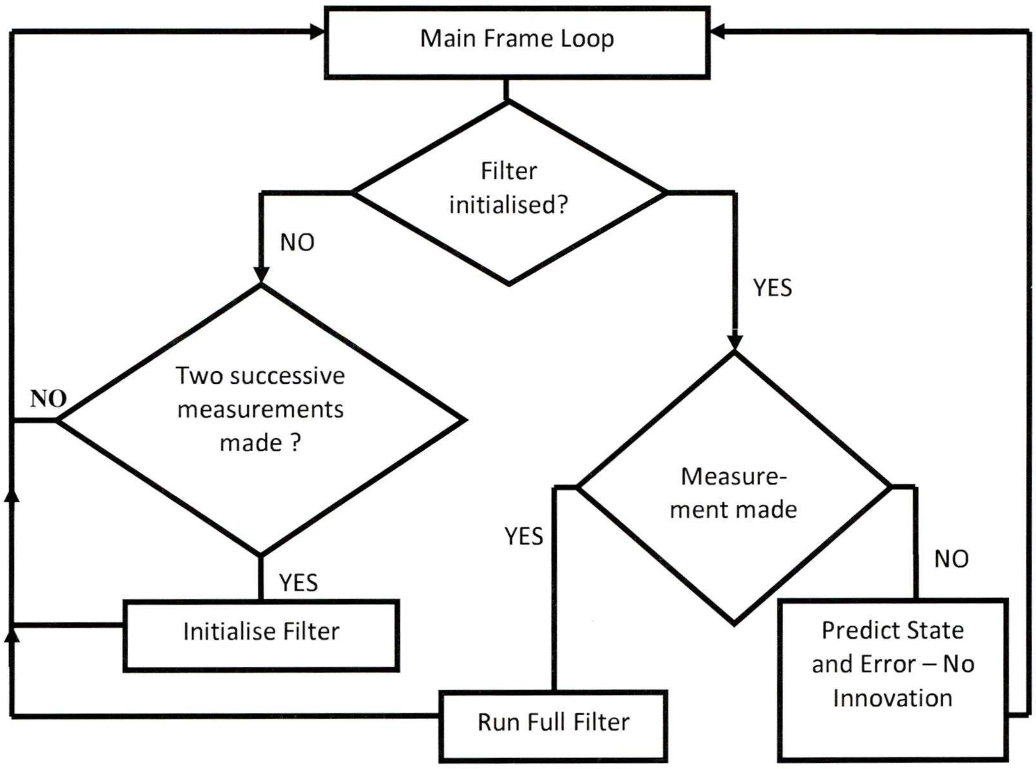


Figure 5-9 : Flowchart explaining the functioning of the Kalman filter with particular attention to the initialisation step.

The filter is initialised by using the two-point differencing method whereby the velocity is first estimated only after two position measurements are made. Once the filter is running, an estimate for the next measurement can be obtained by using information from previous data as shown in the flowchart in Figure 5-9. The results for the different image resolutions are shown next in Figure 5-10 with the thick white lines representing the paths obtained after running the Kalman filter (The estimates have then been scaled up to a resolution of 100 × 100 pixels for comparison purposes). This process is done online, i.e. in real time every time a centroid position update is available. It can be seen how the Kalman filtered paths are smoother the paths obtained after the un-mixing process only. There is less jaggedness even for the very low quality image sequences. The tracks are super-imposed on each other in Figure 5-11 for easier comparison. The implementation code of the Kalman filter is available in Appendix A3.

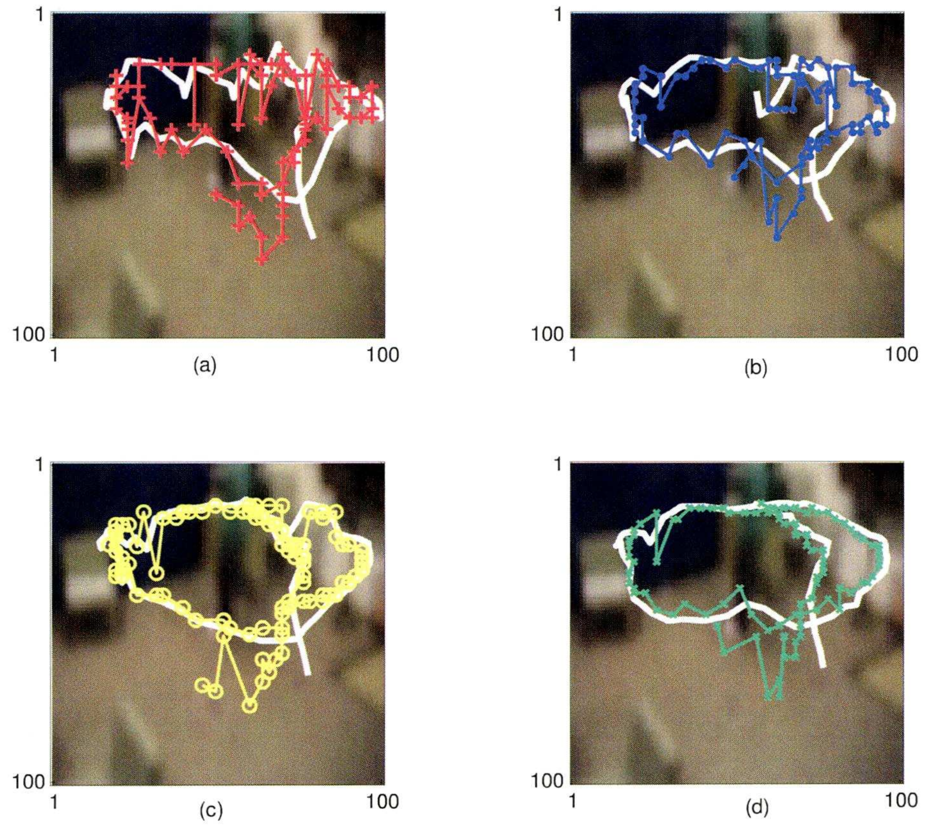


Figure 5-10 : Track estimation shown by thick white line after Kalman filter has been applied to each image resolution of (a) 15 × 15 (b) 20 × 20 (c) 25 × 25 (d) 30 × 30 pixels. The estimates have then been scaled up to a resolution of 100 × 100 pixels for comparison purposes.

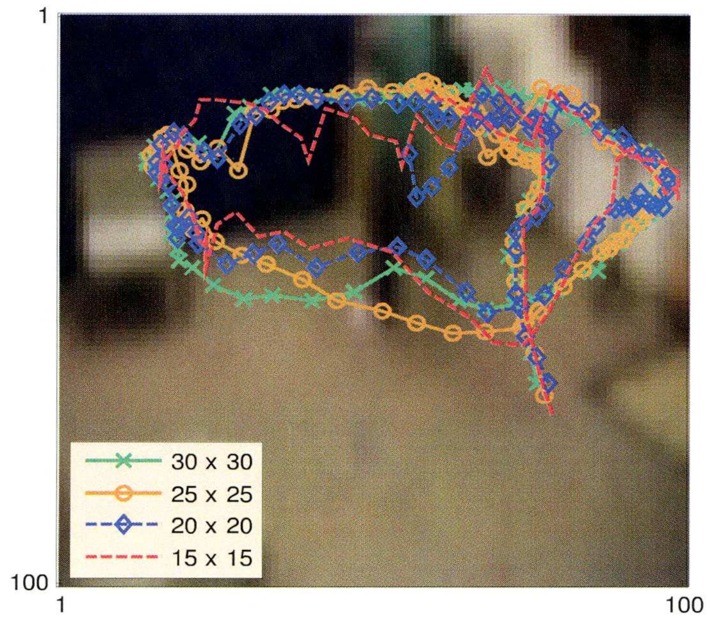


Figure 5-11 : Filtered tracks obtained from different resolutions superimposed on each other

It has been shown how the paths are made smoother but it is difficult to evaluate how efficient the Kalman filter is because of lack of ground truth. Normally this data will be in terms of a certain number of reference points on the floor which the object (here a person) has to cover. Then any deviation from these reference points can be considered as errors in track estimates. However, because of the nature of the un-mixing which singles out and track the person's coloured shirt rather the whole person itself, it is difficult to relate this data to the reference points. These results cannot be tested with other monitoring systems also because none that uses the same technique of tracking the colour only is available. To evaluate the effect of the Kalman filter, it is better to apply it to an artificially generated image sequence which contains three main colours with one of them moving around within the image boundaries. An example of un-mixing and data filtering at an instant in time on a 25 $\times$ 25 pixel resolution image sequence is shown in Figure 5-12. In general, it is found that the Kalman tracker filters the data well as the position errors are smaller in magnitude and smoother than after the un-mixing alone as shown in Figure 5-12 (c). The degree of smoothness is shown in (d) and (e) where the paths obtained after un-mixing only and Kalman filtering respectively are super-imposed on the real path. However every time the object changes direction, higher error values are obtained and this could be partly explained by the fact that a constant velocity model was used. A future improvement is to use a more complex Kalman tracker to better predict the position coordinates of the object when the latter changes velocity and direction. This constant changing of directions is not expected to happen very often within indoor environments with elderly people and one can assume that the constant velocity model is appropriate enough for now.

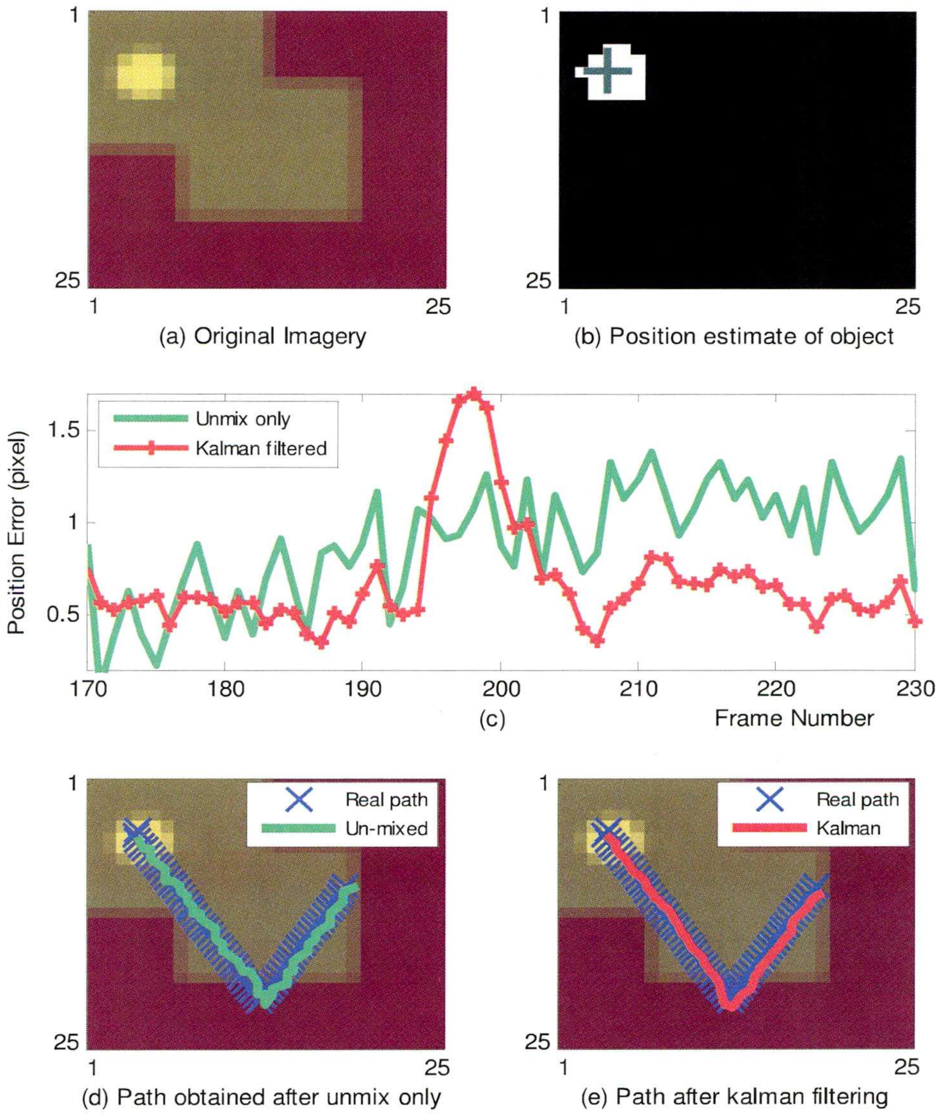


Figure 5-12 : (a) Original imagery at an instant in time - here Frame 230, (b) Position estimate of the object at Frame 230 after un-mixing, (c) Kalman filter gives position error values which generally smaller in magnitude and smoother than the ones obtained after un-mix only, (d) Path after un-mixing only, super-imposed on real path, (e) Path obtained after Kalman filter is used, super-imposed on real path.

One has to be aware that the Kalman filter applied in this manner is not being used to predict paths ahead in time but just for the smoothing of the detected noisy position measurements of the moving object. However sometimes it can happen that a position update is not available, e.g. temporary occlusion of object or failure of wrapping triangle to fit tightly around a quickly changing data points. To show how the Kalman filter can be used to estimate the position without a measurement, another test is done where the filter is only fed position measurements after every other five frames. The result is shown in Figure 5-13. It can be seen that even without any measurement made, the Kalman filter can give an estimate of the path reliably with position error values of less than 2 pixels. However these values tend to go very high when the object changes direction drastically.

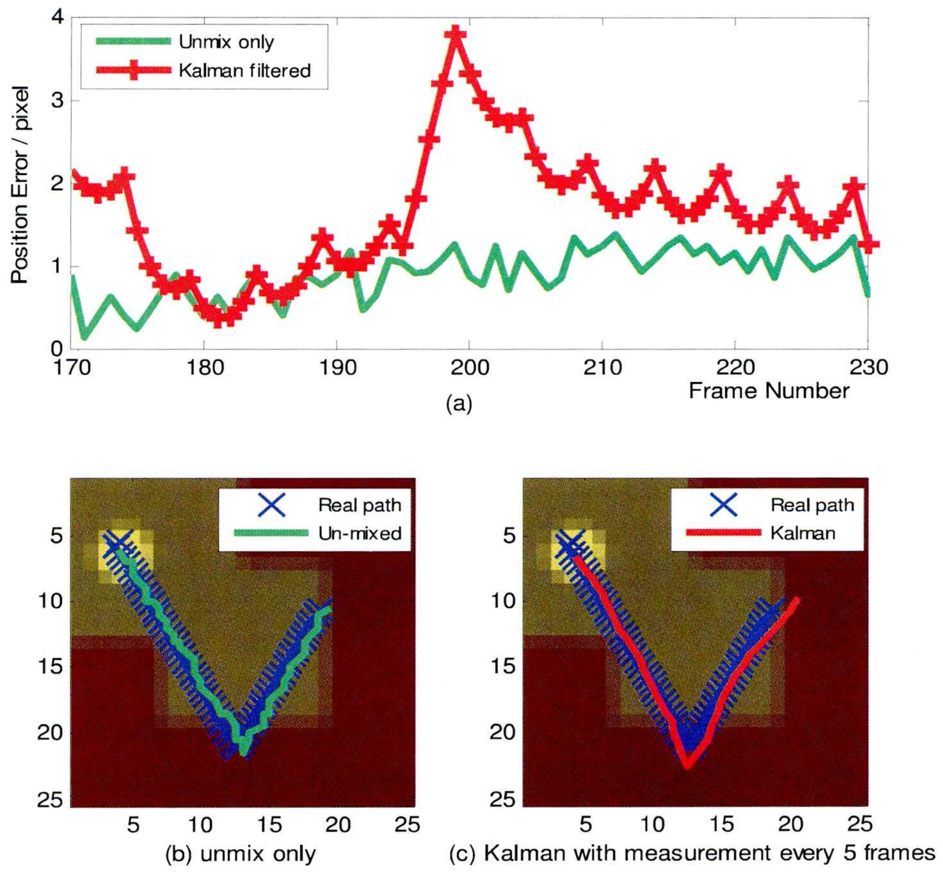


Figure 5-13 : Kalman filter gives position errors which are not always less in magnitude than the ones obtained after un-mix only, (b) Path after un-mixing only, super-imposed on real path, (c) Path obtained after Kalman filter is used with measurement made after every 5 frames, super-imposed of real path

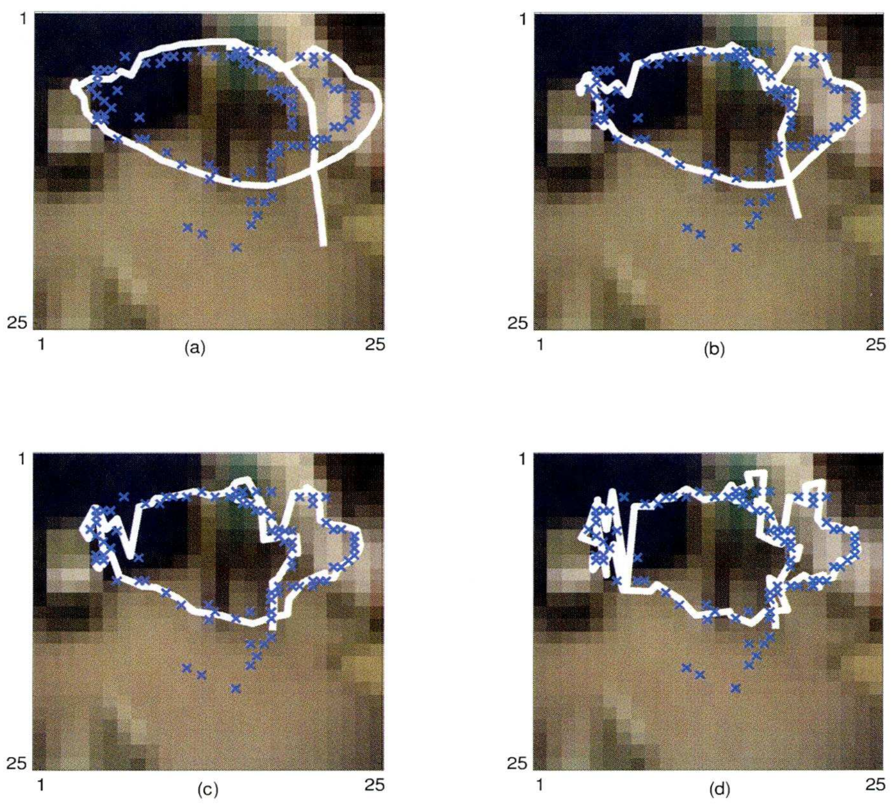


Figure 5-14 : Impact on the filtered path as the process noise is increased from (a) to (d)

In the actual implementation of the filter, the measurement noise covariance is usually measured prior to operation of the filter. Knowing the measurement error covariance is generally possible by taking some off-line sample measurements in order to determine the approximate variance of the measurement noise. The determination of the process noise covariance is on the other hand more difficult to determine as it is typically not possible to directly observe the process that is being estimated. Sometimes a relatively simple (poor) process model can produce acceptable results if one 'injects' enough uncertainty into the process via the selection of the process noise but there is no way to determine this value automatically apart from tuning the parameters and evaluating the effects of doing this action. An example of the impact of the process noise is shown in Figure 5-14. The value of process noise, more specifically the parameter $\tilde{q}$ of equation 2-15, is increased from 0.01 to 10 by using a multiplication factor after each test. One can see how the tracks are smoother for smaller values of process noise but this also means that these tracks obtained can be further from the actual path taken by the object, for example in Figure 5-14 (a), the filtered path on the right part of the diagram is more than two pixels from the detected points. There is a compromise that has to be made in choosing the right value for process noise depending on what the requirements of accuracy are.

5.3.3 Evaluation of the system according to experimental plan

Among the tests to be performed in the experimental plan, those involving the lighting conditions are perhaps the most important as this is a variable that is bound to change at some point when monitoring a person in a room for a long time. In the videos involving bright daylight, cloudy day and electric bulb as sources of light, they all allowed the successful detection of the person walking. These results are available in the appended CD. The main test is the one involving sudden change of light. Figure 5-15 shows two frames, the 1st and 25th with their respective data point distributions, of an image sequence in which the light is switched off suddenly after frame 1. In (a), the centroid of the object is detected without any problem when the room is well lit at Frame No. 1. The room lighting is switched off just after Frame 1. In Figure 5-15(b), the centroid is still detected at Frame No. 25 after the un-mixing triangle has adapted to the new data point distribution. During the period between frame 1 and 25, the triangle tries to fit itself around the changing data points and the evolution of the vertices is shown in Figure 5-16. Thus one can see how the triangle wrapping algorithm developed is effective in adapting to sudden changes in light conditions after a couple of frames when the optimiser 'realises' that the data point distribution has changed drastically. However, the un-mixing procedure is found to fail when the light levels are decreased even further than the light level in Frame 25 because the data points moved towards the origin of the axes representing the *NRGB* colour space and cluster themselves there. Monitoring is not expected to be performed in this type of low level lighting though in real life. Although the data points in the *NRGB* colour space change when lighting conditions are modified (i.e. not satisfying fully the colour constancy requirements), the change occurs in a such a way that the general shape of the data point distribution is conserved and hence allowing pixel decomposition and centroid position to be estimated.

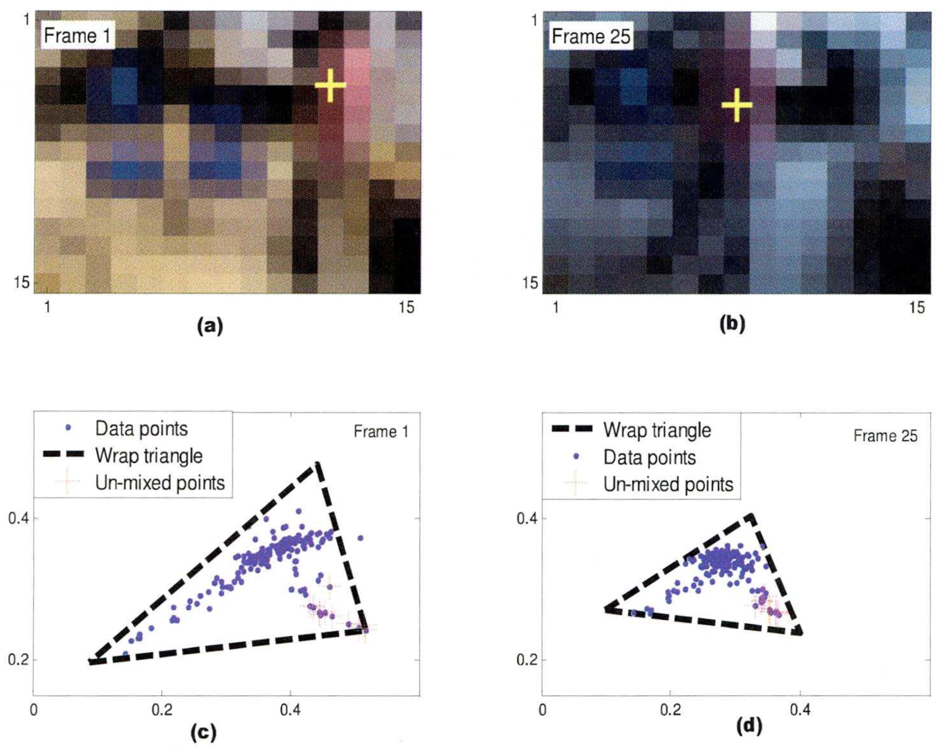


Figure 5-15 : Effect of sudden change of light. In (a), centroid is detected in well lit room. Light is switched off just after Frame 1. In (b), centroid is still detected after triangle adapts to new data point distribution.

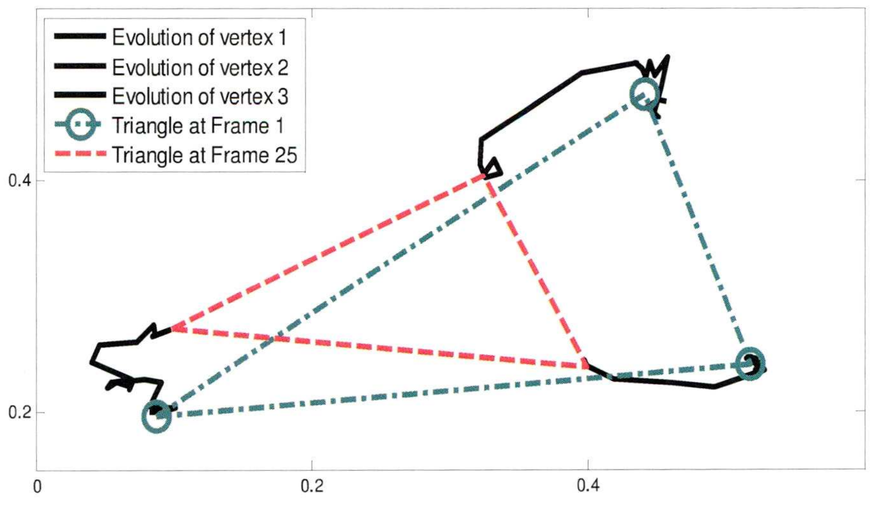


Figure 5-16 : Triangle tries to fit itself to the new data points from frame 1 to 24.

The other variable to be investigated is people. A person wearing a colourful shirt is made to walk in different paths and at different speeds according to the experimental scenario. It is found that the centre of mass of the shirt is detected in all situations (see appendix CD). Thus one can say that both style and pace of walking do not affect the detection process. One part of the experimental scenario that needs a little more attention is when the moving object of interest gets occluded as shown in Figure 5-17. In this figure, a person wearing a green shirt is moving from right to left of the camera's field of view. As the object moves, it encounters another object in its path and by going behind that object, it gets hidden temporarily (between frame 130 and 150 approximately) from the camera viewer. The object then re-emerges from the obstructed view as it carries on its path to the left of the image. When the un-mixing procedure is carried out, the centroid can be obtained quite easily in the beginning. As the object of interest starts to get occluded, the position estimate becomes very erratic and sometimes even non-existent (at frame 140). However, once the object reappears, the triangle adapts to the new data points and the object is detectable again. The bottom row of Figure 5-17 shows how the triangle wrapping process is very quick at detecting changes in the data point distributions, One way to deal with temporary occlusion is to use Kalman filters to 'predict' the path of the object a few frames ahead but its efficiency will depend on how many frames at a time that the object gets occluded.

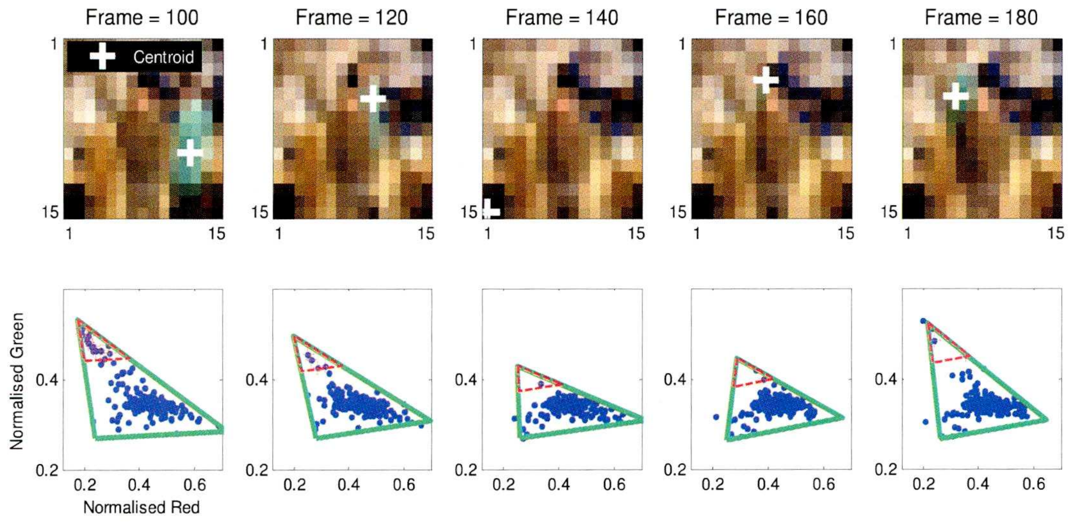


Figure 5-17 : Moving object being un-mixed gets occluded. Centroid position cannot be obtained at frame 140 when the object is hidden but once object reappears, it is detected again.

The last test to be mentioned in the 'people' variable section is the one involving the detection of more than one object of interest at a time. Two persons (objects) with bright coloured shirts, green and yellow, are made to walk in such a way that they are both in the camera's field of view and they are also allowed to interact with each other to confirm the capability of the system to handle occlusions effectively once more. For this experiment, it is necessary to know which vertex represents the colour or object of interest. This can only be done by trial and error unfortunately. Once this is known, the un-mixing process can be done on each of the two vertices and a snapshot of the results is shown in Figure 5-18(a) where two objects with 'bright' colours are detected at the same time and their respective paths are also shown. Figure 5-18(b) shows how the data points near two vertices are being un-mixed simultaneously to give the location of the centroids for object 1 and object 2 respectively. However it must be noted that for the un-mixing to be successful at all times in detecting two objects simultaneously, the objects' colours must be different from each other and also from the background colour. Otherwise the triangle is not properly formed and can result in many false positives. Hence for this system to work in real life, the people being monitored need to wear special clothing based on some colour code when there is more than one person involved. This is not expected to be a problem as the subjects being monitored in hospital wards or prison spaces can be made to wear special clothing to enable the monitoring process to happen without the need to impinge on their privacy.

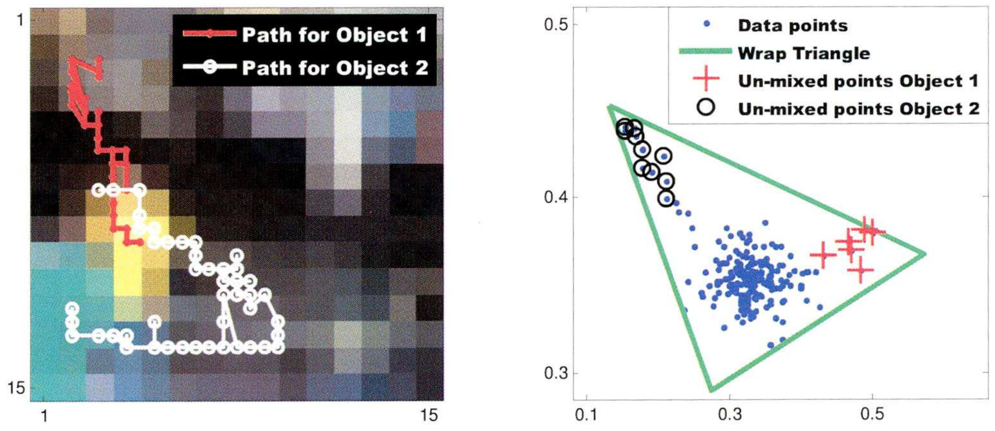


Figure 5-18 : (a) Two objects with 'bright' colours are detected at the same time. A path is obtained after un-mixing a certain amount of frames (b) Triangle wrapping and double un-mixing at an instant in time.

5.4 Development of Graphical User Interface (GUI) for real-time implementation

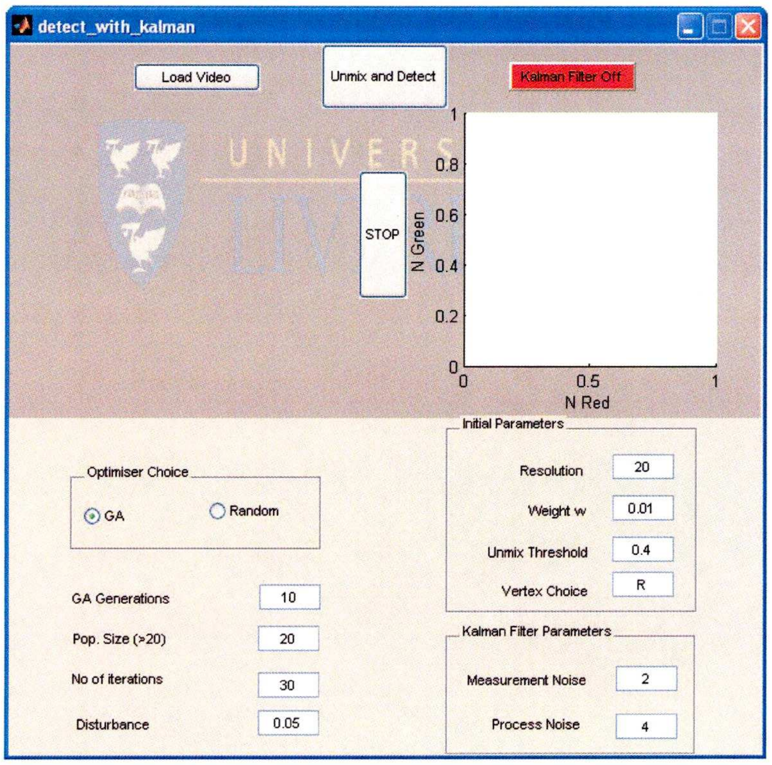
All the tests done so far have been done on computer programs executed from the keyboard or command line. While this approach is good enough during the development stages, it is less appealing when the system is to be deployed in the real world as an application for less technically minded people to use. One way to reduce this unease is by creating a Graphical User Interface (GUI) together with appropriate instructions that can do whatever function the original program can do but in a more user friendly way. A GUI is a type of interface which allows people to interact with electronic devices by using graphical icons and other visual indicators to represent the information and actions available to a user. The actions are usually performed through direct manipulation of the graphical elements.

There are main phases to observe in the development of GUIs [119]. These are: (1) Analysis (2) Design (3) Prototyping. The first phase involves answering questions such as, who will be using the interface and how will it be used? For example, GUIs developed for scientific experiment monitoring applications will have different target audiences and layouts from an interface designed for language learning software applications. The analysis phase can become very involved and complicated depending on the goals and can require developing user case scenarios, identifying the expertise of the user, computer system limitations, and plans for future upgrades based on user feedback [119]. In the design phase, the general layout of the GUI can be started based on certain considerations such as cognitive and physical considerations. Cognition refers to people's ability to think and learn. Organising functions and controls into groupings or not requiring the user to remember many things at once are examples of how using a GUI can be made into a pleasant experience. As for the physical considerations, these refer to the ways of interacting with the GUI such as keyboard, mouse, monitor and other input-output devices. The last phase (prototyping) can then be done. Prototyping usually involves doing a mock-up drawing on paper of the desired GUI with buttons, text entries and plots and arranging them so that everything blends well in the layout. Finally, the actual construction of the GUI can then be started using software such as Matlab's Graphical User Interface Development Environment (GUIDE) [98, 119].

An influential requirement in the successful development of the GUI is that it has to work in real-time on a normal computer or laptop. While all the coding and testing so far are shown to work on offline data, time and processing requirements have not been given much importance. There are many steps one should go through to take an image/video processing algorithm that is developed in a research environment to an actual working product. A common misunderstanding regarding the concept of real-time is that since hardware is getting faster and more powerful each year, real-time constraints can be met simply by using the latest, fastest, most powerful hardware, thus rending real-time a non-issue [120]. The problem with this argument is that it is often the case that such a solution is not viable, especially for consumer electronics embedded systems that have constraints on their total system cost, size, power consumption and user-accepted response time. One needs to address many challenging issues when developing a real-time image or video processing system. The solution often ends up as some combination of hardware and software approaches. From the hardware point of view, the challenges are to determine what kind of hardware platform and architecture are best suited for a given image/video processing task among the available hardware choices. In this thesis, a standard personal computer with Intel Core Duo® processor and 2 GB of memory together with a webcam are used as the host hardware. From the algorithmic and/or software point of view, the challenges involve being able to guarantee that real-time deadlines are met, which for example could mean making choices between different algorithms based on computational complexity or using a real-time operating system to manage various timing demands [120]. The parameters that can be changed to satisfy the demands are the number of function evaluations and population sizes used in the triangle wrapping optimisation process, the frame rate at which the video is acquired and the number of pixels in the input video (size of video). However, there is often a trade-off between algorithmic complexity and performance of the system.

5.4.1 GUI used in real world setting

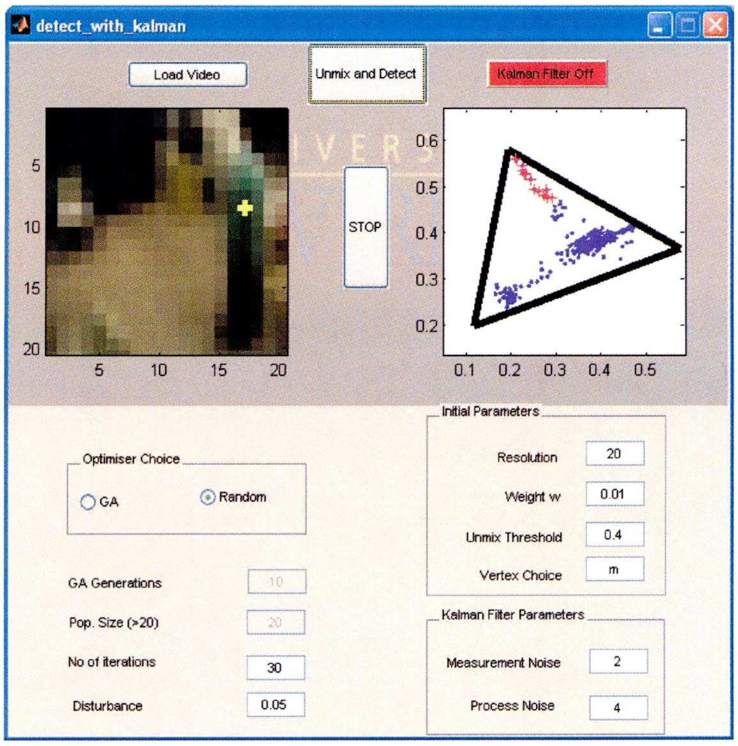
The GUI is created based on the requirements discussed previously. The initial parameters that can be modified by the user are: Video resolution, number of generations and size of population for the GA optimiser, weight factor used in multi-objective cost function, unmix threshold on the triangle and the choice of the vertex that represents the colour or endmember of interest. The GUI also has the option to turn on and off the Kalman Filter, and the measurement and process noises can be modified before the filter is switched on. This is made possible by using a 'Toggle' button, as shown in red in Figure 5-19. The 'Load Video' is used to load recorded movie clips into the system. There is also the possibility to feed the images of a webcam directly into this software. Once the movie is loaded, the detection process can be started by pressing the 'Unmix and Detect' button. A snapshot of the GUI in action is shown in Figure 5-20 and Figure 5-21. The process can be interrupted at any instant in time by pressing the 'STOP' button. This button is also useful if one wants to load a new video without closing the whole GUI.



Optimiser Choice: [✓ GA] Random

GA Generations	10
Pop. Size (>20)	20
No of iterations	30
Disturbance	0.05
Resolution	20
Weight w	0.01
Unmix Threshold	0.4
Vertex Choice	R
Measurement Noise	2
Process Noise	4

Figure 5-19 : Graphical User Interface developed to test the various algorithms



Optimiser Choice

GA [✓ Random]

GA Generations 10

Pop. Size (>20) 20

No of iterations 30

Disturbance 0.05

Initial Parameters

Resolution 20

Weight w 0.01

Unmix Threshold 0.4

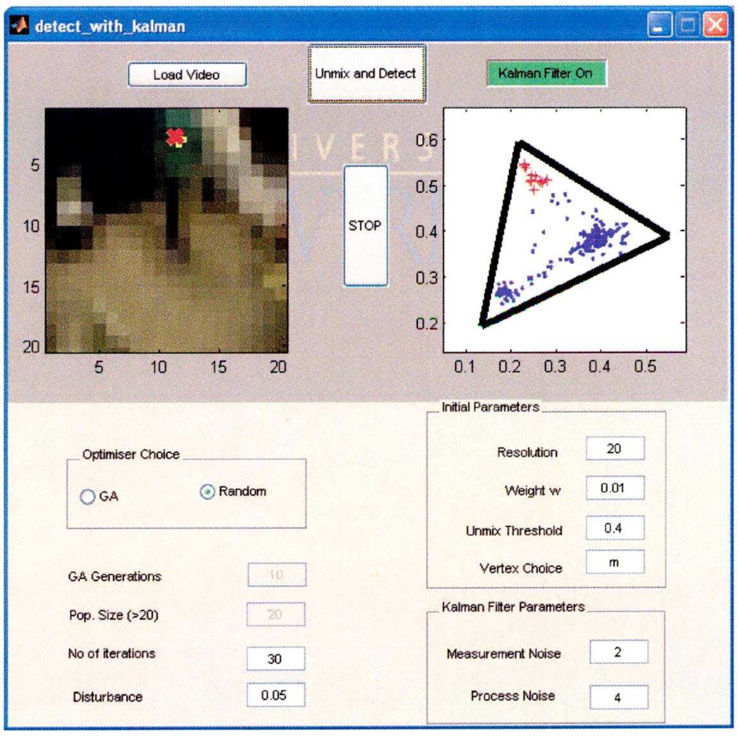
Vertex Choice m

Kalman Filter Parameters

Measurement Noise 2

Process Noise 4

Figure 5-20 : GUI in action



Optimiser Choice

GA [✓ Random]

GA Generations 10

Pop. Size (>20) 20

No of iterations 30

Disturbance 0.05

Initial Parameters

Resolution 20

Weight w 0.01

Unmix Threshold 0.4

Vertex Choice m

Kalman Filter Parameters

Measurement Noise 2

Process Noise 4

Figure 5-21 : Kalman filter switched on

5.5 Summary

For all the tests on real imagery mentioned in this chapter, it is difficult to quantify the detection errors due to the lack of ground-truth data but the detection process is considered to be reliable based on the synthetic imagery results and by visually inspecting and comparing the position of the detected object with the real image. It is not always easy to visualise the results of the detection process on all the videos of the experimental scenario on paper, and hence a CD containing results in video format is attached at the end. It has to be noted that the intensity contribution method, described in section 4.3, to find sub-pixel objects in real imagery is not shown in this chapter. This is because the results are not as good as expected. While this method work well during the tests on artificial experimental data [9], it is very difficult to repeat the same thing on real videos because of the increased impact of noise. This method needs further investigation before being deployed in a domiciliary care environment. Of high importance is the filtering threshold factor that needs to be obtained automatically as it currently has to be adjusted every time a new video is used. However the system as such can perform in biomedical applications such as microscope tracking where the videos are not as dynamic as those describing a room environment. The object detection method based on human skin is also not shown in this chapter because again the results that are obtained are not very reliable, especially when the image resolution is very low and the amount of skin detected by the sensor is of a few pixels only. This method can have its use in other object detection applications where such low levels of image resolutions are not required.

CHAPTER 6

6. Conclusions and further work

6.1 Summary

This thesis has examined the problem of tracking people within indoor environments with minimum privacy intrusion. In particular, the following questions have been answered:

- How can cameras film something but at the same time preserve the privacy of the subjects being monitored?
- What algorithms are useful in achieving detection of small objects in low resolution image sequences of poor quality?
- Can techniques developed for satellite reconnaissance techniques be successfully modified and adapted to indoor environments?

The first question was investigated in Chapter 3. It was shown how the use of low resolution cameras could be an answer to the preservation of privacy. By using this type of cameras that cannot form true images of the subjects being filmed, monitoring systems minimise the intrusion factor whilst still being able to monitor the motion of people and objects within the field of view. This is made possible if all the objects of interest are comparable in size to the sensor elements and hence cannot be fully resolved. It was also shown how the information contained in a colour image could be put to good use when using this type of images. Several representations of colour were analysed and the Normalised RGB representation was chosen as the main colour space during the implementation stage of the thesis.

The second and third questions were answered in chapters 3 and 4. Traditional detection techniques such as image differencing or optical flow methods were not used because of the high level of noise that exist in these images and also given their highly dynamic nature. A new technique, based on satellite reconnaissance applications, to the ones currently used in current monitoring systems was proposed to be used. In this technique, each pixel is assumed to be a linear combination of different classes of objects. In other words, each pixel in a given image contains a proportion of one or more definite colours. Each mixed pixel is then decomposed into a combination of the individual colours by using the un-mixing process to estimate the location and the size of a particular colour (or object) of interest. This pixel decomposition process requires that a triangle is wrapped tightly around certain data points. While several algorithms exist to achieve this wrapping, one has to be aware that they are designed for single instances of satellite images and hence cannot be used in the case of a dynamic image sequence with changing number of target objects and lighting conditions for example. Therefore a novel wrapping algorithm for these environments had to be developed to answer these needs. Several optimisation algorithms were tested and one based on a genetic algorithm was chosen and implemented to achieve this. The combined use of the new wrapping algorithm and the un-mixing process enabled the centres of objects of interest to be detected in colour image sequences of low pixel resolution. A related requirement that was also investigated was to know how low the image resolution could go and still give viable tracks. This detection technique also has the advantage of not requiring a static camera as is the case for frame-differencing methods.

In addition to answering these three questions, tracking, which is another important aspect of monitoring systems, was also investigated. Detecting an object on its own is not that useful and a way to generate continuous tracks or even predicting the object's track as it moves around in a scene had to be found. The state-space model was used to model the state of this dynamic system. It consisted of two sets of equations, the system equations and the observation equations. The Kalman filter was chosen to estimate the state of the system while only having access to noisy and/or inaccurate measurements. It was found that this filter performed well when the system was modelled as a linear Gaussian system and hence there was no need to look at other predictive filters. The algorithms that were developed were then tested on real imagery. As these algorithms were aimed mainly to be used in homes for the elderly, an experimental scenario containing data for all the various possibilities that could happen in these environments was designed. The robustness of the monitoring algorithms was also shown to be reliable when used in a series of real-life challenging scenarios such as quickly changing illumination conditions and occlusion of objects. The triangle wrapping process reacted well to such changes and conclusive path estimation results of walking people were obtained.

6.2 Further work

In the immediate future, the camera used to record the scenarios should be changed to the real low resolution image sensor that is intended to be used. Although care was taken while recording the various experimental videos to make them be as close as possible to reality by turning off all special signal processing features that exist in the camcorder, one can never be sure what the sensor response would be. Some specific characteristics such as dynamic response, gamma rating, and sensitivity as introduced in chapter 3 can only be obtained while using the sensor itself. Together with a change in the sensing equipment, the tests should be carried out over longer time periods. The system has been tested on short video clips of a few minutes so far but this needs to be extended to longer hours to better reflect the requirements of the system. Moreover by doing this, patterns of behaviour may emerge and the system could be further optimised to deal with this new knowledge.

Whilst the algorithms mentioned are sufficiently accurate for the current environment, for other applications where better spatial tracking accuracy is required, the following speculative modifications could be considered to improve the system. The detection process could be improved as the current one is limited to dealing with scenes containing three or four main colours only. If the method depending solely on pixel decomposition is to be preserved to detect more objects and hence colours, it has to be upgraded to be able to create simplexes in higher dimensions, e.g. a tetrahedron. However it must be noted that for the un-mixing to be successful at all times in detecting many objects simultaneously, the objects' colours must be different from each other and also from the background colour. Otherwise the simplex would not be properly formed. The fact that the people being monitored have to wear clothes based on a special colour code also limit the system's capabilities in environments where people wear uniforms or plain shirts, e.g. in an office. The detection process itself could also be modified to become a combination of pixel decomposition and some other technique. For example, a combination of linear un-mixing and knowledge of hue values of objects of interest would help avoid this initial trial and error step that exists while choosing which vertex represents which object. Successful matching of hue values with the appropriate vertex would also mean that the system would be able to lock the object of interest at all times even when new objects enter the scene which result in the change in location of the vertices of the wrapping triangle.

Another improvement that could be brought concerns the optimisation algorithm used in the triangle wrapping process. An adaptive form of the genetic algorithm optimiser and stochastic optimiser could be implemented to increase the computational workload involved currently. Generally, it is not always required to run the computationally hungry GA optimiser on every frame because it was found that the data points to be wrapped did not change by much when there were no major environment change such as sudden flicking off/on of light and new object entering or leaving the field of view. The points were not static but contained some noise due to the poor resolution on the images being captured. For this type of distribution, the simple stochastic optimiser is expected to achieve good results and if a method is found to switch between GA and stochastic optimisation during the running of the system, more memory and CPU resources could be released for other purposes while there are no challenging changes in the environment. Other purposes could be to increase the number of sensors used to improve the detection process by reducing occlusion and giving better track estimates for example.

The current system could also be taken to another level of sophistication by adding 'intelligence' to it. The addition of a multiple hypothesis tracker (MHT) to deal with ambiguous measurements and the motion of more than one person at a time would be one of the ways to achieve that. As mentioned in Chapter 2, the MHT algorithm searches for motion correspondences by examining several frames of data. It also has the ability to create new tracks for objects entering the field of view (FOV) and terminate tracks for objects exiting the FOV. It can also handle occlusions, that is, continuation of a track even if some of the measurements from an object are missing. Another way to achieve intelligence in the system is to design a framework to model the behaviour of people based on prior knowledge of these sensitive environments. This framework could then be used to better predict tracks taken by people by eliminating certain paths. There are several examples of known human behaviour:

- People are not expected to jump from one position to another position far from the original position all of a sudden.
- The speeds of motion of human beings are usually within a defined range in indoor environments.
- People have a tendency not to walk near walls and try to avoid colliding against large scale static obstacles such as tables or chairs objects.
- There are certain places such as beds or sofas where people are expected to be in horizontal position and not moving a lot.
- People on their own will walk faster than people in groups. The behaviour of human beings among other human beings is more difficult to model. Depending on the social context, a human being might try to avoid getting in the way of another person walking towards him or her if they do not know each other or they might stop and talk to each other.

By making use of ideas from psychology research, an automated simulation of human behaviour could be achieved with a view to incorporating it in the monitoring system.

Bibliography:

1. Fuhrmann, D. R., "Simplex shrink-wrap algorithm", Proc. SPIE Vol. 3718 Automatic Target Recognition IX, 501-511, 1999.
2. Nascimento, J.M.P and Dias, J.M.B, "Vertex Component Analysis: A Fast Algorithm to Unmix Hyperspectral Data", IEEE Trans. on Geoscience & Remote Sensing, Vol. 43, 898, 2005.
3. Intille, S. S., Davis, J. W., and Bobick, A. F., "Real-Time Closed-World Tracking", Proc. IEEE Computer Society conference on Computer Vision and Pattern Recognition, 697 – 703, June 1997.
4. Fisher, R. B. et al., "Dictionary of computer vision and image processing", John Wiley & Sons, England, 2005.
5. Wren, C. R. et al., "Pfinder: Real-time tracking of the human body", IEEE Transactions on pattern analysis and machine intelligence, Vol. 19, No. 7, July 1997.
6. Russ, J. C., "The image processing handbook" Third edition, CRC Press, USA, 1999.
7. Haritaoglu, I., Harwood, D., and Davis, L. S., "W^4 : Real-time surveillance of people and their activities", IEEE Transactions on pattern analysis and machine intelligence, vol. 22, No. 8, August 2000.
8. Merton MIMS website, http://www.merton.gov.uk/living/health/mims.htm, retrieved October 2008.
9. Govinda, V., Ralph, J.F., Spencer, J.W., Goulermas, J.Y. and Smith, D.H., "Tracking subpixel targets in domestic environments", Signal and Data Processing of Small Targets 2006, Proc. SPIE, Vol. 6236, 623603, 2006.
10. Xu, S. and Jones, G. R., "Event and movement monitoring using chromatic methodologies", IoP Measurement Science and Technology 17, 3204-3211, 2006.
11. Bowyer, K., Kranenburg, C., and Dougherty, S., "Edge detector evaluation using empirical roc curve", Computer Vision Image Understand. 10, 77–103, 2001.

12. Canny, J., “A Computational Approach to Edge Detection”, IEEE Trans. Pattern Analysis and Machine Intelligence, 8: 679-714, 1986.
13. Horn, B. and Schunk, B., "Determining optical flow", Artificial Intelligence 17, 185–203, 1981.
14. Barron, J., Fleet, D., and Beauchemin, S., "Performance of optical flow techniques", International Journal Computer, Vision 12, 43–77, 1994.
15. Gonzalez, R. C. and Woods, R. E., “Digital Image Processing”, Third Edition, Pearson Prentice Hall, New Jersey, USA, 2008.
16. Koller, D., Weber, J., and Malik, J., "Robust multiple car tracking with occlusion reasoning", Technical Report UCB/CSD-93-780, University of California, Berkeley,California, 1993.
17. Kameda, Y. and Minoh, M., "A human motion estimation method using 3-successive video frames", International Conference on Virtual Systems and Multimedia, 1996.
18. Comaniciu, D. and Meer, P., "Mean shift: A robust approach toward feature space analysis", IEEE Transactions Pattern Analysis Machine Intelligence 24, 5, p. 603–619, 2002.
19. Comaniciu, D., Ramesh, V., Andmeer, P., "Kernel-based object tracking", IEEE Transactions Pattern Analysis & Machine Intelligence 25, 564–575, 2003.
20. Kass, M., Witkin, A. and Terzopoulos, D., “Snakes : Active Contour Models”, International Journal of Computer Vision, 321 – 331, 1998.
21. Snyder, W.E. and Qi, H., “Machine vision”, Cambridge Uni. Press, UK, 2004.
22. Yilmaz, A., Li, X., and Shah, M., "Contour based object tracking with occlusion handling in video acquired using mobile cameras", IEEE Transactions Pattern Analysis and Machine Intelligence 26, 11, 1531–1536, 2004.
23. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T. and Van Gool, L., "A comparison of affine region detectors", International Journal of Computer Vision 65(1/2):43-72, 2005.
24. Moravec, H, "Visual mapping by a robot rover", Proceedings of the International Joint Conference on Artificial Intelligence, 598–600, 1979.

25. Schmid, C., Mohrand, R. and Bauckhage, C. "Evaluation of Interest Point Detectors", International Journal of Computer Vision, 37(2), 151-172, 2000.
26. Trucco, E. and Plakas, K., "Video tracking : A concise Survey", IEEE Journal of oceanic engineering, Vol 31, No. 2, April 2006.
27. Yilmaz, A., Javed, O., and Shah, M., "Object Tracking: A survey", ACM Computing Surveys, Vol. 38, No. 4, Article 13, December 2006.
28. Black, M. and Jepson, A., "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation", International Journal of Computer Vision 26, 1, 63–84, 1998.
29. Jolliffe, I. T., "Principal Component Analysis", Springer-Verlag NY, 2002.
30. Fieguth, P. and Terzopoulos, D., "Color-based tracking of heads and other mobile objects at video frame rates", IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 21–27, 1997.
31. Tao, H., Sawhney, H., and Kumar, R., "Object tracking with bayesian estimation of dynamic layer representations", IEEE Transactions Pattern Analysis Machine Intelligence 24, 1, 75–89, 2002.
32. Sethi, I.K., and Jain, R., "Finding trajectories of feature points in a monocular image sequence", IEEE Trans. on Pattern Analysis and Machine Intelligence, 9(1):56-73, 1987.
33. Veenman, C., Reinders, M., and Backer, E., "Resolving motion correspondence for densely moving points", IEEE Transactions Pattern Analysis Machine Intelligence 23, 1, 54–72, 2001.
34. Welch, G. and Bishop, G., "Course 8 - An introduction to the Kalman filter", ACM SIGGRAPH 2001, California, August 2001.
35. Simon, D., "Optimal state estimation – Kalman, H infinity, and Non linear approaches", Wiley & Sons, NJ, USA, 2006.
36. Goldenstein, S., "A Gentle Introduction to Predictive Filters", Revista de Informatica Teórica e Aplicada (RITA), Volume XI, Number 1, 61-89, October 2004.
37. Brown, R. G., "Introduction to random signal analysis and Kalman filtering", John Wiley & Sons, USA, 1983.

38. Kalman, R. E., "A new approach to linear filtering and prediction problems", Journal Basic Engineering, vol. 82D, 35-45, March 1960.
39. Grewal, M.S., Andrews, A.P., "Kalman Filtering : Theory And Practice Using MATLAB", 3rd Edition, N.J. Wiley, 2008.
40. Maybeck, P. S., "Stochastic models, estimation, and control, Volume 1", Chapter 1, Academic Press, New York, 1979.
41. Bar-Shalom, Y., Rong Li, X., Kirubarajan, T., "Estimation with Applications to Tracking and Navigation", John Wiley & Sons Inc, Chapter 6, 2001.
42. Wan, E. and Van der Merwe, R., "The unscented kalman filter for nonlinear estimation", Proc. of IEEE Symposium 2000 (AS-SPCC), Canada, October 2000.
43. Kitagawa, G., "Non-gaussian state-space modeling of nonstationary time series", Journal American Statistics Association 82, 1032–1063, 1987.
44. Gustafsson, F. et al., "Particle Filters for Positioning, Navigation and Tracking", IEEE Transactions on Signal Processing, Special issue on Monte Carlo methods for statistical signal processing, Vol. 50, Nr. 2, 2002.
45. Bar-Shalom, Y and Fortmann, T.E., "Tracking and Data Association", Academic Press, 1988.
46. Cox, I. J., "A review of statistical data association techniques for motion correspondence", International Journal Computer Vision 10, 1, 53–66, 1993.
47. Reid, D. B., "An algorithm for tracking multiple targets", IEEE Transactions on automatic control, Vol. AC-24, No.6, December 1979.
48. Cox, I. J., and Hingorani, S. L., "An efficient implementation and evaluation of Reid's multiple hypothesis tracking algorithm for visual tracking", International Conference on Pattern Recognition, 437-442, 1994.
49. Isard, M. and Blake, A., "CONDENSATION - conditional density propagation for visual tracking", International Journal of Computer Vision, 29(1):5-28, 1998.
50. Chen, Y., Rui, Y. and Huang, T.S., "JPDAF Based HMM for Real-Time Contour Tracking", IEEE Conference on Computer Vision and Pattern Recognition, 2001.
51. Rabiner, L. and Juang, B., "An introduction to hidden Markov models", IEEE Acoustics, Speech & Signal Processing Mag.,Vol. 3, No. 1., 4-16, 1986.

52. Mansouri, A.R., "Region tracking via level set PDEs without motion computation", IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7):947-961, 2002.
53. Bertalmio, M., Sapiro, G. and Randall, G, "Morphing active contours", IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(7):733-737, 2000.
54. Davies, D., Palmer, P and Mirmehdi, M., "Detection and tracking of very small low contrast objects", Proceedings of the British Machine Vision 9th Conference, 599-608, September 1998.
55. Mallat, s. and Hwang, W.L., "Singularity detection and processing with wavelets", IEEE Transactions on Information Theory, 38(2), 617–641, March 1992.
56. Cheezum, M. K. et al., "Quantitative Comparison of Algorithms for Tracking Single Fluorescent Particles", Biophysical Journal, 2378-2388, Vol. 81, No. 4, October 2001.
57. Anderson, C. M., et al., "Tracking of cell surface receptors by fluorescence digital imaging microscopy using a charge-coupled device camera", Journal Cell Science 101:415– 425, 1992.
58. Gelles, J., Schnapp, B. J., and Sheetz, M.P, "Tracking kinesin-driven movements with nanometre-scale precision", Nature. 331:450–453, 1988.
59. Watman, C. et al., "Fast Sum of Absolute Differences Visual Landmark Detector",IEEE International Conference on Robotics and Automation, New Orleans, LA, April 2004.
60. Bohs, L. N. et al., "Speckle tracking for multi-dimensional flow estimation", Journal of Ultrasonics 38 , 369 -375, 2000.
61. Van Eekeren, A.W.M. et al., "Super-resolution on small moving objects", 15th International Conference on Image Processing, 2008.
62. Aggarwal, J. K. and Cai, Q., "Human motion analysis: A review", Computer Vision and Image Understanding, 73(3), 428-440, 1999.
63. Bobick, A. and Davis, J., "Real-time recognition of activity using temporal templates", Proceedings of the Workshop on Applications of Computer Vision, 39-42, 1996.

64. Ayers, D. and Shah, M., “Monitoring human behavior from video taken in an office environment”, Image and Vision Computing, 19(12):833–846, October 2001.
65. Nguyen, N. T., Bui, H. H., Venkatesh, S., & West, G., "Recognising and monitoring high-level behaviours in complex spatial environments", In IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2003), 620-625, 2003.
66. Sanchez, J. and Canton M. P., “Space image processing”, CRC Press LLC, USA, 1999.
67. Yao, S. et al., "Contrast signal-to-noise ratio for image quality assessment", IEEE International Conference on Image Processing 2005, Volume 1, 11-14, I 397-400, September 2005.
68. Russ, J. C., “The image processing handbook, Third edition”, CRC Press, USA,1999.
69. Busch, D., "Mastering Digital SLR Photography", Boston, MA, USA: Course Technology Publisher, 2004.
70. Wyszecki, G. and Stiles, W. S., “Color Science - Concepts and Methods, Quantitative Data Formulas”, New York: John Wiley, 1967.
71. Schubert, E. F., "Light Emitting Diodes, Second edition”, Chapter 17, Cambridge University Press, 2006.
72. Forsyth, D. A. and Ponce, J., “ Computer Vision A Modern Approach”, Pearson Education, NJ, USA, 2003.
73. Todt, E. and Torras, C., "Color constancy for landmark detection in outdoor environments", 4th European Workshop on Advanced Mobile Robots, Eurobot’01, 2001.
74. Smith, A. R., “ Color gamut transform pairs”, ACM SIGGRAPH Computer Graphics, Volume 12 , Issue 3, 12 -19, August 1978.
75. Govinda, V., Ralph, J.F., Spencer, J.W., and Goulermas, J.Y., "Spectral unmixing for tracking human motion in low resolution imagery", Journal of Physics, Conference Series 76 012031, 2007.

76. Mitchell, J, "MPEG Video Compression Standard", Kluwer Academic Publishers, 1996.

77. Poynton, C. A., “A Technical Introduction to Digital Video”, John Wiley & Sons, Inc., 1996.

78. Schowengerdt, R.A, “Remote Sensing: Models and Methods for Image Processing Second Edition”, Academic Press USA, 1997.

79. Petrou, M., “Mixed Pixel Classification : An overview”, Information Processing for Remote Sensing, World Scientific Publishers, 1999.

80. Singer, R. B., and McCord, T. B., “Large scale mixing of bright and dark surface materials and implications for analysis of spectral reflectance”, Proceedings Lunar and Planetary Science Conference, 10th, 1835-1848, 1979.

81. Boardman, J.W, “Analysis, understanding, and visualization of hyperspectral data as convex sets in n space”, Proc. SPIE Vol. 2480, 14-22, Imaging Spectrometry, 06/1995.

82. Boardman J. W., and Kruse, F. A., "Automated spectral analysis: A geologic example using AVIRIS data, north Grapevine Mountains, Nevada", Proceedings Tenth Thematic Conference on Geologic Remote Sensing, Environmental Research Institute of Michigan, Ann Arbor, MI, I-407 - I-418, 1994.

83. Chang, C. and Plaza, A., "A fast iterative algorithm for implementation of Pixel Purity Index", IEEE Geoscience and remote sensing letters, vol. 3, No. 1, January 2006.

84. Winter, M.E., "N-FINDR: An algorithm for fast autonomous spectral end-member determination in hyperspectral data", Proceedings of the SPIE Imaging Spectrometry V, San Diego, CA, 1999.

85. Hapke, B., “Bidirectional Reflectance Spectroscopy 1. Theory”, *Journal Geophysics Res.*, 86(B4), 3039–3054, 1981.

86. Plaza, A., Martinez, P., Perez, R. and Plaza, J., “A new approach to mixed pixel classification of hyperspectral imagery based on extended morphological profiles”, Pattern Recognition 37 (2004) 1097 -1116, 2004.

87. Clarke, T.A. Cooper, M.A.R. & Fryer, J.G., "An estimator for the random error in subpixel target location and its use in the bundle adjustment", Optical 3-D measurements techniques II, Pub. Wichmann, Karlsruhe:161-168, 1993.
88. Craig, M. D., "Minimum-Volume Transforms for Remotely Sensed Data", IEEE Trans. on Geoscience & Remote Sensing, Vol 32, No. 3, May 1994.
89. Winter, M. E., "A proof of the N-FINDR algorithm for the automated detection of end-members in a hyperspectral image", Proc. of SPIE Vol. 5425 Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery, p 31-41, 2004.
90. Konak, A., Coit, D. W. and Smith, A. E., " Multi-objective optimization using genetic algorithms : A tutorial", Reliability engineering and system safety, 91, 992 – 1007, 2006.
91. Messac, A., Puemi-Sukam, C, and Melachrinoudis, E, "Aggregate Objective Functions and Pareto Frontiers: Required Relationships and Practical Implications", Optimization and Engineering, 1, 171–188, 2000.
92. Das, I and Dennis, J. E., "Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems", SIAM Journal on Optimization, Volume 8, Issue 3, 631 - 657, 1998.
93. Cole-Rhodes *et al.,* "Multiresolution Registration of Remote Sensing Imagery by Optimization of Mutual Information Using a Stochastic Gradient", IEEE Transactions On Image Processing, Vol. 12, No. 12, December 2003.
94. Spall, J.C, "Introduction to stochastic search and optimization : Estimation, Simulation and Control", New Jersey, Wiley, 2003.
95. Karnopp, D.C, "Random search techniques for optimization problems," Automatica, vol. 1, 111-121, 1963.
96. Solis, F.J and Wets, J.B, "Minimization by random search techniques", Mathematics of Operations Research, vol. 6, 19-30, 1981.
97. Nelder, J. A. and Mead, R.,"A simplex method for function minimization", The Computer Journal, vol. 7, 308 -313, 1965.
98. http://www.mathworks.com/, retrieved August 2008.

99. Barton, R. R. and Ivey, J. S., “Nelder-Mead simplex modifications for simulation optimization”, Management Science, vol. 42, 954-973, 1996.

100. Holland, J.H., “Adaptation in natural and artificial systems”, University of Michigan Press, Ann Arbor, MI, USA, 1975.

101. Sigl, B., Golub, M. and Mornar, V., "Solving Timetable Scheduling Problem Using Genetic Algorithms", 25th Int. Conf. Information Technology Interfaces, Croatia, June 2003.

102. MacQueen, J.B, "Some Methods for classification and Analysis of Multivariate Observations”, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1967, Berkeley, University of California Press, 1:281-297, 1967.

103. Barber, C. B., Dobkin, D. P. and Huhdanpaa, H. T., "The Quickhull Algorithm for Convex Hulls," *ACM Transactions on Mathematical Software*, Vol. 22, No. 4, 469-483, December 1996.

104. http://www.qhull.org/ , Qhull code for Convex Hull, Delaunay Triangulation, Voronoi Diagram, and Halfspace Intersection about a Point, retrieved August 2008.

105. Dreyfus, S., “Richard Bellman on the birth of dynamic programming”, Operations Research INFORMS Vol. 50, No. 1, 48–51, 2002.

106. Cheezum, M. K., Walker, W. F. and Guilford, W. H., “Quantitative comparison of algorithms for tracking single fluorescent particles”, Biophysical Journal 81(4) 2378–2388, October 2001.

107. Tan, S. and Hart, D. P., “A novel particle displacement measurement method using optical diffraction”, IoP Measurement Science & Technology 13, No 7, 1014-1019, July 2002.

108. Kakumanu, P., Makrogiannis, S., and Bourbakis, N., "A survey of skin-color modeling and detection methods", Journal of Pattern Recognition, 40, 1106 – 1122, 2007.

109. Angelopoulou, E., Molana, R., Daniilidis, K., “Multispectral skin color modeling”, CVPR01, 2001.

110. Vezhnevets, V., Sazonov, V., Andreeva, A., "A survey on pixel-based skin color detection techniques", GRAPHICON03, 85–92, 2003.

111. Soriano, M., Huovinen, S., Martinkauppi, B., and Laaksonen, M, “Skin detection in video under changing illumination conditions”, In Proc. 15th International Conference on Pattern Recognition, vol. 1, 839–842, 2000.

112. Wang, Y and Yuan, B, ‘A novel approach for human face detection from color images under complex background’, *Pattern Recognition,* Vol. 34, 1983-1992, October 2001.

113. Sobottka, K and Pitas,I, "Face localization and facial feature extraction based on shape and color information", in Proc. IEEE Int. Conf. Image Processing, vol. III, 483 - 486, September 1996.

114. Maggio, E and Cavallaro, A., "Multi-part target representation for color tracking", Proc. of IEEE Int. Conference on Image Processing, Genova, Paper 3229, September 2005.

115. Cai, J. and Goshtasby, A., “Detecting human faces in color images”, Image and Vision Computing 18, 63–75, 1999.

116. Chai, D. and Ngan, K. N., "Face segmentation using skin-color map in videophone applications", IEEE Trans. on Circuits and Systems for Video Technology, Vol 9, No. 4, 551-564, June 1999.

117. McKenna S J and Nait-Charif H, "Summarising Contextual Activity and Detecting Unusual Inactivity in a Supportive Home Environment", Pattern Analysis and Applications 7(4), 386-401, December 2004.

118. Keys, G. K., "Cubic Convolution Interpolation for Digital Image Processing", IEEE Transactions on acoustics, speech and signal processing, Vol 29, No. 6, December 1981.

119. Marchand, P. and Holland, O. T., “Graphics and GUIs with MATLAB”, Third Edition, Chapman & Hall/CRC, 2003.

120. Kehtarnavaz, N., Gamadia, M., “Real-Time Image and Video Processing: From Research to Reality”, Morgan and Claypool Publishers, San Rafael, 2006.

Appendix

All code provided here have been tested on MATLAB® R2008a.

- A1 – OptimiseVerticesThreeMethods.m

```
%%%% INPUT%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% n_red, n_green:    Two vectors of data points x and y
% vertices:          Starting vertices fed into optimiser
% gamma:             Weight w of the cost function
% mode:              Choice of optimiser: random, nongradient, ga
% max_iterations: No. of iterations allowed for each optimisation cycle

%%%OUTPUT%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% vertices:          Optimised vertices obtained
% num_points_in:     No. of points enclosed by triangle
% area_triangle:     Area of triangle
% history:         To see how the cost function changes at each iteration

function [vertices, num_points_in, area_triangle, history] =
OptmimieVerticesThreeMethods(n_red, n_green, vertices, gamma, mode,
max_iterations)

num_points_in = NaN; %local variables CostFunction() results
area_triangle = NaN;
history = [];

switch lower(mode)

    case 'random'
        % Value to be added to vertices
        delta = 0.005;

        for index = 1 : max_iterations
            % Evaluate Cost Function
            [cost] = CostFunction(vertices);

            new_vertices = vertices + delta * randn( size(vertices) );
            [new_cost] = CostFunction(new_vertices);
            % For plotting
            history(index) = cost;

            if new_cost < cost %"<" for minimisation
                vertices = new_vertices;
                delta = 0.95 * delta; % allow step size to decrease
            else
                delta = 1.05 * delta;
            end
        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    case 'nongradient'
        options = optimset( 'TolFun',      1e-5,...
            'MaxIter',     max_iterations,...
            'FunValCheck', 'off',...
            'Display',     'off',...
            'OutputFcn',    @myobjplot...
            );
        history = [];
        [vertices, fval, exitflag, output] = fminsearch(@CostFunction,
vertices, options);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    case 'ga'

        population_size = 30;
        alpha = 0.0001;

        init_pop = alpha * randn(population_size,numel(vertices));
        Q1 = repmat(vertices',[population_size,1]);
        Q1 = Q1 + init_pop;

        options = gaoptimset( 'Generations', 20,...
            'Display',      'off',...
            'OutputFcns',     @mygaplot,...
            'TolFun',         1e-9,...
            'PopulationSize', population_size,...
            'InitialPopulation', Q1,...
            'CrossoverFcn', @crossoverintermediate,...
            'MutationFcn', {@mutationgaussian, 0.1}...
            );

        % Number of variables in objective function
        nvars = size(vertices,1);
        history = [];
        [vertices, fval] =
ga(@CostFunction,nvars,[],[],[],[],[],[],[],options);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end % end of switch-case

%-- Nested subfunctions

    function [cost_function_value] = CostFunction(vertices)

        area_triangle = TriangleArea (vertices); % sub-function
        % Find no. of points inside original triangle
        num_points_in = PointsIn(n_red,n_green,vertices);

 %%% CostFunc = Area + weight x (No. of points OUTSIDE triangle)
     cost_function_value= area_triangle + gamma * ( size(n_red,1) -
num_points_in );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end % end of CostFunction

    function stop = myobjplot(x,optimValues,state)
        if ~strcmp(state, 'iter') stop = false; return; end
        if isfield(optimValues,'fval')
            if isscalar(optimValues.fval)
                history = [history; optimValues.fval];
            else
                plotvector(optimValues.iteration,optimValues.fval);
            end
        else
            plotvector(optimValues.iteration,optimValues.residual);
        end
        stop=false;
    end % end of function myobjplot

    function [state, options,optchanged] =
mygaplot(options,state,flag,interval)
        %GAOUTPUTGEN Prints generations and best fitness value.
        optchanged = false;

        if nargin <4
            interval = 1;
        end
        if interval <= 0
            interval = 1;
        end

        if (rem(state.Generation,interval) ~=0)
            return;
        end

        best = min(state.Score);
        history = [history;best];

    end % end of mygaplot function

end % End of OptmimiseVerticesThreeMethods
```

- A2

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create objects of sizes less than a pixel,
% and investigate how the intensity of a particular pixel
% changes as the objects go through it.
% August 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear, close all
number_of_frames = 75;
high_res = 100; % High resolution image size
for object_size = 2:2:10
    image_plane    = zeros(high_res,high_res);
    % Distance in pixel moved by object per frame
    dx = 1;
    dy = 1;
    % Centre row of pixels being investigated
    centre_row = high_res/2;

    xmin = centre_row - object_size/2 ;
    xmax = xmin + object_size;
    ymax = 20;ymin = ymax - object_size;

 for k = 1:number_of_frames
    %Create matrix for a square object
    image_plane(uint8(xmin):uint8(xmax),uint8(ymin):uint8(ymax))=
255;
    % Define motion of square - horizontal motion only
    ymin = ymin + dy;
    ymax = ymax + dy;
    % Decrease resolution to 10 x 10 to achieve sub-pixel size
    for i = 1:10
        for j = 1:10
            small_image(i,j) = mean2(image_plane(10*(i-1)+ 1:10*i,
10*(j-1)+1:10*j));
        end
    end
```

```
        % re-initialise original image
        image_plane     = zeros(high_res,high_res);
        % pause(0.1)
        % Store the intensity values for graph plotting
        value_intensity(k) = small_image(5,5,1);

    end % end of numframes loop

        hold on
        plot(value_intensity)
        hold off
end
```

- A3

```
% kalman function
 function [state_vector, error_matrix]  = KalmanTrack (state_vector,
error_matrix, position_vector, mea_noise, pr_noise)
      %%%%%%%_______Kalman variables_______%%%%%%%%%
      dtx=1;
      dty=1;
      % State dynamical matrix A (or F)
       A = [1 0 dtx 0;
            0 1 0 dty;
            0 0 1  0;
            0 0 0  1];
       % Define state measurement matrix B (or H)
       B = [1 0 0 0 ;
            0 1 0 0 ];
       % Define size of the measurement noise(Ny)
       meas_noise = [ mea_noise^2 0;
                      0 mea_noise^2];
       % Define process noise (Nx )
   process_noise    = [ (dtx^3)/3     0          (dtx^2)/2         0    ;
                          0        (dtx^3)/3        0          (dtx^2)/2;
                        (dtx^2)/2     0            dtx             0    ;
```

```
                         0        (dtx^2)/2        0             dtx  ]
.*pr_noise ;
     %%%%%%%______End of Kalman variables______%%%%%%%%%
     % Start Kalman Filter
          % ---------------------
          % Step 1 - Predict expected measurement
          est_measurement = B*state_vector;
          % Step 2 - Actual actual measurement
          real_measurement = position_vector;
          % Step 3 - Calculate Kalman Gain (i.e. the weight attached
            %to current estimated state and to the new measurement).
          K = error_matrix*B'*inv(B*error_matrix*B'+meas_noise);
          % Step 4 - Use Kalman Gain to generate new state estimate
          state_vector=state_vector+K*(real_measurement-
est_measurement);
          % Step 5 - Use Kalman Gain to generate new state error ma-
trix
          error_matrix = (eye(size(error_matrix))-K*B)*error_matrix;
          % Step 6 - Use dynamical matrix to predict forward to next
time step.
          state_vector = A*state_vector;
          error_matrix = A*error_matrix*A'+process_noise;
    % End Kalman Filter

     end
     % end of kalman function
```