DOCTOR OF PHILOSOPHY

A Dynamic Stochastic Model for Real-Time Analysis and Predictions for Movement Exhibited by an Object

Cornelius, Ian

Award date:
2019

Awarding institution:
Coventry University

A Dynamic Stochastic Model for Real-Time Analysis and Predictions for Movement Exhibited by an Object

By

Ian Cornelius

July 2018

A thesis submitted in partial fulfilment of the University's requirements for the Degree of Doctor of Philosophy

Abstract

Predicting the movement of an object is a problem that has seen several solutions using stochastic modelling (i.e. Markov Models), artificial neural networks (i.e. feed-forward and recurrent networks) and Bayesian networks. The usual methodology of a Markov Model requires the use of a priori data to build a model that can predict the upcoming direction of movement. In this work, a model is presented that takes the underlying framework of a Markov model and makes an adaptation that will count the frequency of transitions that may occur between states. The collated information on the frequency of each transition will be used to adjust the probability vector of the Markov model to ensure that a prediction can be made without prior learning and historical data. The outcome of the amended model shows that a prediction accuracy can be obtained within the range of 79% to 96%. Whilst this is a high accuracy, the rate is dependant upon the type of movement that is exhibited by the object and number of previous movements that are used, otherwise known as a models order. Adjusting the order of a model will takes into account a collection of previous observed movement to make a prediction. For example, a first order model will take into account a single previous direction, whereas a third order model would take into account the last three directions.

To understand how well the amended Markov model performs in comparison to a traditional model and artificial neural networks. A comparative study is performed on a traditional Markov model and two neural networks. A feed-forward neural network has been used to determine how a simple network can be used to generate a prediction on the next direction of movement whilst a more complex recurrent neural network is used to determine how a 'memory-like' state (akin to the Markov model) can increase the accuracy of the predictions generated. To ensure that the networks follow the same basis as the stochastic model, the networks are designed in a manner to feed the past n movements observed to train the neural network and make a prediction. The results for the feed-forward network shows that the recorded accuracy can vary between 69.43% and 78.50%, whilst the recurrent neural network averaged between 67.06% and 76.27%. A linear progression is seen for the accuracy of the recurrent neural network, with the number of past movements that are used to make a prediction influencing the accuracy.

Comparing the two models, it can be seen that the stochastic model has an advantage over the artificial neural network when it comes to the processing times that has been observed for the completion of video or text file. With the accuracy rates ranging higher for the amended stochastic model and falling within a favourable computation time. The Markov model would be best suited for problems that rely upon predictions being generated within a real-time manner. The work also covers the prospect of recognising patterns within the matrices to determine whether a similarity can be found between the different paths of movement that have been exhibited by an object or pedestrian. The method is applied to the probabilities of the stochastic model and it can be seen that by applying the scoring functions by Haralick (1979), the dissimilarity, entropy and homogeneity scores can describe the path an object or pedestrian has travelled within a scene and can give an insight to the direction of movement. This information can be used to build a model database that can be used when the algorithm is running to select an appropriate model that would generate a prediction with a higher accuracy.

Contents

Abbreviations

BF BruteForce. 42, 63–65

BRISK Binary Robust Invariant Scaleable Keypoints. 7, 12, 15, 20, 24–26, 28, 30, 37, 38, 40–43, 45, 46, 59–63, 65, 68–75, 126–130, 133, 134

BRVO Bayesian Reciprocal Velocity Obstacles. 32, 33

CNN Convolution Neural Network. 30–32

DMM Dynamic Markov Model. 49, 50, 56, 134

DOG Difference of Gaussian. 23, 28

EWMA Exponentially Moving Weighted Average. 44, 128

EXIF Exchangeable Image File Format. 40

FFNN Feed-Forward Neural Network. 47

FLANN Fast Library for Approximate Nearest Neighbours. 42, 63, 64, 69, 73

fps frames-per-second. 45

GLMP Global and Local Movement Pattern. 33

GLOH Gradient Location and Orientation Histogram. 27, 28

ISS International Space Station. 26, 38

KLT Kanade–Lucas–Tomasi. 11, 30, 31

KNN k-nearest neighbours. 42

LSTM Long Short Term Memory. 11, 31, 54

MC Markov Chain. 47, 50

MSE Mean Square Error. 11, 30, 31

MV machine vision. 16

NN Neural Networks. 20, 30, 38, 47, 52

NNDR Nearest-Neighbour Distance Ratio. 27, 28

OpenCV Open Source Computer Vision Library. 41, 43, 46, 123, 125

ORB Oriented FAST and Rotated BRIEF. 28

PPM pixel-per-metric. 43

RNN Recurrent Neural Networks. 31, 47

RVO Reciprocal Velocity Obstacles. 32

SIFT Scale Invariant Feature Transform. 20, 22–28, 37, 38, 40–42, 44, 46, 59–65, 67–71, 129, 130, 133

SURF Speeded-Up Robust Features. 12, 20, 23–28, 37, 38, 41–43, 45, 46, 59–65, 68, 70–74, 126, 130, 133, 134

SVM Support Vector Machine. 31

UHD Ultra High Definition. 134, 135

List of Figures

List of Tables

Chapter 1

Introduction

An increase in demand for applications that can interact with the environment, has seen machine vision (MV) become an important issue. The introduction of cameras on mobile hardware (such as phones and tablets), and personal computers has seen an increase in the development of MV systems in the past few decades/years. MV is defined as system that includes that technology and methodology to extract key components (or information) from an image or video. The information that is extracted from such a process can be used for tasks such as identification and tracking of objects, the guidance of robots or the ability to create a model of the real-world from a collection of images (Jain et al. 1995). The MV topic area has multiple related fields such as image processing, computer graphics and pattern recognition; the various fields can be applied to the machine vision topic to assist in the identification of an object from its environment, i.e. the resizing of an image or adjusting the colour properties.

The area of machine vision has been explored and studied by Hariyono et al. (2014) and Koppula et al. (2013). The work undertaken was concerned with the identification of a pedestrians location and the learning of their movements and various activities. The process of identification for an object within a scene is often only the first step in a more complex machine vision application. With each turn of a decade, the price of technology falls and becomes widely available to the public. With this, the inclusion of on-board cameras are becoming increasingly popular in vehicles. It was only a decade ago that this inclusion was rare and seen as a novelty. However, with the growing popularity of in-vehicular systems consisting of an operating system it leads to an increase of premium features available.

Vehicle brands such as Alfa Romeo and Fiat have used the Windows Embedded operating system in their vehicles (Microsoft.com 2006), whilst BWM use the Apple CarPlay operating system (Kahn 2016). The popularity of operating systems in vehicles is growing, and the inclusion of these operating systems can assist in the additional growing functionality being added; i.e. the use of on-board cameras for assistance in rear-parking or lane guidance. However, an omission of usage is the ability to detect a pedestrian and observe their movements for vehicle-pedestrian safety. By analysing the movement and possible locations of a pedestrian from the environment it could theoretically decrease the risk of a vehicle-pedestrian incident.

Today, it is possible to create vision systems that analyse a changing visual field rapidly enough to identify objects in real-time, tracking their location and even predicting their movements. Predicting the movement of an object is a problem that has seen a number of proposed potential solutions, many of which make use of stochastic models to analyse stochastic features of historical data. Other works describe the use of Hidden Markov models to incorporate the effects of latent variables not directly expressed in a collection of historical statistics. The field of study within this thesis is concerned with the identification and tracking of an objects movement to assist in the prediction of the immediate next movement. This process is examined as both a general problem and

in the context of a specific application. The ideology of the work is to determine the best method that can be used to identify and track an object within a real-time constraint and a methodology that can be used to predict an object's movement within this constraint. The body of work that will underline this thesis will include experimentation with with different feature detectors to determine the best algorithm for recognising an object from its environment, along with the experiments on a Markovian model and artificial neural networks to generate these predictions.

1.1 Aims and Objectives

The ability to predict the movement and future location of object within literature share a common practice, the use of a priori data. This form of data originates from pre-selected data from previous experiments to provide a level of machine learning/training. Building a model or neural network that is based upon this method can influence the predictions generated. It is often seen that this pre-training method is expensive and can take valuable computational resources.

In most cases, this type of training is not readily available or cannot be used for systems that are concerned with running in a real-time constraint. Therefore, a new type of methodology is required for training 'on-the-fly', and the aim of this research is to develop a new method of learning for this purpose. The following questions have been devised to determine whether the purpose of this study can be met:

1. Is it feasible to recognise an object in a real-time constraint on mobile hardware?
2. Based upon the recognition process, is it also feasible to generate predictions in a real-time constraint?
3. Based upon the observations of an objects movement, is it possible to determine patterns from the data collected?
4. Is it possible to use patterns of movement exhibited by tracked objects to incorporate prior learning by selecting them from a set of exemplars, and would this increase the accuracy of predictions more quickly with no a priori data?

The answers to these questions will be supplied in this study to provide a novel mechanism for the task of recognising and predicting the movement of an object in a real-time constraint. Several objectives have been outlined to ensure that the work underpinning this study is met:

- **Real-Time Object Recognition**: An object is able to be recognised from its environment within a real-time constraint with a focus upon mobile hardware.
- **Markov Model Adaptation**: The Markov model is adapted to provide a learning mechanism that can be done whilst the model is running.
- **Real-Time Predictions**: Predictions are generated within a real-time constraint without a reliance upon prior training.
- **Pattern Analysis of the Observed Movements**: The patterns formed within the probability vector of the Markov model are analysed to determine whether a particular type of movement can be observed.

Meeting each of the objectives should provide the underlying working of this thesis. The ability to recognise and track an object as its moving within its environment and be able to make a robust prediction on its next intended direction of movement.

1.2 Scope and Contribution

The overarching aim of this study is the ability to recognise an object and predict its intended direction of movement. The recognition of an object is not entirely new within literature, but the ability to recognise an object on mobile hardware is becoming increasingly popular. With this method of being able to track an object using mobile hardware, or on embedded systems it opens the prospects of being able to predict the movement and trajectory of an object. Therefore, as it stand the following are the contributions that this thesis will bring:

- **Hybrid Feature Detection**: a method has been devised to use two feature detectors to formalise a descriptor to ensure that a robust and real-time recognition can be performed on a mobile device.
- **Adapted Markov Model**: a traditional Markov model is adapted to provide a learning mechanism that can be achieved in real-time.
- **Pattern Analysis of a Markov Models Vector**: the probability vector of the Markov model could contain information regarding the trajectory of an object. Therefore, an analysis of these vectors could indicate a type of movement exhibited by an object.

Although the work undertaken in this thesis is pertaining to mobile hardware, it is theoretically expected that the work could be undertaken on in-vehicular systems. However, due to the limitation of owning a vehicle with the necessary requirements, this is beyond the remit and scope of this thesis.

1.3 Structure of the Thesis

The structure of the thesis has been split into four chapters: literature review, methodology, results analysis and conclusion. An overview of each chapter is provided in their respective sections.

1.3.1 Chapter 2: Literature Review

The literature review chapter will cover the relevant literature that is pertaining to the work undertaken in the thesis. The chapter is split into four sections, detecting features of interest, the process of recognising an object using machines, different methodologies for predicting an object's trajectory and a conclusive evaluation of the literature reviewed and whether there is any missing work that can be undertaken in this thesis.

1.3.2 Chapter 3: Methodology

The methodology chapter covers the various processes and methods that are followed to ensure that a robust recognition is performed in real-time on a collection of images and videos for a mobile device. Another section is included that covers the two methods that have been chosen for implementation to generate predictions. The primary method of choice is a stochastic model that is adapted to perform within real-time and this adaptation is discussed in detail. The secondary method is a neural network that is included to provide a comparative study on how a more complex model is able to generate predictions.

1.3.3 Chapter 4: Results Analysis

The analysis of results will cover the work undertaken for recognising an object in real-time on a mobile device and personal laptop computer. The section will look at the various experiments performed on several datasets that include low and high resolution images and sequential data, i.e. videos and web camera feed. The chapter will also cover the results of the Markov model and neural networks that have been implemented to generate predictions of an objects next direction of movement. The various models and networks have been experimented on two datasets, one pertaining a collection of text-files with pedestrian locations and a secondary dataset of videos that were used in the object recognition experiment.

1.3.4 Chapter 5: Conclusion

The conclusion will provide a conclusive evaluation of the entire study and whether the aims and objectives outlined in this chapter have been met. The chapter will also include a summary of further work that may be undertaken from the basis of work that has been supplied in this thesis. Finally, a decision will be made on whether the original hypothesis has been met.

Chapter 2

Literature Review

The use of feature detectors are prevalent in literature and machine vision tasks to assist in objectives such as recognising an object or handwriting. With each advent of a new detector the methodology they follow alters providing a new set of advantages and disadvantages. This chapter will cover a selection of traditional and binary feature detectors that are used in literature for recognising an object from its environment. Three detectors have been chosen due to their prevalence within literature: Scale Invariant Feature Transform (SIFT) by Lowe (1999, 2004), SURF by Bay et al. (2008) and BRISK by Leutenegger et al. (2011). Work has been undertaken by Coltin et al. (2016) using a combination of two detectors to form a map and is discussed in this chapter. Using a combination of two detectors could provide a methodology that enables the recognition phase to be performed within a real-time constraint on mobile devices.

The chapter will also cover a selection of papers that have predicted the trajectory of an object using a variety of different methods. Methods that have been covered include the use of Neural Networks (NN), Bayesian Networks and a stochastic model known as a Markov Model. The reviewed literature on these methods will provide an understanding of how the models and networks perform. The applicability of these methods for the process of generating a prediction within a real-time constraint are taken into consideration when reviewing the papers. The optimum model for generating predictions in a real-time constraint will then be chosen to be implemented as part of the study.

2.1 Detection of Features for Object Recognition

The terminology 'feature detector' is used in machine vision literature to describe the process of localising key areas of interest from an image or video. The extracted data from this process is often referred to as 'keypoints' and contains information that is pertaining to a pixel of interest and are repeatable. Ensuring that a keypoint is repeatable means that no matter how an image may be transformed, the keypoint is found within the same location. There are various methodologies that can be used to detect these keypoints of interest and depend upon the type of feature detector used:

- **Corner Detection**: this type of method detects keypoints that are localised upon the corners found in the images content. A corner is defined as being an intersection where two edges meet, or the point where two dominant and different edge directions meet. The corner is classified as keypoint when it comes well defined and can be robustly detected from an image.
- **Edge Detection**: this type of method simplifies the contents of an image to a collection of

edges (Jain et al. 1995). There are two types of edges that can be formed based upon the discontinuities of an image: step and line. A step discontinuity occurs when the intensity changes from one value to another; whilst a line discontinuity occurs when the intensity value changes and returns to its original value within a short distance.

The various methods of detecting keypoints from an images does not impact the overall aim of the process; the ability to localise key areas of interest. A trade-off being may occur, whereby the overall robustness of a method may be sacrificed for the amount of processing time. The detected keypoints are used to formalise a feature vector, otherwise known as a descriptor. It is representative of the various properties that are underlying within an image and are classified into two categories:

- **Global**: represents the features of an image in a one-dimensional format that consists of the images colour, texture and shape properties.
- **Local**: represents the features of an image that are based upon the regions of importance in an image that are invariant to changes within the viewpoint or illumination.

Dependant upon the type of feature detector that is used, the extracted descriptors can have an invariance to various image transformations that may occur, i.e. image skewing and rotation. Awad & Hassaballah (2016) discussed various characteristics that a descriptor must meet to ensure their robustness. The descriptors must be repeatable, accurate and generalised efficiently in an abundance to form a descriptor that provides a compact representation of the image. There are two types of descriptors that can be extracted and are classified as the following:

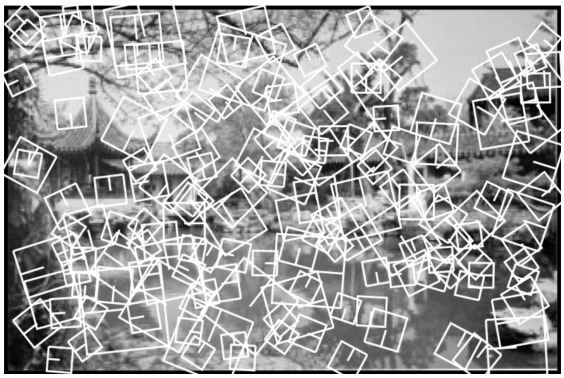
- **Traditional Descriptors** are highly dimensional and a match between them is made by computing the Euclidean distance.
- **Binary Descriptors**: these type of descriptors are represented in the form of a binary string and are matched by the computation of the Hamming distance between a pair of descriptors.

The extracted descriptors from a set of images are commonly used for the task of object recognition. The term 'object recognition' is attributed to the process of being able to pick out and identify an object from its environment. Since the evolution of primates to human beings, the ability to pinpoint objects and recognise them has been available. However, this process of recognition for humans and primates uses various areas of the brain. The dorsal and ventral streams are used to perceive an object and provide the ability for us to recognise it from within its environment. Each stream has a particular role in the visual system, the ventral stream is used for the recognition of an object and its representation within an environment. The stream can be influenced by extra-retinal factors such as attention, working memory and stimuli. Therefore, the ventral stream is able to provide a description of an object and its overall significance within an environment. The dorsal stream is involved with the task of recognising an object and the space it occupies within the environment. It consists of a detailed map of the visual space and is best for detecting and analysing the movement of an object (Bear et al. 2007). Work by DiCarlo et al. (2012) discuss that the dorsal stream is most significant for the guidance of the eyes towards an object and defined the term 'object recognition' as a process of assigning labels to an object that can be used for identification and/or categorisation.

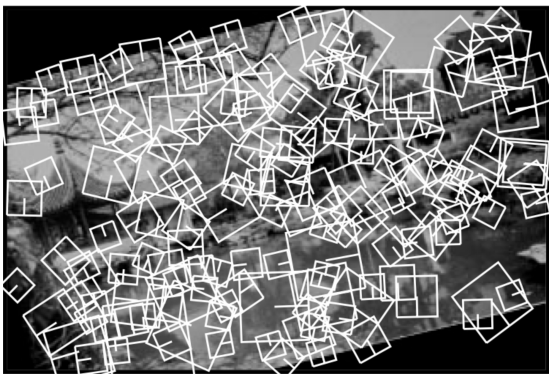
2.2 Feature Detectors for Machine Vision Recognition

The process of recognising an object using machines requires a higher level of understanding and training in order to reach such an achievement. The process typically relies upon the use of a

training source to find an object within a scene, the environment. The method of recognising an object is typically done using a feature detector to detect keypoints of interest which are in-turn formalised into a descriptor. There are various detectors that are available to assist in the formation of these descriptors. A popular detector used within literature is the SIFT detector by Lowe (1999, 2004). The detector is designed to extract a descriptor from an image that is invariant to a variety of image transformations that may occur, i.e. scaling, translations and rotations, and illumination changes. The work by Lowe (1999) used an image that consisted of little or no transformations (image **A**). The image was then subjected to various geometric transformations (image **B**) and were then subjected to their keypoint detection method. Lowe then used a matching algorithm to determine whether a match can be made between a pair of keypoints between image **A** and **B** to determine whether the keypoints were repeatable. An example of an image used during their test is shown in Figure 2.1.



(a) An example of an image used in the test showing a subset of the keypoints detected.



(b) An example of a manipulated image under various image transformations showing a subset of the keypoints detected.

Figure 2.1: Two images used by Lowe (1999) in his experimentation to determine the stability of his feature detector.

The outcome of their matching process gained a retention rate of 78% between the two images, when the following transformations were applied: increase of contrast, decrease of intensity, rotation, scaling, stretching and additional noise. This meant that 78% of the keypoints from image **A** were detected again in image **B** and further results included by Lowe also provided information on how inflicting the image to an individual transformation could affect the repeatability score and is shown in Table 2.1. The first column of the results show the percentage of keypoints that have been matched between the image **A** and **B**, whilst the second column shows the percent of keypoints with the same orientation that have been matched.

Table 2.1: Adjusting the image with various transformations can affect the stability of the detector. The table shows how each transformation can affect the retention rate between the original and transformed image. Table data acquired from the work by Lowe (1999).

Image Transformation	**Match** %	**Orientation** %
Increase contrast by 1.2	89.0	86.6
Decrease intensity by 0.2	88.5	85.9
Rotate by 20 degrees	85.4	81.0
Scale by 0.7	85.1	80.3
Stretch by 1.2	83.5	76.1
Stretch by 1.5	77.7	65.0
Add 10% pixel noise	90.3	88.4
All of the above	78.6	71.8

From the results, it can be seen that solely by adjusting the noise of an image by 10% it has a small affect on the detectors repeatability with a small reduction in the number of keypoints that

were detected, a drop of 9.7%. The second best result was obtained when adjusting the contrast of the image by a factor of 1.2, whereby 89% of keypoints were retained in the transformed image. It can be seen that the SIFT detector is robust with a high stability on the repeatability of the detector when images transformations have been applied to the image. With a large number of keypoints detected and the stability of SIFT, it can reinforce the suitability of using the detector for recognising an object and it can be seen why this is a popularly used detector within literature.

A key disadvantage of SIFT is the large computational cost of the detector. Therefore, over the years numerous derivatives of SIFT have been developed whilst encompassing the key principals. A derivative known as the SURF was introduced by Bay et al. (2008). The authors expanded upon the work by Lowe (2004) and provided an alternative method that was faster for extracting descriptors from an image. The new method used a method known as Hessian-matrix approximation (Mikolajczyk & Schmid 2001) and relied upon the use of integral images. An integral image is a derivative of the summed-area table (Crow 1984) and was introduced in the works by Viola & Jones (2001). The new method by Bay et al. was tested using a collection of images and software by Mikolajczyk & Schmid (2004, 2005). The images consisted of real-textured and structured scenes with various transformations applied, i.e. rotation and zooming, illumination changes, and viewpoint adjustments. The success factor of their detector was determined by a repeatability score and indicates how many of the keypoints are relative to the total number that were originally detected. Bay et al. tested two versions of their detector with adjustments to the Gaussian derivative filter size. Two filter sizes were used, 9x9 and 15x5 and are referred to as FH-9 and FH-15, respectively; and results are shown in the graph in Figure 2.2.

Figure 2.2: The repeatability score for the wall and graffiti images, respectively. The images consisted of transformations made to the viewpoint. The graph has been acquired from the work of Bay et al. (2008).

A negative linear projection can be seen when the viewpoint angle increases for the wall sequence of images with the repeatability score decreasing. A similar pattern can be seen with the sequence of images containing the graffiti. When an angle greater than 40 degrees is present, the detector is unable to match keypoints between the original image and transformed image. The reduction in the repeatability score is most likely due to the complex nature of the image used (graffiti sequence) and the differing structures that are present within the images. However, it can be seen that for a 20 degree viewpoint change, the a similar level of repeatability can be achieved as that of the works by Lowe (2004). The data in Table 2.2 shows the processing time taken for detecting keypoints using SURF and compared to SIFT, referred to as Difference of Gaussian (DOG).

Table 2.2: The threshold of detection, number of keypoints detected and calculation time for the detectors used in the comparison by Bay et al. (2008), for the first image of the graffiti scene.

Detector	**Threshold**	**Number of Keypoints**	**Computation Time** (ms)
FH-15	60 000	1 813	160
FH-9	50 000	1 411	70
Hessian-Laplace	1 000	1 979	700
Harris-Laplace	2 500	1 664	2 100
DoG	$Default$	1 520	400

From the results of the authors' work, it can be seen that their FH-15 detector achieved the best number of keypoints within 0.16 seconds, which was significantly faster than that achieved by SIFT which detected $1,520$ keypoints in 0.40 seconds. The authors discuss that the number of keypoints between the detectors are all very similar in range. However, the threshold rate for each detector were applied to ensure a similar number of keypoints were detected to that of SIFT. The best result for computation was achieved when a 9x9 filter was used, FH-9. It can be seen that although a fewer number of keypoints were detected the amount of time taken was recorded to be significantly lower (0.07 seconds); an overall drop of 0.33 seconds, a reduction of 140.42%.

The SURF and SIFT detectors have a similar method of formalising their descriptors as a traditional detector. The matching method used between the descriptors is computed by the Euclidean distance and these type of detectors are often seen to be computationally expensive. The work by Bekele et al. (2013) highlighted that the likes of a traditional descriptor are too computationally expensive to be considered for applications where processing is to be computed within real-time. Therefore, work by Leutenegger et al. (2011) have introduced new types of detectors that extract descriptors in the form of a binary string that can be easily matched by th Hamming distance. Their detector, known as BRISK, uses a unique sampling pattern to detect keypoints of interest from an image, shown in Figure 2.3.

Figure 2.3: The sampling pattern used by the BRISK detector to determine a pixel as a keypoint of interest from an image. Image acquired from the works of Leutenegger et al. (2011).

The sampling pattern consists of 60 location points (the blue circles) which consist of larger radii (the red dashes) for sampling. A mask is used to determine whether a minimum of nine pixels within a sixteen pixel radius are continuously bright or darker to that of the central pixel, known as 9-16 mask. This results in far fewer sampling points used in comparison to the amount

of pairwise comparisons and therefore the overall complexity of searching the intensity values are far fewer. Leutenegger et al. held a comparative study of their detector with SIFT and SURF, results are shown in Table 2.3.

Table 2.3: The results of the comparison experiment performed by Leutenegger et al. (2011). The table of results is acquired from their body of work.

	SIFT	**SURF**	**BRISK**
Detection Threshold	4.4	457000	67
Number of Points	1851	1557	1051
Detection Time (ms)	1611	107.9	17.20
Description Time (ms)	9784	559.1	22.08
Total Time (ms)	11395	667.0	22.08
Time per point (ms)	6.156	0.4284	0.03737

BRISK is able to detect $1,051$ keypoints from an image in approx. 0.017 seconds, whilst SIFT and SURF were able to detect $1,851$ and $1,577$ in 1.611 and 0.108 seconds, respectively. Although the traditional detectors were able to detect a larger number of keypoints, the amount of time taken was far more significant; with SIFT taking greater than a second. Overall, BRISKis able to detect a moderate number of keypoints within a significantly less amount of time due to the fewer number of sampling points. Due to this methodology, BRISK can be seen as being advantageous for applications that are met with a real-time constraint, or systems with a finite amount of memory (i.e. mobile-hardware or embedded systems). However, BRISK is lacking in the area that it only considers sampling pixels within a two-dimensional axis of an image or video. The detector does not take into account the depth that may be exhibited in an image.

Karpushin et al. (2015) looked at improving the distinctiveness of BRISK by taking into consideration the depth axis, z. The authors' incorporate a depth map that is used to consider the out-of-plane rotations that the normal sampling pattern of BRISK is unable to handle. There has been previous work by authors like Mikolajczyk & Schmid (2004) that have explored the area of handling rotations using the BRISK detector, but were limited to a degree of rotation of up-to forty degrees. The new method proposed by Karpushin et al. (2015) distributes the sampling points of BRISK over the surface of an object, as shown in Figure 2.4.

Figure 2.4: The distribution of sampling points onto the object as described by Karpushin et al. (2015).

Upon the distribution of the sampling points, the original method by Leutenegger et al. is used to detect keypoints of interest from the image. Upon detection of the keypoints, Karpushin et al. compute the local parametrisation at each keypoint. For example, the authors look for the radial and angular co-ordinate of each pixel that are intrinsic to the scenes surface. This results in a descriptor that is extracted based upon the objects' texture and is mapped to the image intensity of the objects' geometry. To test the feasibility of their method, the authors use a synthetic texture and depth dataset. The dataset consists of images with significant out-of-plane rotations. The graph shown in Figure 2.5 shows the results of their detector on a sequence of images with graffiti.

Figure 2.5: The matching score for the graffiti sequence of images used in the experiment. The graph has been acquired from the works of Karpushin et al. (2015).

It can be seen in the graph that the matching score of the authors depth map is significantly greater than that the original BRISK detector. It can be seen that a score of approximately 75% was achieved for the depth map when an angle of view difference was zero. A negative linear progression can be seen when the angle of view increases up-to 60 degrees, of which from there on it begins to increase. The same pattern can be seen with the BRISK detector, but it becomes significantly worse between 20 and 80 degree. BRISK begins to increase positively on the matching score, but there is still a significant delta between the two methods (approx. 20%). Karpushin et al. (2015) conclude that their results show that the depth-map enables BRISK to detect objects within the bounds of 120 degrees for a the sequence of graffiti images. The matching score of their work was approximately 45% for a collection of images used in their research. It can be seen that with the addition of a depth map to BRISK, the robustness of detection can significantly increase. Whilst it can also been seen that the major advantage to BRISK is the low overheads that can be gained by using a binary detector. Not only are the descriptors extracted faster than the likes of a traditional detectors (SIFT and SURF) but the matching time between the binary descriptors are also inherently faster. With this gain in time saved for extracting and matching of descriptors can make BRISK a worthwhile detector for applications that require to be computed within a real-time constraint and have have hardware limitations.

However, the combination of a traditional and binary feature detector could provide a hybrid detector. The benefits of such a detector could provide the robustness of a traditional descriptor with the fast matching of the binary descriptors. Such hybrid models have been discussed in the work by Coltin et al. (2016); whom proposed the use of a combined methodology of SURF and BRISK. The combined method was used to aid robots used in the International Space Station (ISS) to conduct a variety of experiment by helping their placement around the station. The processing power of the robots were constrained, and therefore a method of extracting features from images captured using the onboard cameras within a timely manner was essential. The robots are able to localise themselves around the ISS by using a visual feature map that is precomputed offline on a system using the SURF detector. The feature map is a database that consists of various descriptors and three-dimensional pointers. The robot uses these maps to compare the descriptor extracted onboard to move itself around the ISS. The uniqueness of this work is the authors methodology of using BRISK to rebuild the map when a new feature has been detected. By combining the two feature detectors together, the authors were able to add to the robust feature collection by SURF to rebuild the map of the ISS. However, Coltin et al. recognises that the BRISK detector is not as accurate and robust in comparison to SURF. The low computational cost of BRISK makes it an ideal detector to be used for processing on hardware with constraints in place due to external factors. In the next section, literature will be reviewed to determine how the process of recognising an object by humans and machines are undertaken; and the amount of time that it takes for a human and/or machine to perceive an object from its environment.

2.2.1 Speed of Object Recognition by Humans and Machine

The term 'object recognition' is attributed to the process of being able to pick out and identify an object from its environment. Since the evolution of primates to human beings, the ability to pinpoint objects and recognise them has been available. However, this process of recognition for humans and primates uses various areas of the brain. The dorsal and ventral streams are used to perceive an object and provide the ability for us to recognise it from within its environment. Each stream has a particular role in the visual system, the ventral stream is used for the recognition of an object and its representation within an environment. The stream can be influenced by extra-retinal factors such as attention, working memory and stimuli. Therefore, the ventral stream is able to provide a description of an object and its overall significance within an environment. The dorsal stream is involved with the task of recognising an object and the space it occupies within the environment. It consists of a detailed map of the visual space and is best for detecting and analysing the movement of an object (Bear et al. 2007). Work by DiCarlo et al. (2012) discuss that the dorsal stream is most significant for the guidance of the eyes towards an object and defined the term 'object recognition' as a process of assigning labels to an object that can be used for identification and/or categorisation.

Machines' require a further level of training in order to complete a similar process by human beings that happens subconsciously. There have been several studies performed upon humans and primates to measure the amount of time it takes for them to recognise an object. It can be seen in the work by Thorpe et al. (1996) that humans are able to recognise an object between 0.38 and 0.57 seconds. Latter work by Fabre-Thorpe et al. (1998) have shown that humans are able to recognise objects within a similar time of 0.35 seconds, but their work also included an experiment on primates. From their work, it can be seen that primates are able to recognise an object in 0.25 seconds, which is approximately 0.10 seconds faster than a human. It has been discussed in the work by DiCarlo et al. (2012) that by removing the behavioural time of a human or primate to acknowledge the recognition of an object the though-process behind it can be as little as 0.20 seconds. In the work by Bay et al. (2008), it can be seen that the SURF detector (labelled as FH-9) was able to detect keypoints of interest within 0.07 seconds. By factoring in the rest of the process, i.e. extracting and matching the descriptors, work by Chen et al. (2015) has shown that the recognition of an object can be performed in as little as 0.16 seconds. The performance of feature detectors for machines have been critiqued and evaluated by a number of researchers over the years.

The work by Mikolajczyk & Schmid (2005) studied various feature detectors on a collection of different image datasets. Each dataset consisted of various image transformations that ranged from plane rotations, scale adjustment or Gaussian filtering. The criteria of evaluation for the authors' work was based upon the number of positive matches made between an image pair. Three different matching strategies were used:

1. **Threshold Application**: a match is found between a set of pairs when the distance between the two descriptors fall within a pre-defined threshold.
2. **Nearest-Neighbour**: a match is found between the descriptors if descriptor **B** is the nearest neighbour to descriptor **A**. The distance between these two descriptors then becomes a threshold.
3. **Nearest-Neighbour Distance Ratio (NNDR)**: the method proposed to the author is similar to the nearest-neighbour method. However, a threshold is applied to the distance ratio between descriptors **A** and **B**'s nearest neighbour.

The experiment involved matching 400 features between the collection of images. Mikolajczyk & Schmid (2005) results show that their implementation of a feature detector, Gradient Location and Orientation Histogram (GLOH), gained the best results out of the several detectors featured within their experimentation. SIFT in comparison, performed reasonably well when using the

NNDR matching strategy. The detector was able to match 177 of the features out of 400, with GLOH only slightly beating SIFT with 192 matches. Mikolajczyk & Schmid discuss that SIFT was marginally worse off due to the poor performance on textured images and in images where the edges of an image were unreliably distinguished.

A further study was held by Miksik & Mikolajczyk (2012) where an evaluation was performed upon the effectiveness of BRISK, SIFT and SURF. The work compared the binary and traditional descriptors by experimenting with the extraction and detection of features amongst various images. The main purpose of the study was to determine whether the use of binary descriptors were more efficient in comparison to the traditional methodology. The outcome of the experiments were measured using a repeatability, precision recall and speed-up scoring metric. The results of their work can be seen in Table 2.4.

Table 2.4: The results of the effectiveness study by Miksik & Mikolajczyk (2012), and shows the average computation time of the various detectors used during the experimentation.

Feature Detector	**Run-Time** (ms)	**Speed-up**	**# of Detected Keypoints**
SURF	176	1.9	2911
DoG	338	1.0	1552
FAST	2	169.0	5158
STAR	17	19.9	849
MSER	60	5.6	483
BRISK	10	33.8	1874
ORB	7	48.3	594

Similar to the work of Bay et al. (2008), the SIFT detector is referred to as DOG and it can be seen that BRISK is considerably faster than that of SIFT with a similar amount of keypoints detected. However, it can be seen that FAST by Rosten & Drummond (2005, 2006) is the best performing detector with the ability to extract $5,158$ keypoints in 0.002 seconds. This is significantly faster than SIFT and marginally faster than BRISK. However, the work by Bay et al. is as expected faster than SIFT with it able to compute in almost half the amount of time. Miksik & Mikolajczyk (2012) conclude that the binary detectors (BRISK, FAST and Oriented FAST and Rotated BRIEF (ORB) etc.) are inherently faster than that of the traditional detectors due to the simple binary test between a pair of descriptors. The performance gain seen in the results can be a positive when considering applications that require processing to be done in a real-time constraint.

The robustness of SIFT was studied in prior work by the author whereby an evaluation was performed on the detector using various mobile devices ranging from the budget-friendly to high-end models (Cornelius 2014). The study involved using a collection of images used by Lowe (2004) in their experimentation. The images are shown in Figure 2.6 and the resolution of the images that were used are low when compared to the resolution of images that can be obtained on the mobile devices onboard camera.

(a) Basmati **(b)** Book **(c)** Box **(d)** Scene

Figure 2.6: The collection of images used by Lowe (2004) whilst evaluating his detector to recognise an one of the three objects from the scene image. Images acquired from the work by Lowe (2004).

The minima and maxima of the resolutions used were 265-by-175 and 512-by-384, a total of 45,500 and 196,600 pixels, respectively. When compared to the resolution of images that can be captured on these mobile devices (3264-by-2448, a total of 7,990,272 pixels) these are significantly of lower quality. The methodology of the study involved matching the extracted descriptors from the training and scene image. The study emphasised on the time taken for extraction and matching and whether it was feasible to to be done in a real-time applicability. The specification of the mobile devices and personal computer used for the experimentation are shown in Table 2.5.

Table 2.5: The specification of the mobile devices and Lenovo laptop that were used for the experiment in the study performed by Cornelius (2014).

Device	**Processor**	**Clock-speed**	**RAM**
Archos Titanium 40	MediaTek MT6572	1.3 GHz (Dual-Core)	512 MB
Google Nexus 7 (1st Gen.)	Nvidia Tegra 3	1.2 GHz (Quad-Core)	1 GB
Sony Xperia E	Qualcomm MSM7227A	1.0 GHz (Single-Core)	512 MB
Sony Xperia Z Ultra	Qualcomm MSM8974	2.2 GHz (Quad-Core)	2 GB
Lenovo ThinkPad T420	Intel i5-2520M	2.5 GHz (Dual-Core)	4 GB

The results of the mobile devices were compared to the Lenovo laptop to determine how the differing processing times and computation power between the devices can inflict the applicability of recognising an object in real-time. The experiment was designed to use the images from the work undertaken by Lowe (2004) to determine how long it takes to detect keypoints of interest and to formalise a descriptor. It was purposely chosen to use these images to see how the advancement of technology can impact the processing time of extracting descriptors and recognising the object within the scene image.

A selection of the results from the study can be seen in Cornelius, Table 13, Page 75. The results show the Lenovo laptop performed the best in comparison to the mobile devices. This was to be expected due to the abundance of power that can be found within a laptops process when compared to the processing power of a mobile device. For example, the Google Nexus 7 detected the keypoints for the three images in 0.47 seconds, an average that has been taken across the three object images, whilst the Lenovo laptop was able to detect the keypoints in 0.04 seconds (a tenth of the time taken by the mobile device). The method used for extracting descriptors saw an opposite set of results, with the Lenovo laptop extracting a descriptor in 0.88 seconds, whilst the Google device extracted the descriptors within 0.74 seconds.

A positive progression can be seen on the results for the scene image. The Lenovo laptop was able to extract a descriptor for this image within 0.11 seconds, and the Google device extracted a descriptor in 1.72 seconds. These range of results are more in-line with the general consensus of what is to be expected, with the expectation being that the laptop should extract a descriptor faster than the mobile device. The conclusion of the previous work was that the mobile devices at the time of publication were not applicable for the task of recognising an object within real-time due to the large overheads required for the extraction of the descriptors. However, with the linear direction seen in the improvement of mobile device hardware, such as the processing power, over the last few years since the previous study it may become feasible that mobile devices can now perform these tasks.

Based upon the studies by Miksik & Mikolajczyk (2012) and Mikolajczyk & Schmid (2005), it can be seen that the detection methods of machine vision algorithms are similar (or quicker) in regards to extracting key features from an image. In the work by Leutenegger et al. (2011) it can be seen that the total amount of time taken for recognising an image from a scene is 0.02208 seconds for their algorithm. This process of recognition undertaken by a machine is significantly faster than that of a human with the average time of recognition being 0.35 seconds (Thorpe et al. 1996). The difference between the human and machine is 0.328 seconds and would have an impact in the time taken for applying the brakes in a vehicle autonomously when compared to a manual application. The difference between the time of a human and the machine could be the difference between a collision occurring or not. With the incorporation of the work previously undertaken, it can be

seen that mobile devices were able to detect keypoints within a reasonable amount of time, but the extraction method used for the descriptors the amount of time taken was significantly higher. However, with the possibility of using the BRISK detector, the amount of time taken for extracting features from an image may increase significantly due to the composition of the descriptor.

2.3 Object Trajectory Prediction

The previous section has discussed the use of feature detectors to be able to recognise an object from within a scene. With the ability for machines to be able to recognise objects and track their movement throughout a scene, the possibility of being able to predict its movement using various different methods are prolific within literature. There is an abundance of work available that discuss the use of **?**, Bayesian Models or Stochastic Models to generate predictions for a trajectory of an object. In this section, the three aforementioned models are discussed and how their method is used to predict the trajectory of an object.

2.3.1 Neural Networks

Neural Networks are prolific in literature and are used for a multitude of varying tasks. The structure of a NN is inspired by the architecture of the human brain and are composed of a large collection of interconnecting neurons (Zou et al. 2008). They have been used in literature to model a pedestrians behaviour in a crowded environment to be used in applications for predicting the pedestrians walking path and intended direction of movement. The ability to model a pedestrians behaviour is a challenging issue and the decision process of a pedestrian can be influence by external factors, such as the interaction with stationary or moving people.

The work by Yi et al. (2016*a*) uses a Convolution Neural Network (CNN) to make a reliable prediction on the pedestrians walking path, or intended direction of movement. The input to their network is a section of an observed pedestrians walking path from a previous collection of frames; with the intended output to a predicted walking path for the next collection of successive frames. The path of a pedestrian are represented as pixel displacement values, the difference between the previous and current location. A prediction was generated using a CNN that has been trained in two different methods. The first model was trained using a collection of human-annotated pedestrian locations; whilst the second model was trained using the KLT (Tomasi & Kanade 1991) trajectories. The results of the work by Yi et al. are shown in Table 2.6 and using a scoring metric known as the MSE for the two training methods that were used.

Table 2.6: The prediction results (MSE) of the different methods trained using the annotated pedestrian locations, or KLT trajectories on the two datasets. Table of results acquired from the work of Yi et al. (2016*a*).

The authors CNN, known as the 'Behaviour-CNN', can be seen to achieve the best results out of the various other methods that have been evaluated. The best method of training used for their

work was the human-annotated locations of the pedestrians. It is expected to see that this method would achieve the lowest MSE score due to the accurate marking of a pedestrian through multiple scenes. There is always a possibility that using an algorithm like KLT could incorrectly label a pedestrian in a crowd; this would account for the slightly higher MSE score. However, it can be seen that both methods have performed favourably in comparison to other methods such as the Support Vector Machine (SVM) regression and constant acceleration.

The work undertaken by Alahi et al. (2016) have used a neural network to predict the motion dynamics of pedestrians that are situated within a crowded scene. There are instances where pedestrians are able to take various paths to accommodate changes within their environment, i.e. obstacles that move into their path. Alahi et al. (2016) use a LSTM neural network to learn the movement of a pedestrian and predict their future trajectory. The model by the author is able to account for the behaviour of people within the crowd to ensure that a trajectory is accurately predicted; they call their model a 'Social-LSTM'. The model is experimented with on a collection of datasets that consists of trajectory data for multiple pedestrians that are moving within a crowded environment. Three different scoring metrics have been applied to the predictions: average displacement error, final displacement error and the average non-linear displacement error. The model was compared to other leading models that were available at the time of publication, and the results for the average displacement error are shown in Table 2.7. The full collection of results can be found in the work by (Alahi et al. 2016, see Table 1).

Table 2.7: The average displacement error of the predictions using the LSTM model built by Alahi et al. on various datasets.

Dataset	**LSTM**	**O-LSTM**	**Social-LSTM**
ETH	0.60	0.49	0.50
HOTEL	0.15	**0.09**	0.11
ZARA 1	0.43	**0.22**	**0.22**
ZARA 2	0.51	0.28	**0.25**
UCY	0.52	0.35	**0.27**
Average	0.44	0.28	**0.27**

The results shown in bold were the best result that was obtained out of the collection of results. It can be seen that the Social-LSTM gained a majority of the best results from their evaluative study, with a low error rate being recorded across all metrics. When accounting for all the dataset results and calculating an overall average of the displacement error, it can be seen that the Social-LSTM was the best performing across two out of the three metrics. Alahi et al. discusses that their model out performs the other models on the standard dataset and provide qualitative analysis of the Social-LSTM. The analysis involved using the scenes of people where they interact with other people in a particular pattern. Based upon their qualitative analysis the authors' conclude that their model is able to make an intelligent prediction on the future trajectory of a pedestrian in a crowded environment.

The work by Yi et al. (2016*a*) and Alahi et al. (2016) have both used neural networks to predict the trajectory of a pedestrian for their future destination or for a route to avoid collisions with other obstacles. The applicability of their work can be used in work where predicting the final destination of objects can be obtained by analysing the current trajectory of an object. Recurrent Neural Networks (RNN) could be used to provide a long-term prediction by using a short-term prediction to be fed-back into the network. Yi et al. conclude in their work that by using a CNN to model the behaviour of a pedestrian can be used to build an encoding system to encode the pedestrians behaviour from a crowded scene. A common trait from the work by Yi et al. and Alahi et al. (2016) is that the datasets used for testing their networks have used human-annotated data. An alternative method has been used by using KLT trajectories (Yi et al. 2016*a*), but this method led to a slight increase in the MSE scores of their study.

A method of labelling pedestrians from crowded scenes autonomously and without a prior need for training has been explored by Shao et al. (2015). The authors used a CNN to count the number

of people that exist in a high-density crowd and is training by using a switchable learning process with two main objectives: building of a crowd density map and counting the number of people within the crowd. The CNN model used by the authors can learn the crowd-specific features of an environment which can be more effective and robust than using hand-crafted features (Shao et al. 2015). The authors model their neural network on three different datasets, and the results of their work can be seen in Table 2.8.

Table 2.8: The statistical data of the three datasets used by Shao et al. (2015). The results are acquired from the paper published by the author.

Dataset	**Number of Frames**	**Number of Scenes**	**Resolution**	**FPS**	**Minimum & Maximum**	**Total Number of People**
UCSD	2 000	1	158x238	10	11 to 46	49 885
UCF_CC_50	50	50	-	-	94 to 4 543	63 974
WorldExpo	4 440 000	108	576*720	50	1 to 253	199 923

The neural network built by the authors were able to label 199, 923 pedestrians from the 'WorldExpo' dataset, which consisted of over four million frames for analysis. This dataset was introduced by Shao et al. and was built from the observation of 1, 132 frames from 108 surveillance cameras. The labelling of a pedestrian was recorded at the centre of their head across a collection of 3, 980 frames. The authors conclude in their work that the model is effective for annotating people from an environment with a high crowd density that is typically considered to be 'unseen'. By employing this method of annotation to a scene that has not been previously observed, it can assist a neural network in being able to detect pedestrians and employ a model by Yi et al. (2016*a*) to track their trajectory. Using such a model like this would ensure that pre-training would not be required to detect pedestrians and could be undertaken during the real-time analysis of a video. However, a major limitation with the use of neural networks is the amount of computation power that is required to build and train the model. A majority of the time taken for a neural network is spent in the training phase to ensure that an accurate prediction is made; and without investing time in this training phase the resulting outputs could be of a poor quality. Considering the constraint that is imposed on hardware by the limitation of processing power, it would not be feasible to use a neural network of such high cost to make predictions. Therefore, alternative methods such as Bayesian models and stochastic models have are used within literature for prediction generation due to a low computational cost due to the nature of their methodologies.

2.3.2 Bayesian Models

Bayesian models have been in literature for predicting the trajectories of pedestrians in a similar fashion to the work that has been done by Yi et al. (2016*a*) and Alahi et al. (2016). A Bayesian model represents the conditional independences between a set of random variables Ghahramani (2001) and have been used for trajectory prediction by Kim et al. (2015) and further expanded upon in the work by Bera et al. (2016) to build an individual motion model for each pedestrian. The work undertaken by Kim et al. uses various sensor data from the robot to improve the motion model of a moving pedestrian from the perspective of a robot and its environment. The work is an expansion of work by Van Den Berg et al. (2011) and is used by Kim et al. (2015) to reproduce the pedestrian behaviours when moving, i.e. lane formation and the style of movement. The Reciprocal Velocity Obstacles (RVO) motion model by Van Den Berg et al. was chosen for its suitability to be used with various sensors that can be found on a robot.

The work of Kim et al. (2015) assumes that a pedestrian is a human entity and will be sharing their environment with a robot that has the ability to move. They also assume that the humans will automatically seek to not collide with another human but not with a robot; and therefore propose a model that will predict the trajectory of pedestrian to move the robot so a collision does not occur. There are many factors in a robots environment that could influence the prediction tracking method of the authors work (Bayesian Reciprocal Velocity Obstacles (BRVO)). Poor

lighting conditions can reduce the efficacy of the camera sensors which could in-turn increase the noise in the sensor data. People are also able to change their velocity of movement frequently in order to avoid colliding with other obstacles. This would increase the uncertainty in their motion path. Factoring these environmental factors can make it increasingly difficult in predicting a pedestrians movement.

The results of the work by Kim et al. (2015) show that the stability of their work is robust and able to generate a high-quality motion prediction for various scenarios. The authors work was used on a variety of datasets and measured the mean the mean prediction error for each robot in the scene for the entire video sequence. The graph in Figure 2.7 shows hows the density of a scene could affect the error of the predictions made.

Figure 2.7: The error for various densities, where a lower score is more favourable. The graph is taken from the work of Kim et al. (2015).

From the graph, it can be seen that the work by Kim et al. performs exceptionally well across all three densities with only a marginal increase in the lower densities. It has been discussed by the author that robots move relatively slower in a higher density crowd and would explain the reason behind why the error rates are lower for a larger crowd. It can be seen in a low density crowd the error rates for the alternative methods of motion prediction the BRVO method performs significantly better.

The work of Kim et al. has been used by Bera et al. (2016) to compute the individual motion model for each pedestrian, known as Global and Local Movement Pattern (GLMP). The authors model learns the movement flow from a pedestrians two-dimensional trajectory that have been extracted from a video source. The authors emphasise in their work that their approach involves no pre-computation or learning and is able to compute in real-time. A Bayesian-interface is used when extracting the trajectories to compensate for any errors that may occur and for when the state of each pedestrians movement is computed. The movement patterns of a pedestrian is extracted using their model which describe the trajectory motion or behaviour of a pedestrian for a single frame from the video. The results of the work have been compared to the works of Kim et al. and it can be seen that the prediction accuracy of the GLMP model is significantly higher in comparison to BRVO. The GLMP model is able to make two types of predictions:

1. **Short-term**: a prediction is made for the next one second of a pedestrians movement
2. **Long-term:** a prediction is made for the next five to six seconds of a pedestrians movement

The GLMP methods gains the best results for both the short and long term predictions that are made, especially within scenes where a high density crowd is observed. The accuracy of predictions for this density is within the range of 60.2% and 71.2%, whilst the low density crowds were recorded to be between 79.5% and 83.8%. The model is able to work best under conditions where the frame-rate of the video is relatively low, or when the video data has been sampled at large intervals. This

type of process would typically not perform well for conditions when the trajectory data needs to be computed continuously for each frame of a video, or in the case where collision avoidant is imperative.

2.3.3 Stochastic Models

There are various types of stochastic models that have been developed over the years. A general definition of a stochastic model is a collection of random and independent variables that can be defined by,

$$\underline{\mathrm{X}} = \{X(t), t \in T\} \tag{2.1}$$

where for each t in index T, $X(t)$ is a random variable. The variable, t, is often interpreted as the time of a process. Therefore, $X(t)$ is considered to be the state of a process at time t and if T is a countable set then $\underline{\mathrm{X}}$ is a discrete-time process, otherwise it is a continuous time process. A continuous time process consist of increments that are independent for all of t_n for the random variables: $X(t_n) - X(t_{n-1})$ (Ross 1996). There are a variety of different models that can be used for generating a prediction or estimation. Some of these models are relatively simple, i.e a random walk, whereas other models can be more complex, i.e Markov Models.

The Markov model, introduced by Markov (1906), is a stochastic process that models sequential data and this process provides a method of modelling the dependencies of current information on previous information (Meyn & Tweedie 2009). The simplest form of a Markov model is a Markov chain and these models have a condition independence property where each state is dependent upon only the previous state and the initial probability vector. An n^{th}-step transition can be defined by the following notation:

$$p_{ij}^{(n)} = P(X_{x+1} = j | X_n = i); n = 0, 1, ... \tag{2.2}$$

where n is the number of steps. The main application of these models is the generating of a prediction or estimation of a given problem. They can also be used for pattern recognition and statistical learning of sequential data. Other derivatives of the Markov model exist within literature such as Hidden Markov Models and Semi-Markov Models.

Due to the nature of how a Markov model works, the generation of a prediction can be done computed within real-time as the simple comparison is made between the probability vector and a uniform random number. Markovian models have been used for predicting the movement and location of an object in the work by Nižetić et al. (2009) and Nižetić & Fertalj (2010). Their work explored the use of a Markov chain to predict an object's next cell of occupation, whereas their latter work built a case-study on the different movements that are exhibited by wildlife, vehicles and people. The use-cases were used to build a model schema that would be used select an appropriate Markov model to generated predictions for an objects next movement. The study by Nižetić et al. (2009) evaluated the use of a Markov chain to predict the movement of an object based upon movement in a subset of cells. The authors' work involved splitting the environment of an object into a subset of cells so the object's location is identifiable from each cell, as shown in Figure 2.8. For the purpose of their study, the authors derive the size of the cell to be based upon the speed at which the object is travelling, and it will automatically adjust the size of the cells over the progression of time.

Figure 2.8: The subset of cells that the authors propose to use to predict an objects movement when transitioning between the neighbouring cells. (Image Source: Nižetić et al. (2009))

The work undertaken by Nižetić & Fertalj (2010) considers the movement of people but not their pace. Wakim et al. (2004) studied the behaviour of a person's movement and how their pace may affect a transition that may occur in a model. The model state space consists of a set of all possible paces and direction changes that can be made by a person, i.e. a person standing still may easily be able to change direction, however, someone running may not. Further information can be modelled about a person's movement, it can also include information such as last chance escapes, i.e. ducking and jumping or side-steps. Wakim et al. tested their model using a Monte-Carlo simulation to evaluate the possible occurrence of a vehicle-pedestrian incident. Four evaluative areas can be identified based upon the results of their experiments:

- *between 0-10m*: the likelihood of an incident occurring is non-existent
- *between 10-30m*: a large chance of an incident occurring is likely
- *between 30-52m*: the risk of an incident disappears, but an uncertainty is still present
- *over 52m*: the chance of an incident happening is zero

From the study, it can be seen that there are other behaviours that need to be considered when looking at the movement of people and not just their path of movement (as discussed by Nižetić & Fertalj (2010), Nižetić et al. (2009)). The pace of which a person is moving could inhibit a transition occurring between states, such as the speed of a vehicle may inhibit the next road-segment to be predicted. Nižetić & Fertalj (2010) conclude that although the characteristics of movement between people, vehicles and wildlife are different, they all have one similarity: their movement is made in a spatial domain. Therefore, for any data that is inserted into their schema a desired model is selected and the accuracy of predictions can be influenced. However, depending upon the vagueness of the data inserted into the schema it could affect the model that is selected and therefore ultimately influence the outcome of the predictions.

The movement of people, vehicles and wildlife are not the only models that have been studied using a stochastic model. The prediction of an intended route by drivers can also be considered and the study by Krumm (2008) states that approximately 60% of journeys by drivers are repeated. The work by Krumm studied the short-term prediction of a 'near-term' route using an n^{th}-order model. The terminology 'near-term' is used as a definition for the upcoming road-segment that

the driver is most-likely to take. The state space of the authors' model represents different road-segments and are defined by:

$$X_i \in \{X_{-2}, X_{-1}, X_0, X_1, ..., X_{10}\} \tag{2.3}$$

where X_{n-1} are the proceeding road-segments, X_0 a current road-segment, and X_{n+1} the future road-segments. The model gives a probabilistic prediction on the future road-segment a driver could take and is based upon a number of past road-segments that have been traversed. The authors built an n^{th}-order model to predict the next number of road-segments a driver is most likely to traverse, and experiment with how the number of past road-segments could alter the prediction accuracy of their model.

The authors test their model based upon a collection of results from participants fitted with a GPS tracker in their vehicle. The dataset consisted of GPS coordinates alongside a time-stamp for the duration of their journey. A journey was determined using an algorithm where a gap in the recorded data that was greater than five minutes is trimmed and is considered to be a journey. False positives may occur and therefore the authors destroyed data where fewer than ten sample points were collected. The trip data was further refined to determine the different road-segments using an earlier algorithm developed by the authors (Krumm et al. 2007) which is done by transforming the coordinates into usable data. The 'leave-one-out' method was used to test their model, where all the data excluding the last set of data was used to train the model. The last set was used to test their model and generate new predictions to gain a measurement on the accuracy levels. The authors' model reached an accuracy rate of 90% in predicting a single future road-segment that was 0.15 miles away, with the results tailing off to 50% for predicting road-segments up-to 1.5miles away. The model is sensitive to the amount of historical data that is inserted, and from their results it can be seen that the higher the number of past roads inserted into the model, then the overall accuracy increases. For example, with only one previous road-segment inserted into the model, an accuracy rate of approximately 60% was gained, whereas with two road-segments 80% was achieved, as shown in Figure 2.9.

Figure 2.9: The graph showing the results of the experiment by Krumm (2008) and how increasing the order of the model can increase the overall accuracy of the predictions made. (Image acquired from Krumm (2008).)

The use of Markov models within literature have yielded predictions that are highly accurate. It can be seen that the use of a N^{th}-order model in the work by Krumm can increase the prediction accuracy when taking into account larger number of previous road segments that have been observed. However, it can also be seen that simple Markov chains can yield a fairly high accuracy rate with the work by Nižetić et al. citing that their use of a Markov chain 'is appreciably more accurate than random guessing which would be nearly 11%'. Their model was able to achieve an average of correctly predicted cells at 50% and the average of nearly predicted cells to be 34.5%. The simplicity of the Markov models make them useful for generating predictions in real-time for a single step in-time. However, higher-order models can be utilised to make a prediction for the

next location of an object in n steps. In order to predict the trajectory of a person using Markov models is lacking due to the lack of information that is collected and observed by the model. Markov models are great for being used to predict the next direction of movement or an intended destination but is unable to determine a path of movement.

Semi-Markov models have been used in a study by Xia et al. (2011) where the authors studied the spatial and temporal modelling of a tourist's movement. The movement of a tourist is defined as being a random spatial and temporal process and can be represented by, $\{X_t, t \in T\}$; where T represents either the discreet or continuous time sequence of a process. The states in the authors' model are the destinations and transit route locations of a holiday resort. Two assumptions are made by Xia et al. on the tourists attractions:

- the probability of a tourist visiting an attraction is dependent upon the most recent visited attraction
- the distribution of time spent at each attraction is dependent upon the current attraction being visited and the next intended attraction

Paths of a tourists route between attractions were captured with tourists drawing their intended route upon a supplied map. The authors distributed questionnaires and responses were entered into a database to model decisions made by tourist on the choice of attraction they would like to visit. The decisions were made by assessing the spatial and temporal interactions between the tourist and the attraction. The outcome of the authors' study can provide tourism managers with information for destination management and on-site movement planning. The authors' semi-Markov model estimates the probability of a visit to an attraction, with managers using this information to estimate the number of visitors who will move between one attraction to another and decide the placement of paths and other crowd controlling mechanisms. The authors conclude that most studies focus upon the movement of tourists and solely upon the pattern of movement. However, in this study the authors also analyse the time that is spent at each attraction and this novel method of incorporating the time spent at an attraction can further be used in the prediction of the movement of an object. By providing a secondary layer to the prediction process and analysing the duration of an object's time in a state could alter the prediction that is generated, depending upon the time spent in a state in previously recorded data.

2.4 Conclusions on the Reviewed Literature

With the introduction of various features over the years, it has enabled developers to introduce algorithms that recognise objects from within their environments, similar to human vision. With popular feature detectors like SIFT and SURF being used in literature for these tasks, it can be seen that the use of such traditional feature detectors are computationally expensive. With the continuing rise of mobile devices and the inclusion of high resolution cameras, the prospect of being able to recognise objects on these devices are becoming intrinsic.

It has been discussed in this chapter, that the use of SIFT on mobile devices for real-time object recognition is not feasible (Cornelius 2014) and although the work of Bay et al. (2008) has lowered the computational overheads with their SURF detector the requirements are still far too high to be considered useful for mobile device applications. An evaluative study performed by Leutenegger et al. (2011) shows that their BRISK detector is able to perform significantly faster than SIFT and SURF. Leutenegger et al. discusses that BRISK is 'an order of magnitude faster than SURF' (Leutenegger et al. 2011) with the total time for extracting a descriptor taking 0.66 and 0.039 seconds, respectively. It can also be seen that the matching performance also saw a significant difference with BRISK matching all descriptors within 0.029 seconds, compared to 0.195 seconds by SURF. Therefore, the possibility of recognising an object within a real-time constraint on mobile devices is becoming more feasible.

However, it has been acknowledge in work by Coltin et al. (2016) that BRISK is not as robust of a descriptor in comparison to SURF and have proposed using a combination of both detectors to build a map of the International Space Station (ISS). Therefore, the following work will be undertaken for object recognition using,

- **Traditional Detectors**: recognising an object on mobile devices will be undertaken using solely traditional feature detectors SIFT and SURF.
- **Binary Detectors**: recognition of an object will be performed on mobile devices using BRISK.
- **Binary and Traditional Detector**: a hybrid approach of using BRISK and SURF to extract descriptors.

The outcome of this work will be used to ensure that a robust recognition of an object on mobile devices can be performed and that continuous tracking can occur of an object moving throughout its environment.

With the possibility of being able to recognise and track an object moving within its environment, it can open the doors towards the possibility of predicting the intended trajectory of an object and the next direction of movement. From the literature reviewed, it can be seen that their is prolific use of neural networks (NN) to predict the trajectory of an object and its intended final destination. However, it is common to see with this sort of predictive method that a high level of training and pre-emptive learning is required to ensure that an accurate prediction is made. Using a neural network on mobile devices could inherently prove rather difficult due to the large computational cost that is spent on training and learning; and without the ability to scale-up the hardware it could take a large amount of time to make a single prediction. Therefore, the use of neural networks would not be feasible for making real-time predictions on a mobile devices.

Bayesian models have shown to be highly accurate with a relatively low computational cost. However, from the literature reviewed, the work by Bera et al. (2016) used a source of video that was sampled at high intervals and therefore information between a pedestrians recorded location movement is missing and could possibly have influenced the output of the Bayesian model. A similar probabilistic model has also been discussed, the Markovian Model which has been used in work by Nižetić et al. (2009) to predict the next cell of occupation by an object, whilst latter work by the author has explored various case studies that can influence the state movements of a human, vehicle and animal (Nižetić & Fertalj 2010). Higher dimensionality Markovian models have been explored in the work by Krumm (2008) with a study on how increasing the number of road segments observed in the Markov model could influence the prediction accuracy. It could be seen in their work that a single road segment observed could provide a prediction rate of approximately 70%. However, when increasing this to nine previous road segments the prediction accuracy was increased to 90%.

Bayesian models also perform in a similar fashion, with a prediction being based upon probabilistic values but are concerned with underlying factors that are 'hidden'. The hidden values could inflict the outcome of a prediction and are often not able to be recorded. Therefore, the Markovian method is the most suitable method for generating these predictions. However, the probabilistic values are often static/fixed and do not change unlike a Bayesian model. Therefore, it is proposed that work will be undertaken to adapt the Markovian Model to adjust the probabilistic values each time a state transition occurs. The body of work for this part of thesis will involve the use of two Markov Models:

- **Traditional**: a traditional model is built whereby the probabilistic values of the state transitions are fixed and determined by pre-trained data.
- **Dynamic**: an adapted version of the traditional model whereby the probabilities of the state transitions are adjusted each time a transition occurs.

It is expected that with the traditional model requiring training data to ensure an accurate prediction is made that this model would not be feasible for the applicability in real-time applications. However, the adapted Markovian model would fair much better for mobile applications.

Chapter 3

Methodology

The literature reviewed discussed the various feature detectors that are prevalent for formalising a descriptor. Two various types of detectors have been discussed, traditional and binary. Each type have their own set of advantages and disadvantages, and it can be seen in the work by Leutenegger et al. (2011) that BRISK is computationally more efficient in comparison to its traditional counterparts. Previous experimentation has shown that SIFT is unfavourable for machine vision tasks on mobile devices (Cornelius 2014). Therefore, the use of a new feature detector is proposed to be used to recognise an object from its environment. This will enable work to be undertaken on a method to predict the next intended direction of movement.

In this chapter, the method behind recognising an object using a feature detector is discussed and how the methodology will be experimented to determine whether the use of BRISK is able to provide a recognition within a real-time constraint. The experiment will also discuss whether the hybrid nature of using a traditional and binary feature detector can affect the robustness of the recognition phase whilst ensuring a low computation time is achieved. Based upon the recognition of an object, the various properties that can be extracted based upon the process is also discussed, with a focus upon the boundaries of the detected object and the measurement of it's movement in the environment.

Literature has shown that Bayesian models, neural networks and Markov models have been used to predict the trajectory or movement of a pedestrian. Work by Krumm (2008) have used Markovian models to predict the next road-segment a road-user would take to get to their intended direction, and shows favourable results in regards to the accuracy of predictions that are made. The detected movement of an object is then used to generate a prediction on the next direction of movement. The chapter will discuss the various neural networks that are implemented to provide a comparative study to the traditional Markovian model. This chapter will introduce the Markovian model that features a dynamically updated probability vector, and discusses how the adaptation of a traditional model can ensure that robust predictions are made.

3.1 Object Recognition on Mobile Devices

Capturing an image can be done using a digital camera, either it be a stand alone unit or in-built into the mobile hardware. The pixel data that is captured within this image is not the only information that is collected. Information stored within the file of the image, known as Exchangeable Image File Format (EXIF) is readily accessible and can provide information on:

- **date and time**: the time-stamp of an image or video when it was taken or recorded is stored in the image file properties
- **camera properties**: information regarding the camera brand and model and other information such as orientation, focal length and shutter speed are stored
- **thumbnails**: a small representation of the image is stored to be used for temporary viewing on the camera device or software
- **descriptions and copyright**: information regarding the image and any copyright information pertaining the image are stored, i.e. the GPS location

However, other there is other pieces of information that can be extracted from an image with the use of feature detectors. The recognised object by using these detectors can provide information regarding the size of an object and its movement within the environment. Other properties such as the distance from the cameras viewpoint can also be ascertained. These properties are often approximated with human vision; however, when it comes to the task for machine vision it requires a higher level of understanding and knowledge. Therefore, the process required for recognising an object from images or videos is discussed in this section, along with the various calculations that can provide meaningful data to be used in the predicting algorithm.

3.1.1 Recognising an Object using Feature Detectors

Recognising an object using machine vision has been previously discussed in Section 2.2. It has been shown that the BRISK detector is significantly faster in comparison to traditional detectors such as SIFT and SURF. To see the difference between the binary and traditional detectors, the traditional feature detection method has been used in the previous work (Cornelius 2014) and will be used again, but with the inclusion of BRISK to compare the performance between a traditional and binary detector.

The methodology for each feature detector contains a set of parameters that will ensure a robust method for formalising a descriptor. Improving the recognition rate of the algorithm can be achieved by adjusting the parameters of each detector. For the purpose of this experiment, the default values shown in Table 3.1 are used. The values are suggested by the Open Source Computer Vision Library (OpenCV) (Bradski 2000) which has been used due to the pre-implementation of the BRISK, SIFT and SURF detectors.

Table 3.1: The various parameters that have been used for the BRISK, SIFT and SURF feature detectors. The parameters are the default values as suggested per the OpenCV framework.

Detector	Parameter	Description	Value
SIFT	# of features	determines the number of features to retain that are ranked by their score	0
	# of octave layers	determines the number of layers in each octave and computed automatically based upon the resolution of the image	3
	contrast threshold	filters out any weak features that may occur in low contrast regions, the larger the number then the fewer features that are detected	0.04
	edge threshold	filters out features that are localised upon an edge	10
	sigma	the amount of Gaussian blur that is applied to the image at the first octave	1.6
SURF	hessian threshold	a threshold for the keypoint detector to return keypoints where the hessian is larger	100
	# of octaves	the number of octaves that are formed in the pyramid scale	4
	# of octave layers	the number of images that are present in each octave	2
	extended	boolean value, whereby set to false the dimensionality of the descriptor is 64 elements, whereas if it is true then the dimensionality is set to 127	false
	upright	boolean value, whereby if set to true the orientation of a descriptor will not be computed, otherwise it will be	false
BRISK	threshold	the detection threshold of the keypoints	30
	# of octaves	the number of octaves used in the sampling pattern	3
	pattern scale	the scale of the pattern for sampling the neighbourhood around the keypoint of interest	1.0

Although two varying types of feature detectors are used the process they follow to recognise an object from within its environment is similar. The process is split into three key areas:

Step 1 **Keypoint Detection and Formalising a Descriptor**
Localising key areas of interest will be performed upon a set of images. The detection method requires two images, one to represent the object to be detected (known as a training image). A second image will consist of the object within it's environment (known as a scene image). Key areas of interest will be detected to formalise a descriptor for each detector. The process is computed once for the training image, and is repeatable for the scene image depending on whether a singular image, or sequence of images/video frames are used.

Step 2 **Keypoint and Descriptor Matching**
A matching algorithm is used to match the keypoints and descriptors between the training and scene images. Dependant upon the type of descriptor that is used, the type of algorithm used will be either the BruteForce (BF) or Fast Library for Approximate Nearest Neighbours (FLANN). Each method will use the k-nearest neighbours (KNN) matching sequence which will return the k best matches. In this instance, the two best matches will be returned from the collection of results and will be used in a refining process.

The refinement process will be done using a ratio test between the matched results distance and will then be subjected to a further refinement. A comparison test is made between the distance of the refined matches to ensure that they are equal in length and ensures that a 'good' match has been made. The matching method can influence the robustness of the recognition phase; however, this is not the only factor as the number of keypoints can also influence the robustness of recognition.

Step 3 **Formalising the Recognised Objects Shape**

In order to show that a recognition has been successful in the scene image, the boundaries of the training image are transformed so they can appear around the 'detected' object in the scene image. The process that is followed to do this will use the homography function that is in-built into OpenCV. The process assists in calculating the perspective transform of an object between the shape in the training image and its detected self in the scene. The transform is a collection of co-ordinates that can then be used to draw a line around the detected object.

The boundaries/shape of the detected object can then be used to determine the location and subsequent movement of an object when it moves through a sequence of images/video frames. This will provide key information that can be used in the neural network and Markovian models discussed in Section 3.2.

Each step of the process are repeatable, depending upon the type of media used. If a video source was used, then each frame of the video will be subjected to the recognition process and it can become quite a lengthy process due to the amount of computing that is undertaken. Therefore, its important to ensure that the original frame-rate of the video is maintained. It is understandable as to why the use of traditional detectors are considered to be unfavourable in these circumstances due to the large overheads. Therefore, alternative methodologies for detecting the object can be undertaken, i.e. the use of a sole binary detector or the hybrid model used by Coltin et al. (2016). Using a hybrid model could theoretically reduce the amount of computation that is required to recognise an object from its environment. It has been discussed in literature that the BRISK detector is not as robust in comparison to SURF and therefore it is proposed that a combination of the two methods will be used. The BRISK detector will be used to detect keypoints from an image due to the large number of keypoints that can be detected within a reasonably low computation time, and SURF will be used to formalise a descriptor based upon the keypoints that were extracted.

3.1.2 Calculating the Dimensions and Movement of a Recognised Object

The dimensions of an object are useful for assisting in the calculation of an object's distance from the camera. They can be measured by using the boundaries of an object that has been found within its environment using the recognition process discussed in Section 3.1.1. Two opposing corners from the boundaries will be chosen and the distance between the two corners are represented in a pixel measurement. The pixel measurement is not useful for ascertaining the true dimensions of an object in the real world. Therefore, to convert this measurement into something meaningful, such as millimetres, a variable known as the pixel-per-metric (PPM) is calculated by using the following formula,

$$\text{ppm} = \frac{p_w}{\mathbf{K}_\text{w}} \tag{3.1}$$

where p_w is the width of an object in pixels, and $\mathbf{K}_\text{w}$ is the known width of the object in millimetres. The PPM is used to scale an object's dimensions accordingly to determine the dimensions of the object in real-life using the formula,

$$\text{R}_{(\text{w,h})} = \frac{p_{(w,h)}}{\text{ppm}} \tag{3.2}$$

where $p_{(w,h)}$ is the pixel width and height of an object. The extracted dimensions can then be used for determining the direction of movement an object may exhibit when moving in a sequence of images or frames from a video.

The movement of an object can be done in two different methods: velocity or pixel displacement. Due to the simplicity of the displacement method this makes it suitable for calculations that require to be processed in a real-time constraint. To determine the movement of an object, the location it situates within a scene is required. The location in this instance is the central pixel that is within

the confines of the detected boundaries and can be calculated by the following equation,

$$\mathrm{C}_{(\mathrm{x,y})} = \left(\frac{(c_1 - c_2)}{2}, \frac{(c_2 - c_4)}{2} \right) \tag{3.3}$$

where c_1, c_2 and c_4 are the various corners of an objects shape. Due to a susceptible amount of noise that may be present when the detection of the boundaries are performed, fluctuation in the location of the detected central pixel may occur. Therefore in order to minimise the amount of noise the Exponentially Moving Weighted Average (EWMA) (**?**) is applied,

$$\mathrm{C}_{(\mathrm{x,y})} = \alpha \cdot \mathrm{C}_{(\mathrm{x,y})} + (1 - \alpha) \cdot \mathrm{C}_{(\mathrm{x,y})-1} \tag{3.4}$$

where $\mathrm{C}_{\mathrm{x,y}-1}$ is the previous location and $\mathrm{C}_{\mathrm{x,y}}$ the current location of the object and α being a pre-defined value chosen by the user. Adjusting the value of α could reduce the rate of sporadic movement that occurs on the central pixel by a poor transform of the objects shape. Using the EWMA formula will ensure that fluctuations of the central pixel is reduced to a minimum and does not affect calculating the direction of movement. In order to calculate the movement of an object, the previous and current location of an object is required. The following formula,

$$\mathbf{C}_{(x,y)} - \mathbf{C}_{(x,y)-1} \tag{3.5}$$

will subtract the current location from the previous to provide an absolute difference. The result is then subjected to conditional check to simplify the direction to a bipolar/binary value:

$$\mathbf{M}_{(x,y)} = \begin{cases} 1 & \text{if } \mathbf{C}_{(x,y)} - \mathbf{C}_{(x,y)-1} \geq 0.5 \\ \text{-1} & \text{if } \mathbf{C}_{(x,y)} - \mathbf{C}_{(x,y)-1} < 0.5 \\ 0 & \textit{otherwise} \end{cases} \tag{3.6}$$

where $\mathbf{C}_{(x,y)}$ is the current location, $\mathbf{C}_{(x,y)-1}$ is the previous location and $\mathbf{M}_{(x,y)}$ is the simplified movement of the object.

The conditional check will determine if the pixel differentiates over a certain threshold, and will then be simplified to the relevant bipolar/binary value. The values returned are a set of coordinates that will represent the direction of movement for a two-dimensional plane. The simplified values are used to represent one of eight directions of a compass (shown in Table 3.2. A ninth movement is added to the list of directions to include a stationary movement.

Table 3.2: The eight possible directions of movement in their simplified versions, along with a stationary move.

North	**South**	**West**	**East**	**Nort-East**	**South-East**	**North-West**	**South-West**	**Stationary**
(0, 1)	(0, -1)	(-1, 0)	(-1, 0)	(1, 1)	(1, -1)	(-1, 1)	(-1, -1)	(0, 0)

3.1.3 Datasets used for Experimentation

In order to understand how well the recognition process of the various feature detectors perform, a collection of different datasets are required. For the experiments to be useful, a four various datasets have been proposed to use to gain an insight in how various resolutions and continuous data input can affect the recognition methodology. Similar to the work undertaken by Cornelius (2014) the dataset used by Lowe (2004) to experiment with SIFT will be used for this study as the first dataset. The four images by Lowe (2004), shown in 2.6, were used in the original work and are of a significant low-resolution and in a grey-scale format. With the resolution of these images being significantly lower in comparison to those that can be captured using mobile hardware, it could potentially affect the performance of a detector due to the natural Gaussian filtering that may occur. With the advancement of modern technology and a cameras mega-pixel count becoming increasingly larger, the applicability of the dataset by Lowe is becoming out-dated. For example,

a mobile device consisting of an eight megapixel sensor would be able to capture a still image with a resolution of 3266-by-2449. Therefore, to ensure that an experiment can be performed upon the types of images/videos that can be captured with modern devices, a secondary dataset has been formed.

The second dataset consists of images that are high in resolution and are scaled to account for each type of resolution that is commonly used within media: 540p (960x540), 720p (1280x720) and 1080p (1920x1080). The new collection of images that will be used for the experiments are shown in Figure 3.1.

(a) Training Object **(b)** Non-Occluded Scene **(c)** Occluded Scene

Figure 3.1: The new collection of images used for the experimentation to determine how traditional and binary detectors perform on high-definition images.

The images for this dataset consists of a single training object, which will be subject to the recognition process to be detected within two differing scenes. The first scene consists of the object within an environment with no occlusion preset. The second scene consists of the object in the same location but with a variety of objects with different textures that are partially covering the object. The dataset by Lowe (2004) will continued to be used to provide a comparative study on how the difference in resolutions and number of pixels may affect the overall computation time of the process for recognising an object. However, the recognising an object from static images is a trivial task. The ability to recognise an object whilst its moving and in a collection of successive frames is inherently more difficult. Therefore, two further datasets have been formed, one using the in-built web camera of a laptop and the second being a duo of videos that have been scaled in various resolutions.

The third dataset consists of using the web camera on a personal laptop, which will feed data into the object recognition process. This dataset has solely been used to understand how using a hybrid combination of BRISK and SURF can affect the robustness of the recognition phase. The web camera of the laptop is set to record at a 720p resolution to ensure that an optimum performance is achieved, and is run indefinitely to ascertain the frame-rate of video and the rate of recognition. Similarly, a fourth dataset has been composed of videos to experiment with the hybrid detection method on sequential data; i.e. the numerous frames of a video. The videos will consist of an object moving in its environment in two different trajectories: a straight line, or a rectangular trajectory. These videos will provide optimum conditions for the process of recognising an object from its environment, and the drawing of an objects boundaries. The videos are exported in a 1080p resolution which will be scaled appropriately for each of the other media resolutions, programmatically. The frame-rate and run-length of the video have been set to a fixed value: thirty frames-per-second (fps) and a length of five minutes, respectively. These variables have been selected as they will provide a total of $9,000$ frames.

3.1.4 Experimental Plan and Measurable Metrics

The experimental plan for recognising an object from its environment has been split into four different categories, with each one representing a particular dataset:

1. **Low-Resolution**: an experiment is performed using the low resolution dataset as used by

Lowe (2004) in the evaluation of the SIFT feature detector. The dataset is used to evaluate the following feature detectors: BRISK, SIFT, and SURF.

2. **High-Resolution**: the newly formed dataset discussed in the previous section is used for evaluating the detectors: BRISK, SIFT, and SURF.

3. **Live-feed**: a continuous input of data using the web-camera on the personal laptop is used to evaluate the following feature detectors: BRISK, SIFT, and SURF. A hybrid combination of the SURF and BRISK detector is also used.

4. **Computer-Generated Videos**: the computer generated videos are used to evaluate the features detectors as aforementioned, along with a hybrid combination of the SURF and BRISK detector.

By splitting the experiments into four, it can provide a comparative study between the various datasets. An insight can be provided on how an input of sequential data (the videos and web camera) can have an adverse impact on real-time recognition when compared to the same task on static images. Each experiment category is repeated three times to provide an average across the three set of results. Repetition has been undertaken to ensure that any anomalies that may occur can be reduced and not impact the analysis of the results. Analysis will be undertaken by using the following measurable metrics:

- the time of processing for detecting the keypoints, extracting a descriptor and performing a match
- the amount of keypoints detected
- the robustness of detecting an object in live-feed data

The experiments have been undertaken on a couple of devices, a personal laptop computer and the Sony Xperia Z3 Tablet Compact mobile device. The two devices have been used to provide a comparative study on how the performance of recognising an object can differ between the hardware used in a mobile device and laptop can significantly affect the time taken for completing the task. Measuring the time that was taken for detecting the keypoints of interest and the other methods can be computed by taking a timestamp before the process is begun and another at the end of processing. The two values can then be subtracted by one another by using the following formula,

$$t = \frac{end - start}{frequency} \tag{3.7}$$

where *end* and *start* are the respective time-stamps that are collected, and $frequency$ is the number of ticks per second as the timestamps collected are a reference to a tick number.

The number of keypoints detected can be measured by using the OpenCV libraries in-built functionality which will return the total number of keypoints that have been detected. This number is measured to determine how the size can influence the robustness of the recognition that is performed. The robustness is measured by visually inspecting the boundaries that are drawn around the object that is to be detected. An example of what is considered good and bad is shown in Figure 3.2. This measurement is taken solely on the recognition process that is undertaken using the live-feed data, i.e. data that is continuously input into the algorithm either from a video or web-camera.

(a) Bad Recognition **(b)** Good Recognition

Figure 3.2: The detection outcome using the third set of parameters for BRISK. Although a strong detection can be made as shown on the right, a weak detection can also occur (as shown on the left).

3.2 Predicting the Movement of an Object

The process used for recognising an object from it's environment and tracking it's movement has been discussed in Section 3.1. Using the basis of this process and the various formulas provided to determine the movement of an object. The prediction of the next intended direction of movement can be generated by using a NN or Markovian Model. For the purpose of this study, it has been chosen to use a Markov Model in one of two ways: a traditional methodology, and a new methodology that can adjust the weightings of a transition. Two types of NN's have been implemented to provide a set of comparative results which enables a study to be held on the amount of computation power (and the processing time) that is required by the networks and models. The two networks of choice are the Feed-Forward Neural Network (FFNN) and RNN. The latter model is closer in methodology to a Markovian model whereby a 'memory-like' state is used to feedback the output back into the neural network. In this section, the methodologies of the Markovian models and neural networks are discussed in detail.

3.2.1 Using a Traditional Markovian Model for Predictions

The movement of an object is defined in this work to be moving within one of the eight compass directions, with the inclusion of a ninth state for being stationary. The types of movement are defined in Table 3.2 and can be classified as a random spatial and temporal process. Simple Markov models, known as Markov Chain (MC) can be denoted as:

$$P(X_{n+1} = i_{n+1}|X_n = i_n, ..., X_1 = i_1) = P(X_{n+1} = i_{n+1}|X_n = i_n) \tag{3.8}$$

where $\{X_0, X_1, ...\}$ are a sequence of random variables and $Z = 0, \pm1, \pm2, ...$ the union of sets and their realisations. The models consist of a set of states, $S = \{s_1, ..., s_n\}$, that have a set of transitions that can occur between each of the states. They have a conditional independence property where each state is solely dependant upon the previous state and a initial probability vector. MC's are a simplified form for a model that is known as a first-order. They are often denoted as the following (Meyn & Tweedie 2009),

$$P(s_t|s_1, s_2, s_{t-1}) = P(s_t|s_{t-1}) \tag{3.9}$$

A commonly used example to demonstrate the workings of a Markov model is the application to predicting the weather. The model consists of three states with each state representing a typical weather pattern: sun, cloud or rain. An initial probability vector is constructed that will consist of arbitrary numbers to provide a weighting of a transition that may occur:

$$\mathbf{P} = \begin{array}{c} \\ sun \\ cloud \\ rain \end{array} \begin{array}{c} \begin{array}{ccc} sun & cloud & rain \end{array} \\ \begin{bmatrix} 0.8 & 0.05 & 0.15 \\ 0.2 & 0.6 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix} \end{array} \tag{3.10}$$

The probability vector is then used to determine a transition between states; simply, a prediction is made on the next weather pattern that is based upon the currently occupied state. For example, if the current weather is sunny, then the probability of it staying sunny would be 80%. The process of making a prediction for the next state of transition is done by generating a random uniform number and comparing it against the probabilities for its current state of occupation:

$$\mathbf{Pre} = \begin{cases} sun & \text{if } r \leq sun \\ cloud & \text{if } sun < r \leq sun + cloud \\ rain & \text{if } sun + cloud < r \leq sun + cloud + rain \end{cases} \tag{3.11}$$

For example, if $r = 0.43$ and the current state of occupation is *cloud* then the next movement can be predicted by substituting the variables in the previous equation, to supply the following:

$$\mathbf{Pre} = \begin{cases} cloud & \text{if } 0.2 \leq \mathbf{0.43} \leq 0.6 + 0.2 \end{cases} \tag{3.12}$$

The returned prediction is *cloud* as the randomly generated number is less than that of the probability values for a transition between *sun* and $cloud + rain$. However, if the random number had been greater than 0.8 then the outcome would have been different, and it would have predicted *rain*. However, the use of a first-order model only takes into consideration a single previous transition. Markov models have the ability to track a collection of previous movements to make a more informed prediction. These type of models are often referred to as N^{th}-Order models and memory can be built into the model by using the variable level of orders, i.e. two, three or four. The models can be expressed by the following notation:

$$P(s_t | s_{t-1}, s_{t-2}, ..., s_1) = P(s_t | s_{t-1}, ..., s_{t-n}) \tag{3.13}$$

where s_t is the state at time t. The application of higher order models have been used by Krumm (2008) to retain more knowledge about the number of road segments traversed, and their work has shown that it can increase the prediction accuracy of a model. However, increasing the order of a model has a negative impact on the number of transitions that can occur within the model. For example, a first order model will consist of 9 rows in the array, which results in a total of 81 possible transitions. However, a third order model would result in 729 rows of possible movements, which would have a total of $6,561$ transitions. The growth in the number of transitions that occur between states is shown in Figure 3.3.

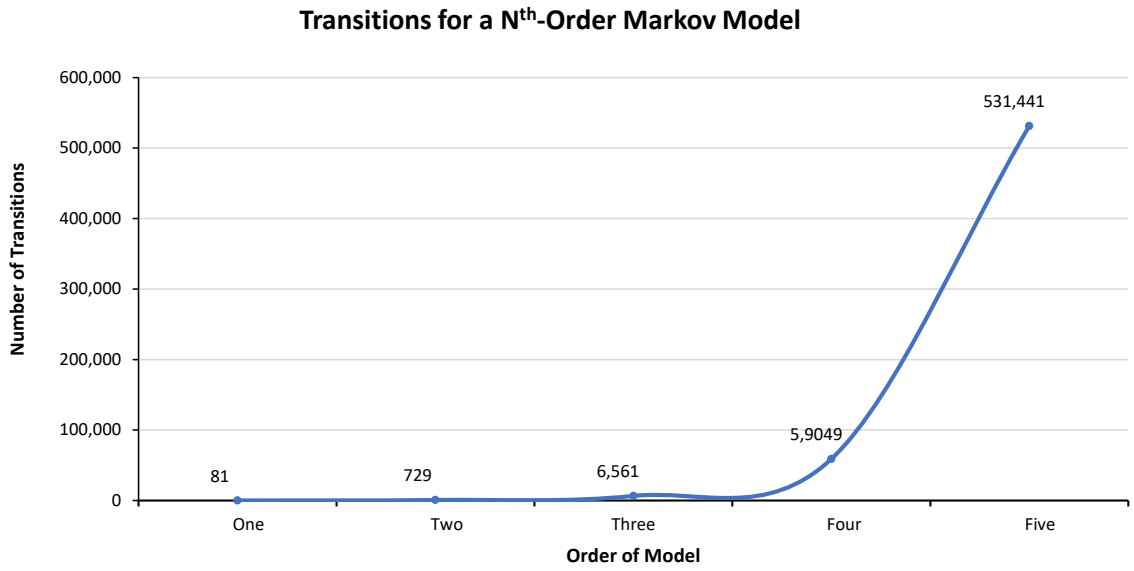


Figure 3.3: The increased number of rows for a higher order model will have a significant impact on the number of transitions that can occur. An exponential growth can be seen for a linear increase in the models order.

It can be seen from the graph that up-to a third order model sees a relative small growth. However, from a fourth order onwards the growth becomes inherently larger and more significant as it goes on. This exponential growth on the number of transitions could impose a restriction on the order of a model that would be suitable for applications to run in a real-time constraint. However, the advantages of using a higher order model could infer a positive outcome with the accuracy of the predictions increasing. Therefore, an alternative method is developed to ensure that an accurate prediction could be yielded from using a lower order model and is discussed in Section 3.2.2.

3.2.2 Amending a Traditional Markovian Model for Predictions

The adapted Markov model follows the methodology of the traditional model, but an adaptation is made to the calculation of the probability vector. Typically, the values of the vector would be static and calculated based upon prior training or historical data. To remove the reliance upon the prior training, a secondary vector can be introduced which will count the frequency of transitions that occur between states. This new vector will be termed as a 'Frequency vector' and is denoted as, $\mathbf{F}$. The probabilities are updated using the frequency vector and is achieved using the following formula:

$$\mathbf{P}_{ij} = \frac{\mathbf{F}_{ij}}{\sum_{i=1}^{k=9} \mathbf{F}_{ik}} \tag{3.14}$$

where $\mathbf{F}$ is the frequency vector, i and j are the indices of the vectors row and column, and k is the number of movements that can be exhibited by the object. The formula will sum up the nine columns for a selected row of the vector, and then individually do a divisional calculation of each column against the total summed row. This will provide an updated weight for the transitions that will occur for the selected state at the time.

The introduced model is termed as a Dynamic Markov Model (DMM) due to the methodology that is used to adjust the probabilities as a transition occurs. To demonstrate how this dynamic

model will work, an initial probability vector is constructed,

$$\pi = \begin{bmatrix} 0.33 & 0.33 & 0.34 \\ 0.33 & 0.33 & 0.34 \\ 0.33 & 0.33 & 0.34 \end{bmatrix} \tag{3.15}$$

where an equal probability has been set for the first initial movement of an object. An initial probability is set at the beginning of the modelling to ensure that an equal chance is available for a prediction to be generated. This process will remove any sort of bias for the first prediction, and therefore it is expected in most cases that the first prediction would be incorrect. If a collection of fifteen transitions have occurred, then the frequency vector will have been updated. For example,

$$\mathbf{F} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 0 & 2 \\ 3 & 5 & 0 \end{bmatrix} \tag{3.16}$$

then the probability vector will be adjusted accordingly using the formula to,

$$\mathbf{P} = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.33 & 0 & 0.67 \\ 0.38 & 0.62 & 0 \end{bmatrix} \tag{3.17}$$

The updated vector will then be used to generate a prediction in the same method as discussed in the traditional Markov model (Section 3.2.1). A frequency vector could theoretically increase the accuracy of predictions for a Markov Chain (MC), with the dynamic aspect of this model it is expected to see that the model will learn the types of movement that are exhibited by an object. Therefore, patterns within the vectors will form and an analysis can be performed upon the patterns to determine whether a specific type of movement can be extracted from the values or placement of them to determine whether there is any similarity between vectors for different observed movements.

3.2.3 Pattern Analysis of an Objects Probability Vectors

The introduction of the DMM can open avenues for the analysis of patterns that may occur within the probability vectors for each observed object. Determining whether there are any patterns in the vector can be done using a selection of formulas that were introduced by Haralick (1979): dissimilarity, entropy and homogeneity. The various formulas are often used for the process of analysing texture within images but could be potentially useful for scoring the probability vector. The various outcomes from the scoring metrics can be collectively combined to analyse the movement and behaviour of an object. Objects are able to take the same route over a period of time, especially when taking into account the environment they are interacting with. An example of this is the entrance to a building at Coventry University, where a funnelling effect can be seen. The students and staff are funnelled towards the entrance of the building (depicted in red) as shown in Figure 3.4.

Figure 3.4: The entrance of the Engineering and Computing building (ECB) of Coventry University. The students/staff are funnelled towards a main entrance (highlighted in red) and a card-access door (highlighted in green).

The red and green boxes highlight the two entrances to the building, where the red highlight is the public entrance and is freely accessible. However, the green highlight is a second entrance that is typically accessed using an identification card, possessed by students and staff members of the university. Therefore, it is most popular for students (or members of the public visiting) to gravitate towards the entrance in that is highlighted in red. It would be expected that the students may head towards this entrance in a straight line and this type of movement may be represented in the probability vector in some way. There may also be scenarios where a sudden change in direction may be exhibited due to either slow moving pedestrians or other obstacles that may occur in the way. Analysis of the vectors can be done to determine whether two observed movements are similar to each other, and the scoring metrics can be used to select a model from an online database. This would subsequently substitute the current model with another that is intrinsically more accurate. The models that are contained in the database could possibly consist of observations that have been done over a longer period of time and are therefore inherently more accurate.

Using the various formulas by Haralick (1979) can be useful for applying this scoring metric. For example, the dissimilarity formula can be used to measure the distance between values in a vector and if the score is increasing then it can be seen that the values do not lie within close proximity to each other in the vector. The formula for calculating this score is shown,

$$\sum_i \sum_j (i-j)^2 \cdot \mathbf{P}_{ij} \tag{3.18}$$

where $\mathbf{P}_{ij}$ is the probability vector at indices i, j for a given object. There is a possibility that this score may only indicate a particular type of movement. Therefore, combining the score with other metrics such as the entropy score could provide something more meaningful. Using the entropy formula would determine the randomness or degree of disorder that may be present in the probability vector. The formula for this equation is show,

$$-\sum_i \sum_j \mathbf{P}_{ij} \cdot log(\mathbf{P}_{ij}) \tag{3.19}$$

where $\mathbf{P}_{ij}$ is the probability vector at indices i, j for a given object. The value of the entropy score is at its largest when all the elements of the vector is the same, or lowest when they are unequal. However, the homogeneity formula could also be applied to understand the distribution of the elements towards the diagonal of the vector (Haralick 1979). The homogeneity can be calculated using the following formula,

$$\sum_i \sum_j \frac{1}{1 + (i - j)^2} \cdot \mathbf{P}_{ij} \tag{3.20}$$

where $\mathbf{P}_{ij}$ is the probability vector at indices i, j for a given object. The dynamic model is built so that any transitions where a pedestrian is walking in a straight line is situated upon the diagonal of a models vector. Therefore, if a homogeneity score is low, it would be assumed that the object is moving within a straight line. However, the direction at which the object was heading towards would be unknown and therefore the other scoring metrics may be able to highlight the direction of movement.

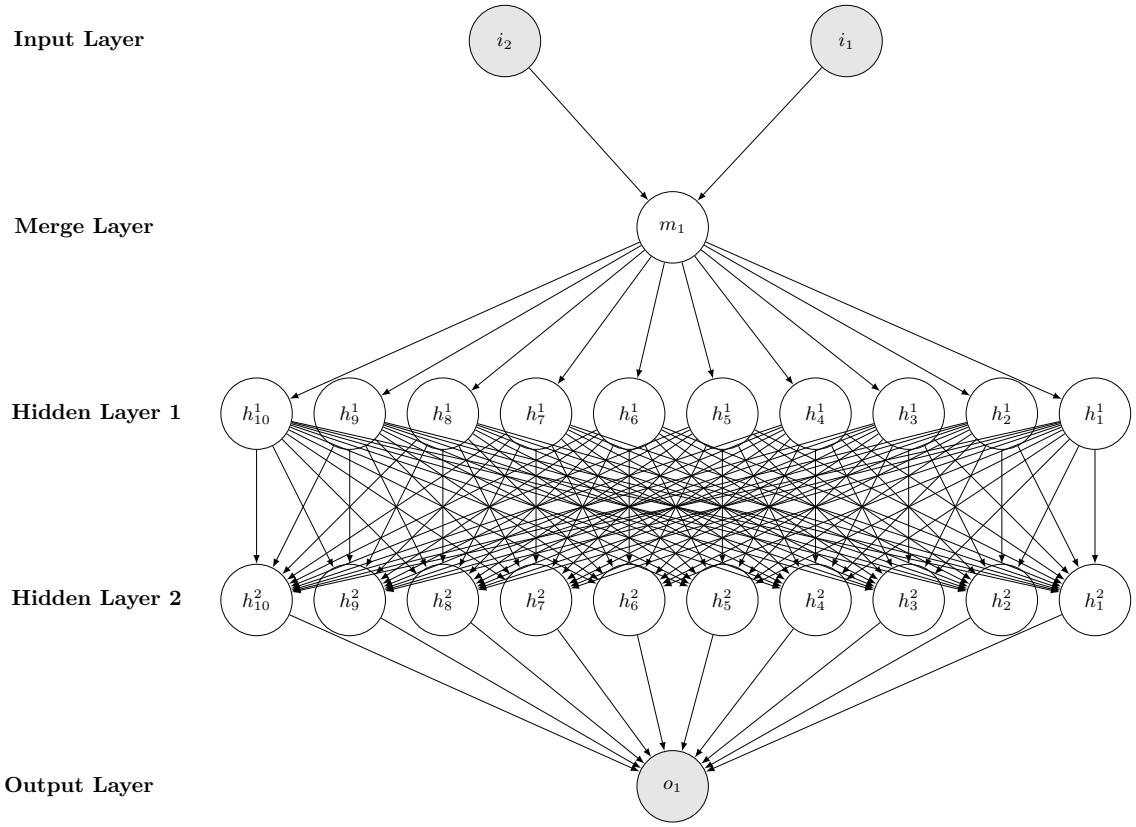
3.2.4 Using Neural Networks for Movement Prediction

NN are predominately used in literature for predicting the trajectory of an object or pedestrian. The ability to predict a trajectory can assist in applications where collisions are to be avoided, i.e. robot navigation (Kim et al. 2015). Two types of neural networks are implemented to provide results for a comparative study with the Markovian models:

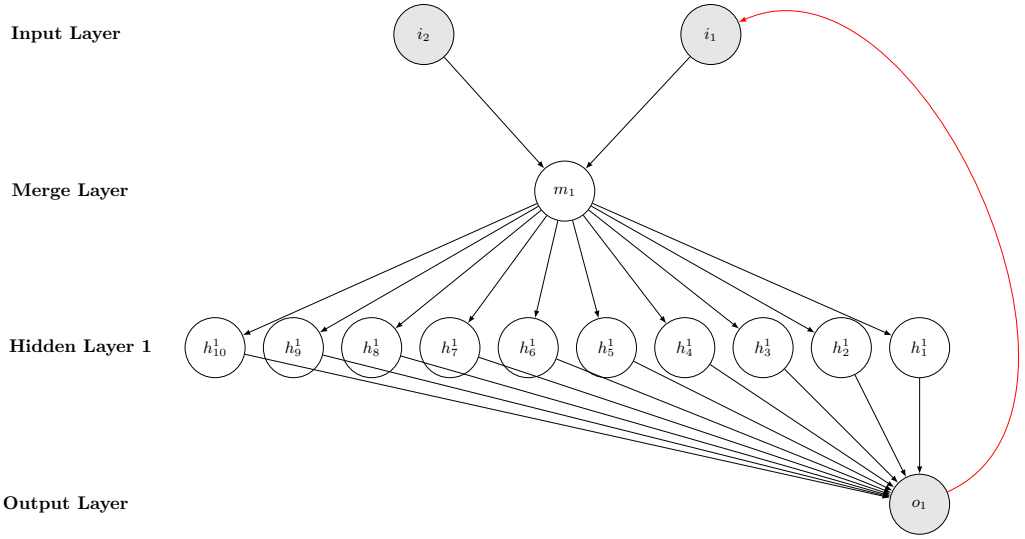
- **Feed-Forward**: these types of networks are symbolic to the process of how the data moves, in a forward direction.
- **Recurrent**: these types of models have a 'memory-like' structure and are similar in process to that of a Markovian model.

The two neural networks are implemented to provide results on how a simple and more complex neural network can be used to make predictions on the direction of movement for an object. Literature reviewed has shown that the amount of time that is taken for training the neural networks means that they are not applicable for using within applications that have a real-time constraint.

The structure of the two models have been built in a similar fashion, with the number of nodes used for the input and output being the same. The only difference between the networks is the number of layers that were used for the hidden layer. The feed-forward network consists of two hidden layers with twenty nodes, whereas the recurrent neural network consisted of a single layer with ten nodes. A further difference between the two networks is the feed-back mechanism that is used in the recurrent network to feedback the predicted outcome to be used as a historical prediction. This feedback mechanism is used to determine how supplying the output of a network can be influence the prediction accuracy when compared to using the known historical movement of the object that has been used in the feed-forward network. The structure of the two networks that will be implemented are shown in Figure 3.5.



(a) Feed-Forward Neural Network



(b) Recurrent Neural Network

Figure 3.5: The structure of the neural networks that will be implemented for the comparative study with the traditional and dynamic Markov models.

Each layer of the neural networks have an independent function and use-case in the network. The input layer consists of two nodes, i_1 and i_2. The two nodes are respective to the data that is inserted into the model, with i_1 being the historical or predicted data and i_2 being the current movement of the object. The second layer, known as the merge layer, is used to concatenate the

movements into a single input that will be fed into the hidden layer for computation. The hidden layer consists of nodes that will use an activation function to make a prediction based upon the data that is inserted. The chosen activation function for the neural network is the linear function. A linear combination of the weights and inputs are computed by using the following formula:

$$\text{net}(x) = b + (x_1 \cdot w_1) + (x_2 \cdot w_2) + (x_n \cdot w_n) = z \tag{3.21}$$

where b is the bias, x is the input and w the weight of a connection between the nodes. The supplied output will be a set of binary or bipolar values that are representative of an x and y co-ordinate for the predicted direction of movement. The outcome will be used to compare against the observed movement using the same conditional check as shown in Equation 3.23. Depending upon the type of network that is used to generate the prediction, it will be fed back into the network as an input in conjunction with the current movement being exhibited by the object.

The type of recurrent network that is implemented will be based upon a LSTM model (Hochreiter & Schmidhuber 1997). A LSTM network consists of internal contextual state cells that will act as a 'long-term' or 'short-term' memory cell. The output that is obtained by these type of networks are modulated by the state of the long/short-term cells and is useful for generating predictions that are dependant upon prior historical movements that have been observed, rather than using just the last movement. An LSTM network is often used in literature to classify and process a time-series prediction.

3.2.5 Datasets used for Experimentation

Two different datasets have been used for the experimentation on the Markovian models and neural networks for generating predictions on the movement of an object. The two datasets are different in the terms of their construction, the first dataset was used by Yi et al. (2016*a*) in their work for building a model to learn the behaviour of pedestrians to predict a trajectory of movement. The dataset of their work has been collated from surveillance of people moving around the concourse of a train station. Their data was collected over a one hour interval and captured the locations of $12,684$ pedestrians (the data sampled for every twentieth frame). The collection of data has been stored in text-files that consist of a list of locations and frame-keys, shown in Listing 3.1.

Listing 3.1: Example of a text file from the dataset provided by Yi et al..

```
1 525
2 122
3 0
4 530
5 144
6 20
```

The first and second line of the file represents the location of a pedestrian in pixels, whereas the third line is the corresponding frame of detection. However, to make this data useful and in a format that can be used for the Markovian models and neural networks, they need to be converted into a list of movements. The list of movements is converted by using the pixel displacement from the current and previous location, as shown in equation 3.5. Once the conversion of the text-files is completed, they require a validation check to ensure that there are no missing frames or locations. It can be seen that in some text-files a gap will occur in the detection of a pedestrians location between one frame and another. Upon investigating the individual pedestrians and frames, it can be seen that they are stationary (typically within a queue) and therefore their location is not recorded until a movement has been exhibited. In order to populate the missing data, the following formula is used,

$$f = \begin{cases} f_{-1}, & \text{if } (f - f_{-1}) > 20 \\ \textit{continue}, & \text{otherwise} \end{cases} \tag{3.22}$$

where f is the frame in question and f_{-1} is the previous frame. The missing frames and movement could potentially influence the outcome of the prediction from the neural networks and Markov models. Therefore, by ensuring all data is available in a text-file it will ensure an accurate prediction can be made when using this dataset. An example of the type of data supplied by Yi et al. is shown in Figure 3.6.

Figure 3.6: An example of the dataset provided by Yi et al.. The image shows the path taken for a pedestrian from the dataset overlaid from a sample image of a frame.

From the sample image, it can be seen that when the movement is collected for every twentieth frame a gap can be seen between the recorded locations. It could be possible that a pedestrian could change direction between the two points that have been recorded, albeit momentarily. Therefore, the video dataset discussed in Section 3.1.3 is also used in this experiment to predict the next direction of movement when a continuous collection of frames are inserted in the neural networks and Markov models. The two type of trajectories that have been used in the videos have been chosen to see how the performance of the predictions generated are influenced by the behaviour of an object following a simple movement of an object moving left to right, and whilst an object is following the path of a rectangle. The second trajectory has been included to ensure that at-least four of nine directions are met and will provide an insight in how the network and models perform in respect to a more complex movement.

3.2.6 Experimental Plan and Measurable Metrics

To experiment with the task of predicting an objects next direction of movement using the various neural networks and Markov models a collection of experiments have been formed. For this thesis, three categories have been formed and are explained:

1. **Dynamic Markov Model**
   The models probabilities are updated by using the adaptation of using a frequency matrix to count the transitions that occur. Two types of predictions are made using this model due to the dynamic nature of the probabilities:
   - *random generation*: a prediction os made using a generated random-uniform number

that is compared against the various values for a selected row from the probability vector.

- *maximum-likelihood:* a prediction is made by always selecting the maximum value from a selected row of the probability vector.

2. **Traditional Markov Model**
   The Markov model is pre-trained using a collection of historical data that has been collated from the DMM. The predictions for this model have been generated using the traditional method of prediction by generating a random number to compare against each value within the selected row of the probability vector.

3. **Neural Networks**
   A feed-forward and recurrent network will be implemented and have been discussed in Section 3.2.4. The experiments will require the networks to be trained as the observation of an objects movement is made and is done to closely resemble the process of a Markovian model. The output of the neural network will be considered to be a prediction of the next movement to occur.

Similar to the experimentation for recognising an object on mobile devices, the experiments for this section are repeated three times to remove any anomalies that may occur in the collection of results. During the experiments, a set of metrics are collected to determine the performance of the neural networks and Markov models. The following metrics are measured:

- computation time of the neural network and Markov models
- the accuracy of the predictions generated
- the similarity of the pedestrian paths using the texture analysis formulas by Haralick (1979)

The modelling is performed upon a variety of different machines to analyse how the varying hardware can affect the performance of the neural networks and Markov models. A collection of three machines have been used: a server at Coventry University, a personal computer and a BeagleBone Black. The BeagleBone hardware has been used to simulate the embedded hardware that is often found within vehicles and home-security systems. To measure the performance of the neural networks and Markov models on the prediction accuracy, it can be measured by a comparison test on the current direction of movement and the prediction generated, using the following formula,

$$\mathbf{I} = \begin{cases} 1, & \mathbf{Pre}_t = \mathbf{C}_t \\ 0, & otherwise \end{cases} \tag{3.23}$$

where $\mathbf{I}$ is an indicator of whether the prediction ($\mathbf{Pre}$) is the same as the current location ($\mathbf{C}$). The accuracy can then be calculated at the end of a simulation by using the following formula,

$$\text{acc} = 100 \cdot \frac{\sum I_j}{n} \tag{3.24}$$

where n is the total number of steps made by the pedestrian and $\mathbf{I}_f$ is the indicator for a given frame.

The amount of time taken for processing the text-file or videos will also be captured. This captured information will give an indication into whether the framework is able to compute within a real-time constraint and could also be used to assist in the calculation of the overall frame-rate for the video that are used in the second dataset. To calculate the time taken it is proposed to capture a time-stamp at the beginning of the video and then capture a timestamp at the end and apply the formula shown in Equation 3.7. The overall run-time can then be used to calculate the frame-rate, which is done by the following formula:

$$\text{fps} = \frac{\text{total frames}}{t} \tag{3.25}$$

where total frames are the total number of frames that are within the video, and t is the overall-time that was taken for processing the video.

Chapter 4

Results Analysis

In this chapter, a discussion is held on analysis of results for the various experiments that have been performed. Due to the large amount of data that is to be covered in this chapter, it has been split into five sections:

1. **Results of the Object Recognition Experiment**
   In this section, the analysis of the results for the task of recognising an object using mobile devices are discussed. Four different datasets have been used and have each been split into two categories: low and high resolution images and live-feed data and videos. The sections will provide the necessary results for the experiments that have been performed on a mixture of mobile devices and a personal computer. The conclusive study of these results will determine whether it is feasible to recognise an object in a real-time constraint on mobile devices.

2. **Results of the Pedestrian Prediction Experiment**
   In this section, a discussion is held on the results of the Markov models and neural networks that have been implemented to generate the prediction of a pedestrians next intended direction of movement. The traditional and dynamic Markov models are discussed in their respective sections, with a separate section discussing the results of the neural network.

   The section also includes the evaluative study of the dynamic Markov models probability vectors to determine whether any patterns in the vector can indicate the type of movement being exhibited by a pedestrian. Finally, a section is included to compare the results of the stochastic models with the implemented neural networks and discusses which out of the two methodologies are most suitable for predictions in real-time.

3. **Results of the Object Trajectory Videos Predictions**
   In this section, an analysis is performed upon the results of a collection of computer-generated videos. The videos were generated to understand whether the traditional and dynamic Markov models are able to be used in conjunction of the recognition algorithm to provide predictions in a real-time constraint, i.e. without impacting the frame-rate of the video.

   The section discusses each of the implemented Markov models along with a section on the processing time of the models. The latter section will conclude whether the model is able to make predictions without impacting the frame-rate of a video.

4. **Conclusion of the Object Recognition Experiments**
   This section will critique the experiments that have been performed and their overall effectiveness. The section will discuss whether an appropriate method has been found that can enable a recognition to be performed within real-time using live-feed data on mobile devices.

5. **Conclusion of the Prediction Experiments**
   This section will critique the key findings from the experiments that have been performed on

the pedestrian dataset by Yi et al. (2016*a*) and the computer-generated videos. The section will determine whether the process of recognising an object and employing a Markov model for predicting an objects next direction of movement can be employed within a real-time constraint.

4.1 Results for Recognising an Object

A collection of various datasets have been employed to evaluate the different methodologies that are used to recognise an object from its environment. In this section, the results of the experiments performed on the low and high resolution images are discussed, alongside with the results of the live-feed data and computer-generated videos. Each section will analyse the amount of time that has been taken to localise key areas of interest to formalise a descriptor. The time taken for matching these descriptors will also be discussed to provide a 'total' amount of time that was taken to perform the recognition task. This analysis is undertaken on both sets of data that have been used, images and videos.

4.1.1 Low and High Resolution Images

An observation can be made from the collection of results that have been achieved for these experiments. The resolution of an image can directly influence the amount of keypoints that have been detected from the various feature detectors used, with a linear increase being observed on the Lowe (2004) dataset. For example, SIFT on the smallest resolution image (basmati) is able to detect 489 keypoints, whereas on the highest resolution image (the scene of all objects) the detector was able to localise 969 keypoints. The results for the experiment on the low resolution images for a mobile device can be seen in Table 4.1 and shows results for the SIFT, SURF and BRISK feature detectors.

Table 4.1: The collection of results for the Lowe (2004) dataset on the Sony Z3 Tablet Compact mobile device.

SIFT				
Image	**Keypoint Detection** (s)	**Descriptor Extraction** (s)	**Matching Time** (s)	**Number of Keypoints**
basmati	0.0480	0.0640	0.1250	489
box	0.0850	0.0960	0.1460	604
book	0.0900	0.1190	0.1620	848
scene	0.2530	0.2820	–	969

SURF				
Image	**Keypoint Detection** (s)	**Descriptor Extraction** (s)	**Matching Time** (s)	**Number of Keypoints**
basmati	0.0190	0.0370	0.1410	491
box	0.0300	0.0650	0.1440	803
book	0.0450	0.0720	0.1460	889
scene	0.9480	0.1420	–	1448

BRISK				
Image	**Keypoint Detection** (s)	**Descriptor Extraction** (s)	**Matching Time** (s)	**Number of Keypoints**
basmati	0.0500	0.0240	0.2020	576
box	0.1130	0.0700	0.4500	1662
book	0.1410	0.0870	0.5760	2094
scene	0.1900	0.1240	–	2786

From the results of the experiment, it can be seen that a large number of keypoints can be detected from a relatively low resolution. The number of keypoints detected using a binary detector like BRISK provides the most amount in comparison to traditional feature detectors. This can be seen in the graph shown in Figure 4.1.

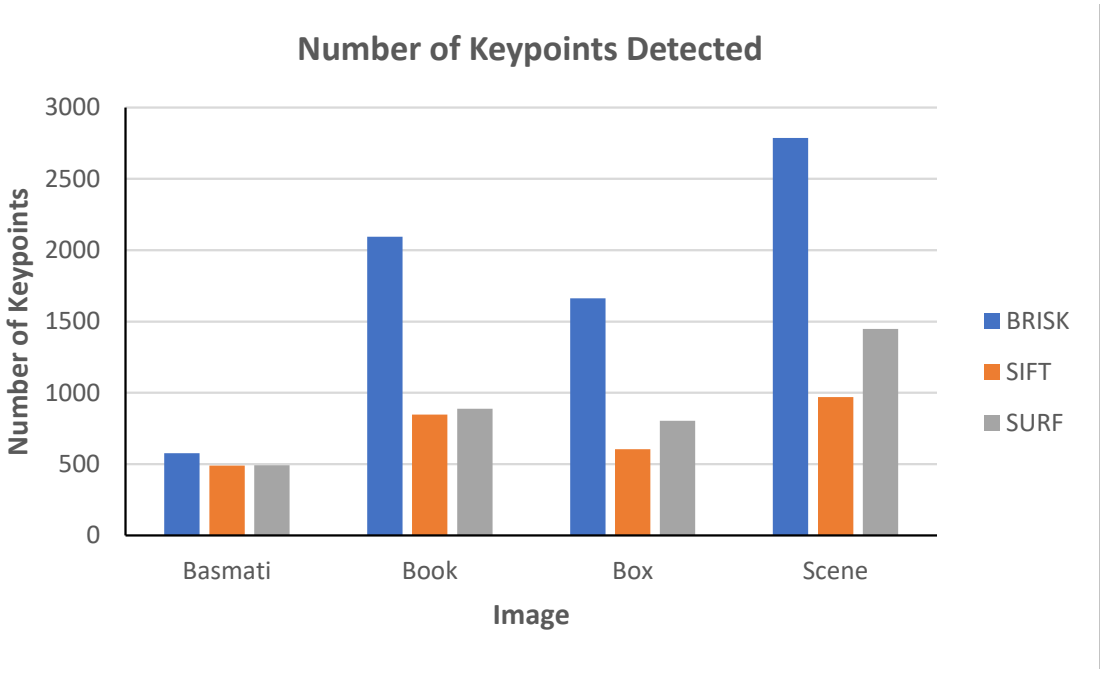


Figure 4.1: The number of detected keypoints from each image of the Lowe (2004) dataset using the three feature detectors on a mobile device.

The least amount of keypoints that were detected was achieved by SIFT. Between the two traditional detectors it can be seen that there is a larger number of keypoints detected when using the SURF detector. For example, the image of a scene achieved 33.08% more keypoints with SURF than it did with SIFT. The reason behind this could be that SURF is more receptive to the colour properties of an image, due to the large varying grey colours that are present in the image. With the increase in the number of pixels in a higher resolution, then the likelihood of a larger number of keypoints being detected is expected as there are more potential candidates for selection. However, for this linear increase in the number of keypoints detected, an increase in the time taken for detection is observed.

The increase in time can be seen for the results of SURF where from the smallest to largest image an increase of 4889.47% is observed in the amount of processing time taken. However, it can be seen from the results that BRISK performed optimally with the lowest resolution image computing within 0.05 seconds and the largest image within 0.19 seconds. It would be expected to see that BRISK would perform quicker due to the nature of how the detector was built and the sampling pattern that was used. The graph in Figure 4.2 shows the amount of time taken for detecting the keypoints for each detector used in the experiment on the mobile device.

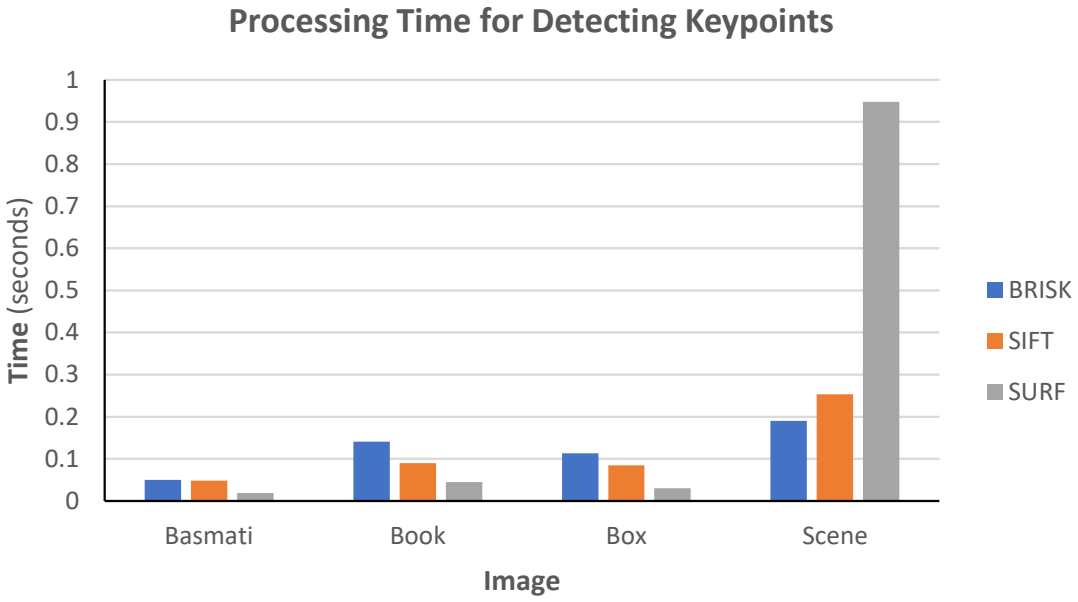


Figure 4.2: The processing time for detecting the keypoints of interest from each image of the Lowe (2004) dataset on a mobile device.

From the graph, it can be seen that the scene image takes the most amount of time for the detection of keypoints. The SURF detector is the worse performer in this instance, with the detection process taking a significant 0.948 seconds. Excluding this result, it can be seen that SIFT is the slowest detector for keypoints out of the three detectors. Surprisingly, BRISK did not perform as well on the lower resolution images in comparison to the scene image. This is interesting as SURF out-performed the detector on these images, but it becomes apparent that BRISK is better suited for larger resolution images.

The process of extracting a descriptor using the traditional detectors are observed to take longer than the detection of keypoints. It can be seen between SIFT and SURF that there is a direct correlation with the number of keypoints detected and the amount of time taken for formalising the descriptor. Between the two traditional detectors, it can be seen that SURF was the fastest for formalising a descriptor. However, in comparison to BRISK it can be seen that the two traditional detectors take a significant amount of more time to formalise a descriptor. This can be seen in the graph shown in Figure 4.3 where BRISK achieved the lowest time on two out of the four images.

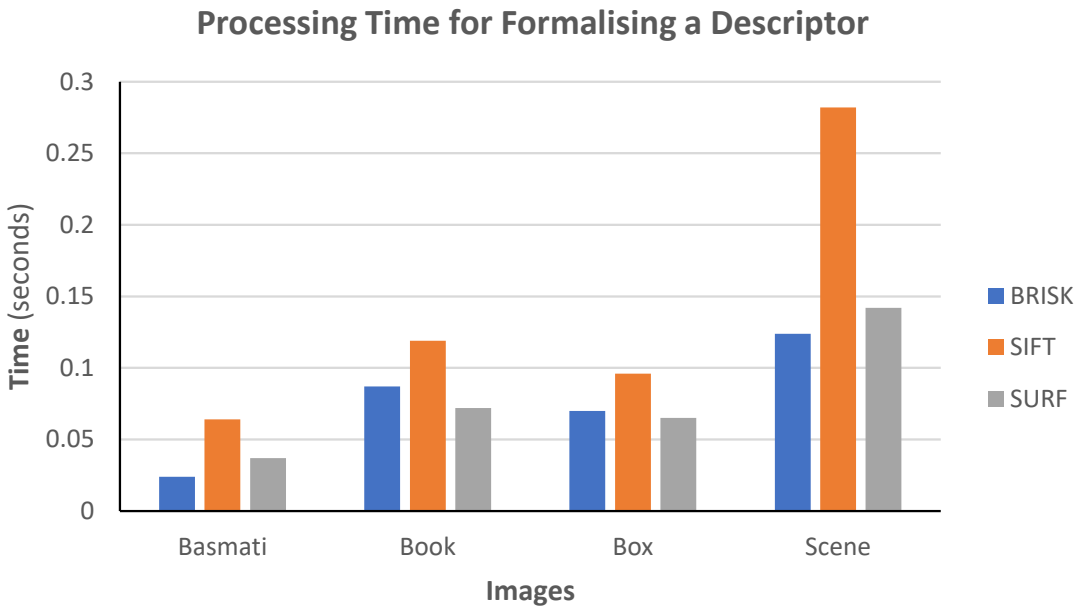


Figure 4.3: The processing time for formalising a descriptor from each image of the Lowe (2004) dataset on a mobile device.

Traditional detectors like SIFT take the longest amount of time whilst formalising a descriptor. However, SURF was considerably better, and in some instances out performed the binary detector. For example, it can be seen in the graph that the book and box image performed better on formalising a descriptor with the SURF detector than BRISK. The lower number of keypoints could of influence the amount of time taking, but it could be seen on the larger resolution that BRISK performed the best.

The collection of descriptors extracted from the image are used with a matching algorithm. The algorithm will recognise the three object images from the scene image and two different matching algorithms were used. The first method used was FLANN for SIFT and SURF, whilst the BF method was used for BRISK. The graph in Figure 4.4 shows the amount of time taken for matching the object images with the scene image.

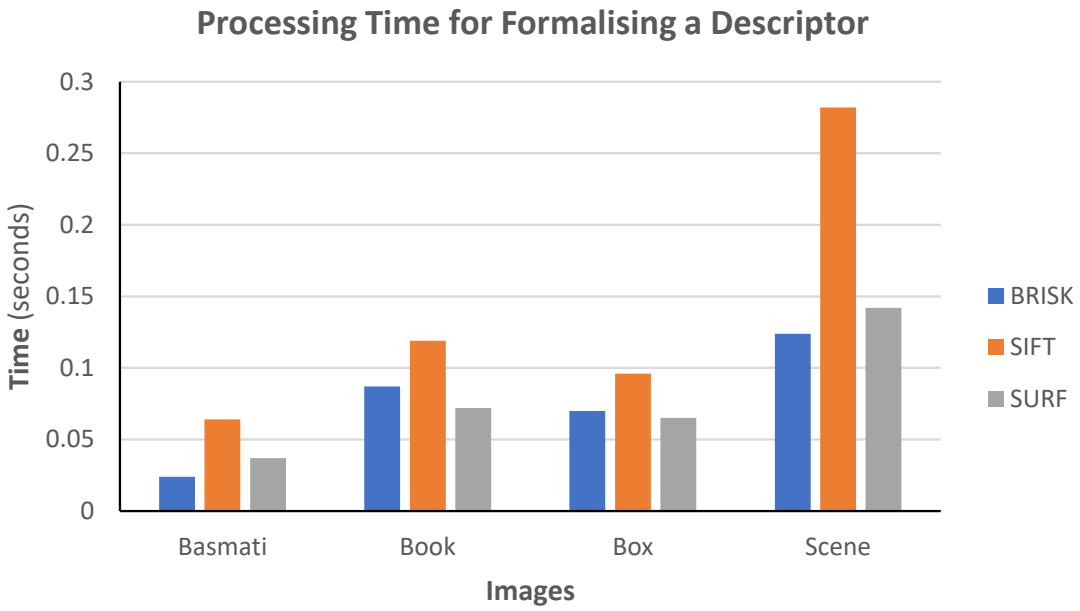


Figure 4.4: The processing time for matching the descriptors from each object with the scene image of the Lowe (2004) dataset on a mobile device.

The data for the matching algorithm is shown in Table 4.1 and is applicable to the three training images that were used. The observed times were faster for SIFT and SURF due to the algorithm that was used for matching the descriptors. FLANN is fast due to the methodology that it uses for finding the nearest neighbour. The BF matcher on the other hand is slower due to the methodology it follows for matching the descriptors by going through each pair and comparing them against one another.

The results for the Dell XPS laptop are shown in Table 4.2. The experiments are the same as those performed on the Android operating system and is developed in the C++ language to ensure that the same code can be used. From the results it can be seen that the same number of keypoints are extracted for each image, which is to be expected as the same parameters for the feature detectors are used from the Android experiment. The key motivation of running this experiment was to determine whether the hardware of a mobile device can match the power of a laptop for the task of object recognition. Therefore, the important aspect of these results are the times for detecting the keypoints, extracting descriptors and the matching between the two image descriptors. From the results it can be seen that the SIFT detector performs significantly quicker on the laptop compared to the mobile-hardware. For example, on the scene image a reduction can be seen on the processing time for detecting keypoints of 91.3%, the same was also recorded for the descriptor extraction on the same image. A reduction on extraction of a descriptor was recorded at 90.07%. It can be seen across the board of detectors a decrease was achieved on the times recorded; most notable was a decrease seen in the time taken for localising keypoints and extracting a descriptor on the scene image, the largest out of all the images.

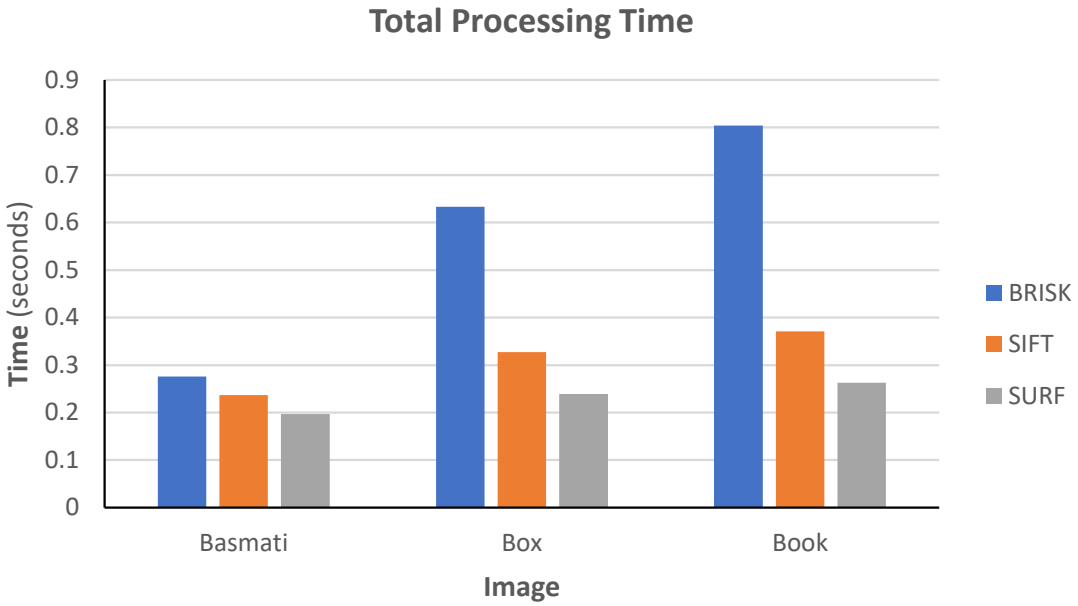
Table 4.2: Results of the experiment performed on the Linux operating system using the Dell XPS laptop, with the images originally used by Lowe (2004) when testing his feature detector.

SIFT				
Image	**Keypoint Detection** (s)	**Descriptor Extraction** (s)	**Matching Time** (s)	**# of Keypoints**
basmati	0.0330	0.0410	0.0290	489
book	0.0400	0.0470	0.0290	604
box	0.0370	0.0390	0.0280	848
scene	0.0220	0.0280	–	969

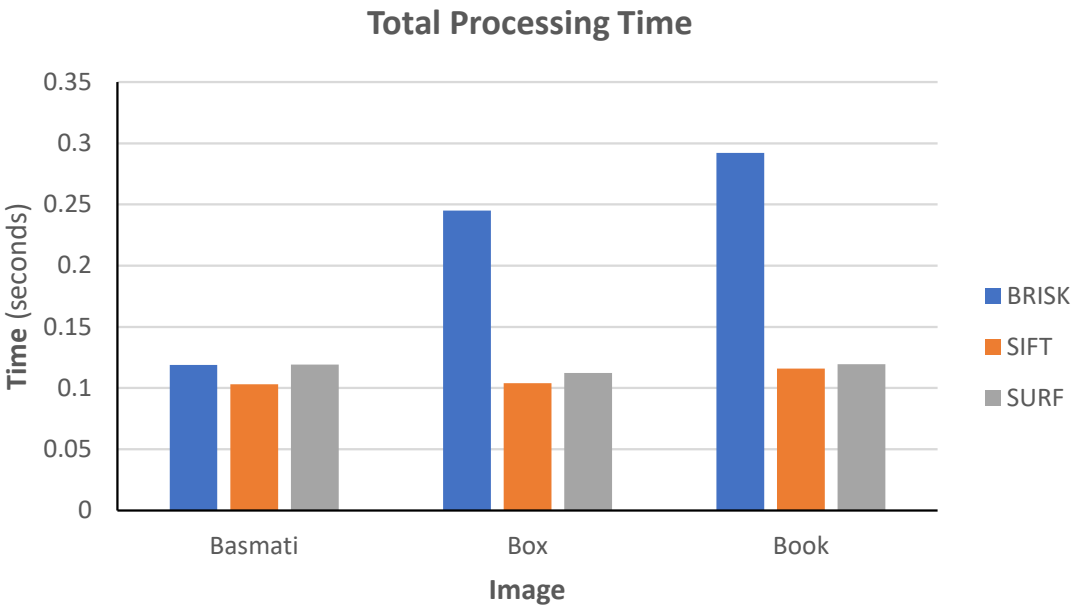
SURF				
Image	**Keypoint Detection** (s)	**Descriptor Extraction** (s)	**Matching Time** (s)	**# of Keypoints**
basmati	0.0265	0.0671	0.0255	491
book	0.0259	0.0639	0.0296	803
box	0.0284	0.0557	0.0283	889
scene	0.0200	0.0493	–	1448

BRISK				
Image	**Keypoint Detection** (s)	**Descriptor Extraction** (s)	**Matching Time** (s)	**# of Keypoints**
basmati	0.0300	0.0230	0.0660	576
book	0.0300	0.0240	0.2380	1662
box	0.0300	0.0240	0.1910	2094
scene	0.0300	0.0230	–	2786

To show the contrast between the two sets of results, the graph in Figure 4.5 plots the total processing time for each image between the two devices used. It can be seen in the figure that the laptop's abundance of power is far greater than that of the mobile hardware, especially when larger resolutions are used. The graphs show that the BRISK detector took the longest amount of time in total for recognising an object from the scene. A big part of this reason is due to the BF matching algorithm that is used to match the training descriptors with that of the scene image. The algorithm is used to find the best match between pair of descriptors to determine a robust match in the image, unlike the algorithm used for the SIFT an SURF detectors which approximates the best match between the descriptors.



(a) Mobile Device



(b) Laptop

Figure 4.5: The total computation time for each image (summed keypoint, descriptor and matching times) between the laptop and mobile hardware for each detector.

From this experiment on the Lowe dataset, it can be concluded that although the mobile hardware is still not as powerful as a laptop counter-part, it could be theoretically possible to recognise objects from a scene within a real-time constraint (on low resolution images). However, these images are not conclusive as they are so fundamentally low in the number of pixels compared to images that are now captured. Therefore, the second dataset of high definition images could provide an insight into how these feature detectors perform on images with a greater number of pixels. The computer-generated videos have a large growth in the number of pixels when the

resolution is increased, as depicted in Figure 4.6.

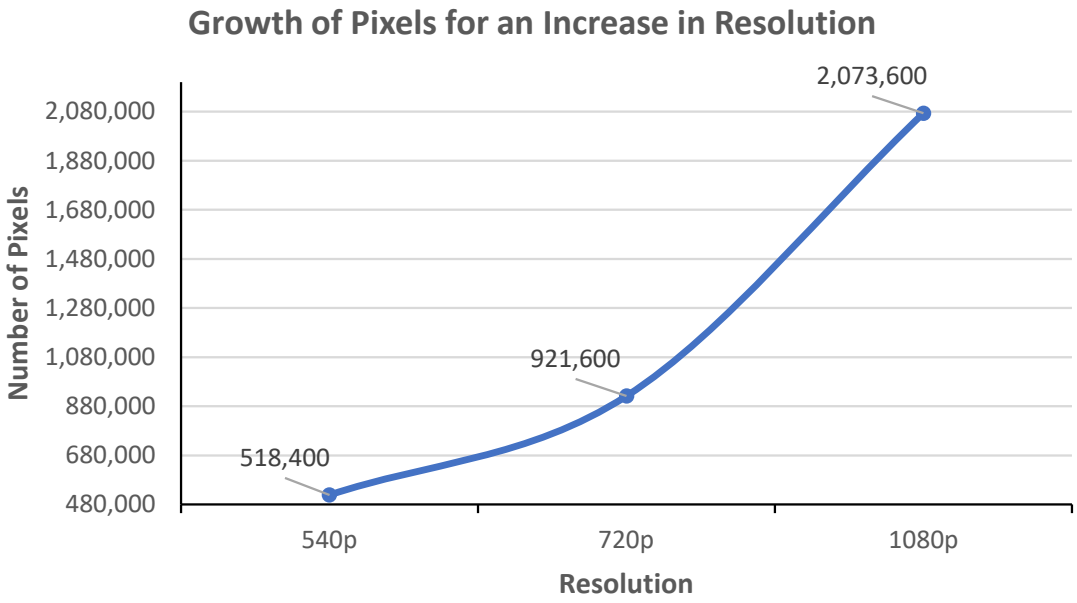


Figure 4.6: The growth in pixels for an increase in the resolution of the image is particularly significant when reaching 1080p in comparison to the 540p resolution.

It can be seen that by doubling the resolution from 540p to 1080p the pixels inside the image increase three-fold (300%), and it is expected with these significant jump in pixels that the processing time for this resolution will be significantly longer. The results for the Android experimentation are shown in Table 4.3 and it can be seen that the processing time for high resolution images for the mobile hardware is significantly impacted. The SIFT detector was the worse performer with the times for localising keypoints through the range of images for resolution 720p and 1080p exceeding a second, with the same amount of time also being taken for the extraction of descriptors. This amount of time taken for determining the key features of an image could severely impact the process of recognising the object within a real-time constraint. The times observed on the SIFT detector are to be expected due to the number of pixels that are present within the 720p and 1080p images. By contrast, when comparing the scene image of the Lowe dataset with the 540p image of the high-resolution dataset, there are $321,792$ more pixels present in the 540p image. Due to this larger number of pixels present in the image, it is then expected to see an increase of approximately 0.4 seconds between the scene image of the Lowe dataset (0.535 seconds) and the 540p image (0.927 seconds).

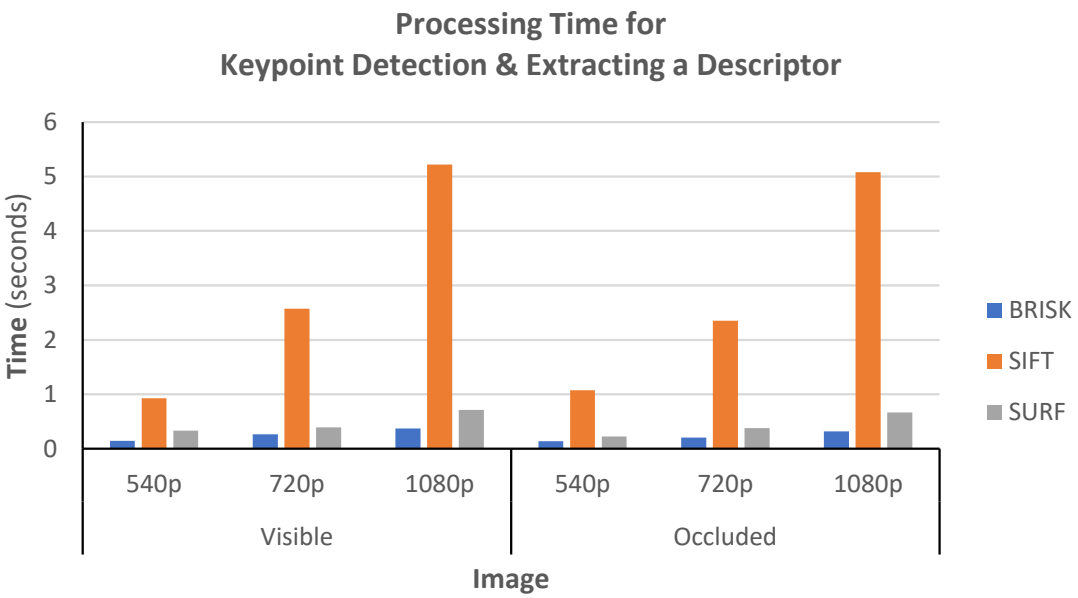
Table 4.3: Results of the experiment performed on the mobile hardware with the high-resolution images.

SIFT				
Image	**Keypoint Detection** (s)	**Descriptor Extraction** (s)	**Matching Time** (s)	**# of Keypoints**
Visible - 540p	0.528	0.399	0.076	291
Visible - 720p	1.268	1.303	0.092	439
Visible - 1080p	2.661	2.559	0.12	682
Occluded - 540p	0.646	0.426	0.07	234
Occluded - 720p	1.217	1.133	0.083	292
Occluded - 1080p	2.621	2.457	0.09	377
ID Card	0.176	0.16	–	639

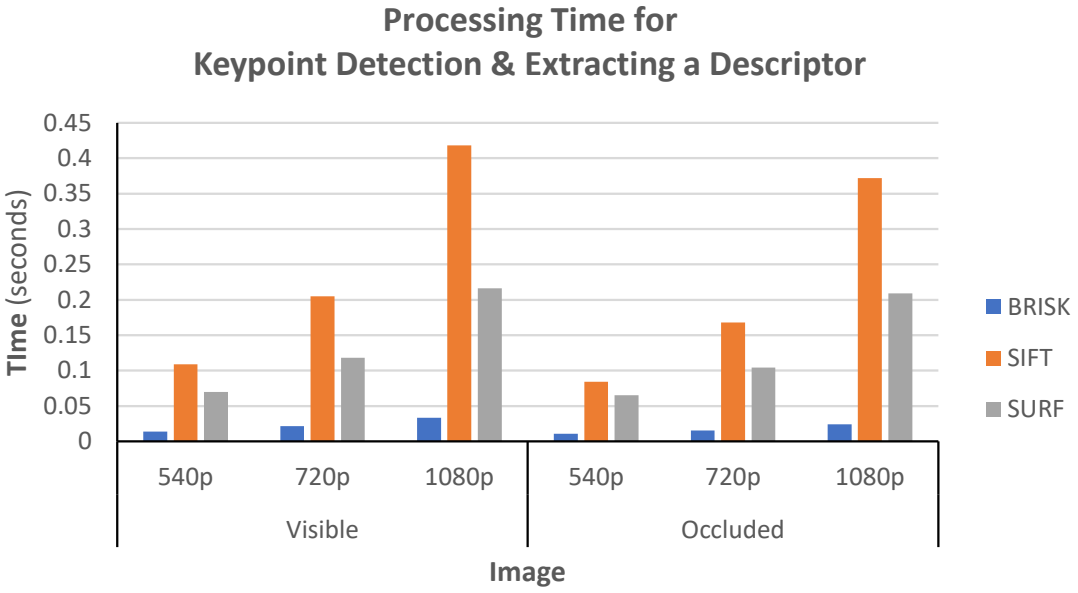
SURF				
Image	**Keypoint Detection** (s)	**Descriptor Extraction** (s)	**Matching Time** (s)	**# of Keypoints**
Visible - 540p	0.107	0.224	0.073	584
Visible - 720p	0.188	0.205	0.091	800
Visible - 1080p	0.393	0.318	0.159	1158
Occluded - 540p	0.112	0.115	0.07	515
Occluded - 720p	0.211	0.166	0.074	580
Occluded - 1080p	0.386	0.282	0.09	748
ID Card	0.063	0.082	–	660

BRISK				
Image	**Keypoint Detection** (s)	**Descriptor Extraction** (s)	**Matching Time** (s)	**# of Keypoints**
Visible - 540p	0.110	0.037	0.109	514
Visible - 720p	0.212	0.054	0.109	742
Visible - 1080p	0.305	0.067	0.174	931
Occluded - 540p	0.115	0.023	0.076	316
Occluded - 720p	0.165	0.039	0.076	361
Occluded - 1080p	0.281	0.038	0.149	391
ID Card	0.098	0.054	–	1090

It has been said the the growth of pixels between the 540p and 1080p image is 300% and therefore it would be expected to see that the computation time would increase within the same range. However, the time between the 540p and 1080p images for the SIFT and SURF detectors detecting keypoints of interest increased by 403.98% and 267.29%, respectively. Although the times of the BRISK detector were very similar to that of SURF detector, it performed better on the 1080p image and saw an increase of 177.27% in the computation time. The graph in Figure 4.7 shows the time taken for localising keypoints and extracting the descriptors from each image. The graph shows that the BRISK detector is most favourable for when extracting descriptors from an image due to the low processing time that was observed. However, it is not the same for the times recorded on detecting the keypoints from the image. The amount of time taken for this process is similar to the times obtained by SURF, with BRISK marginally better on the 1080p images.



(a) Mobile Device



(b) Laptop

Figure 4.7: The amount of time taken for detecting keypoints of interest and extracting descriptors for each detector.

However, the matching process of BRISK is still the slowest between the two algorithm used. Although the Brute-force matcher is slower as its performing a check for the best match possible it is the most robust. Although it is slower, the times are still relatively close to the FLANN matching algorithm (with the difference becoming minimal in the occluded images). Figure 4.8 shows the total amount of computation time for recognising an object from its environment using the high resolution images on the mobile hardware for the occluded scene image.

From the graph it can be seen that the SIFT detector is the worst for recognising an object

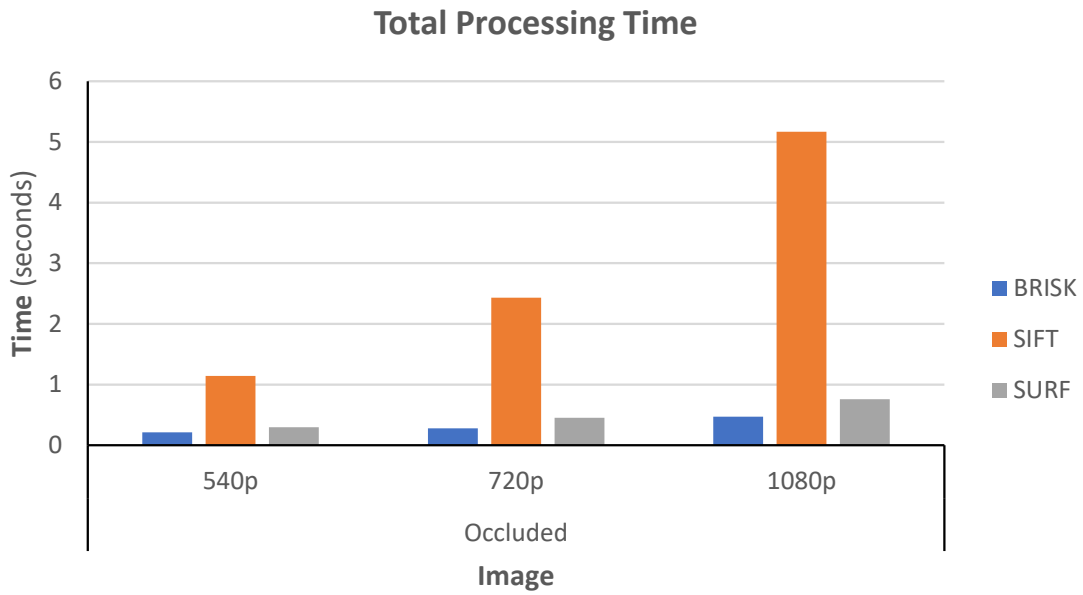


Figure 4.8: The total computation time for recognising an object from the occluded scene image for each detector.

from the high-resolution images and could therefore pose to be the worst for when experimenting on the video dataset. Although it seems on the graph the SURF and BRISK are fairly similar in terms of computational time, this is not the case as can be seen in the results of Table 4.4. The difference shown is the percentage difference between the SURF and BRISK detector and it can be seen that for an increase in the resolution the difference becomes noticeably larger.

Table 4.4: The percentage difference between the computation time for object recognition using the BRISK and SURF detectors.

Resolution	**SURF Time** (s)	**BRISK Time** (s)	**Difference**
540p	0.297	0.214	-27.95%
720p	0.451	0.280	-37.92%
1080p	0.758	0.468	-38.26%

The results in Table 4.5 show the outcome of running the experiments on the laptop in the same scenario as the previous experiment on the smaller resolution files. It can be seen that the results of the SIFT detector are significantly better on the laptop in comparison to the mobile hardware.

Table 4.5: Results of the experiment performed on the Linux operating system using the Dell XPS laptop, with the new high resolution images.

SIFT				
Image	**Keypoint Detection** (s)	**Descriptor Extraction** (s)	**Matching Time** (s)	**# of Keypoints**
Visible - 540p	0.056	0.053	0.013	291
Visible - 720p	0.102	0.103	0.016	439
Visible - 1080p	0.227	0.191	0.02	682
Occluded - 540p	0.045	0.039	0.012	234
Occluded - 720p	0.087	0.081	0.013	292
Occluded - 1080p	0.195	0.177	0.014	377
ID Card	0.016	0.021	–	639

SURF				
Image	**Keypoint Detection** (s)	**Descriptor Extraction** (s)	**Matching Time** (s)	**# of Keypoints**
Visible - 540p	0.027	0.043	0.015	584
Visible - 720p	0.047	0.071	0.018	800
Visible - 1080p	0.108	0.108	0.023	1158
Occluded - 540p	0.026	0.039	0.014	515
Occluded - 720p	0.046	0.058	0.015	580
Occluded - 1080p	0.107	0.102	0.017	748
ID Card	0.012	0.027	–	660

BRISK				
Image	**Keypoint Detection** (s)	**Descriptor Extraction** (s)	**Matching Time** (s)	**# of Keypoints**
Visible - 540p	0.0085	0.0051	0.0182	514
Visible - 720p	0.0137	0.0077	0.0253	742
Visible - 1080p	0.0224	0.0107	0.0322	931
Occluded - 540p	0.0071	0.0036	0.0109	316
Occluded - 720p	0.0108	0.0045	0.0124	361
Occluded - 1080p	0.0178	0.0062	0.0134	391
ID Card	0.0136	0.0094	–	1090

The results of the SIFT detector lay within approximately 10% of the original results of the mobile hardware. For example, the results of the Scene (540p) image on the Android hardware was recorded to be 0.528 seconds, whereas on the laptop it was recorded at 0.056 seconds which is approximately within 10% of the mobile hardware time. From the results it can be seen that the laptop hardware is considerably more powerful than the mobile hardware and is best suited for the real-time aspect of recognising an object. There is a stark difference in the computation time between the mobile device and laptop.

From the collection of results, it can be seen that BRISK is the most favourable detector for the task of object recognition with the computation times all falling below 0.6 seconds. However, SIFT and SURF both fall within 0.9 and 5.0 seconds, respectively. The BRISK detector has a significant impact on the processing time in comparison to the SIFT detector, especially when considering that BRISK is able to detect more keypoints in comparison to SIFT. For example, the SIFT detector was able to detect 682 keypoints on the image of a scene in a 1080p resolution. Whereas BRISK was able to detect 931, an increase of 249 keypoints in a similar amount of time. However, it can be seen that the SURF detector was able to extract even more keypoints than BRISK ($1,158$) in half the amount of time, 0.108 seconds. Therefore, based upon this analysis it can be seen that although BRISK is computationally least expensive on the extraction of a

descriptor, the SURF detector performs better with handling the keypoint detection.

4.1.2 Web Camera and Videos

The results in Table 4.6 show the outcome of the first experiment using the SURF detector solely for detecting an object from its environment using the in-built web-cam of the Dell XPS laptop. It can be seen from the results the the number of octaves and layers within each octave can negatively impact the frame-rate of webcam, whereby the larger the number the the lower the frame-rate becomes.

Table 4.6: The results of using the SURF only detector on a webcam inserting live-feed data.

Experiment 1: SURF Recognition			
Parameter	**Value**	**FPS**	**Object Detected**
hessian threshold	100	11	Yes (strong)
# of octave	4		
# of octave layers	2		
hessian threshold	50	17	Yes (weak)
# of octave	2		
# of octave layers	1		
hessian threshold	200	20	Yes (strong)
# of octave	2		
# of octave layers	2		
hessian threshold	400	25	Yes (strong)
# of octave	2		
# of octave layers	2		

By increasing the Hessian threshold of the detector and lowering the octave and octave layers the frame-rate of the live-feed can be steadied to almost within the original frame-rate of the webcam. As shown with the final set of parameters and the frame-rate measuring at 25 frames-per-second, only a five frame drop from the original (30). The results also show that only a strong detection is made when the number of octave layers within each octave is two and that the Hessian threshold is above the default level of 100.

Table 4.7: The results of using the BRISK only detector on a webcam inserting live-feed data.

Experiment 2: BRISK Recognition			
Parameter	**Value**	**FPS**	**Object Detected**
threshold	30	23	Yes (weak)
# of octaves	3		
pattern scale	1.0		
threshold	60	30	No
# of octaves	6		
pattern scale	1.0		
threshold	50	30	Yes (weak)
# of octaves	4		
pattern scale	0.5		
threshold	30	25	Yes (strong)
# of octaves	5		
pattern scale	1.0		

The results in Table 4.7 shows the outcome of the second experiment for solely using the BRISK detector. It can be seen that although BRISK was computationally fast at extracting descriptors on

the high-resolution dataset, when it comes to using the detector in videos it is unable to maintain a steady frame-rate when being used with a webcam. Although it was able to reach the desired frame-rate of 30 FPS, the detection of the algorithm was either non-existent or very weak when the object was moving, as shown in Figure 4.9.

(a) Bad Recognition **(b)** Good Recognition

Figure 4.9: The detection outcome using the third set of parameters for BRISK. Although a strong detection can be made as shown on the right, a weak detection can also occur (as shown on the left).

The weak detection could be due to the number of keypoints detected by the BRISK algorithm, and therefore the third experiment should be able to apply a strong detection whilst maintaining the frame-rate. The third experiment will use the best of the previous two experiments parameters for the BRISK and SURF hybrid and then determine the best parameters that can offer a robust detection and optimum frame-rate; results of this experiment can be found in Table 4.8.

Table 4.8: The results of using the BRISK only detector on a webcam inserting live-feed data.

Experiment 3: BRISK and SURF Hybrid Recognition				
Detector	**Parameter**	**Value**	**FPS**	**Object Detected**
SURF	hessian threshold	400	22	No
	# of octaves	2		
	pattern scale	2		
BRISK	threshold	30		
	# of octaves	3		
	pattern scale	1.0		
SURF	hessian threshold	600	28	Yes (strong)
	# of octaves	3		
	pattern scale	3		
BRISK	threshold	20		
	# of octaves	2		
	pattern scale	0.35		

From the results it can be seen that the two best parameters for each detector did not work as expected with the object not being able to be detected and the frame-rate below the desired FPS. However, by adjusting the BRISK and SURF parameters quite significantly a recognition can be made; albeit with a slightly different matching process used. The previous experiments used the FLANN-based matching procedure, however for this experiment the Brute-force matcher was used to find the best match. It was also found to perform quicker than the FLANN-based method. Based upon the outcomes of the results of the live-feed data, the data in Table 4.9 shows the results of the hybrid feature detector method for recognising the object in a high definition video.

Table 4.9: The results of the experiment on the high-definition video dataset using a hybrid of detectors to recognise an object.

Video	**Average FPS**	**Run Length** (s)
Square (540p)	20.90	431.70
Square (720p)	13.59	662.09
Square (1080p)	6.55	1374.78
Straight (540p)	21.08	427.99
Straight (720p)	12.69	721.33
Straight (1080p)	6.88	1307.79

It can be seen from the results that the ability to recognise an object using a hybrid of the SURF and BRISK detectors is unable to meet the recorded frame-rate of the video (30 FPS). The best result obtained was on the video of the object moving within a straight line where the frame-rate reached 21.08 frames-per-second. Due to this drop in frames it extended the run length of the video to 428 seconds. The performance for each subsequent resolution becomes worse with the frame continuously dropping significantly. For example, the video of an object moving within the axes of a square shows that the frame-rate drops from 20.9 FPS to 6.55 when the resolution is doubled, this is a drop of approximately 68.6% (-14.35 frames). This sort of drop could severely impact the possibility of being able to predict within a real-time constraint as the prediction is made on a frame that is fourteen frames behind. The graph in Figure 4.10 shows the linear increase in the computation time for each resolution and video.

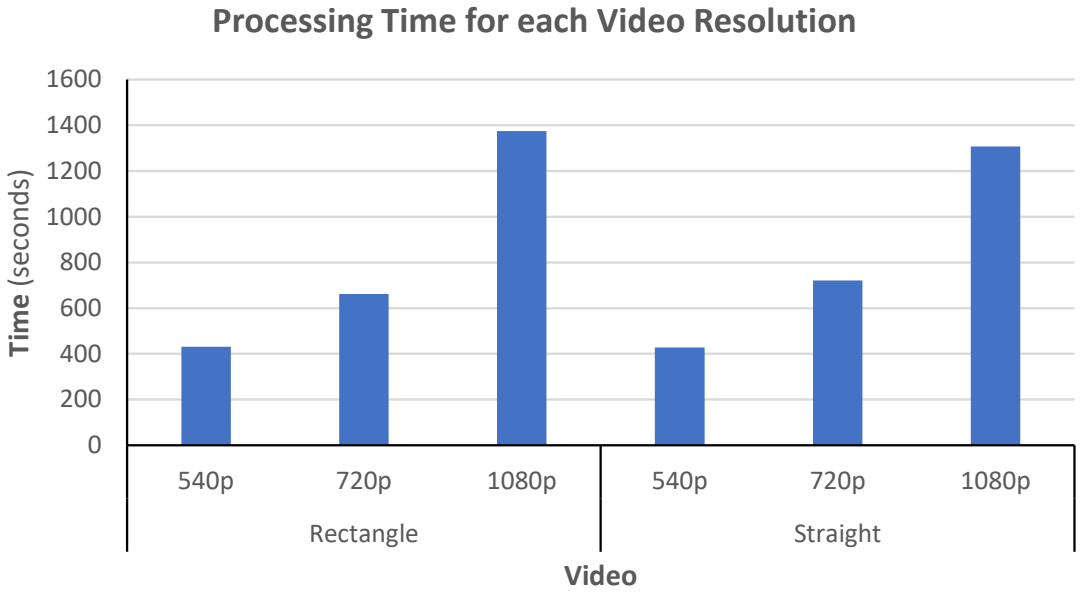


Figure 4.10: For an increase in the resolution of the video the computation time also increases. The original time of the video was 300 seconds.

From the graph, it can be clearly seen the exponential growth in the computation time when the resolution of the video is increased. The original run-length of the video was 300 seconds (five minutes) an it can be seen that even on the lowest resolution this original run-length was not met. Interestingly, the movement of an object in a straight line saw a more gradual increase in the time taken, whereas the more complex movement time saw a larger spike on the 1080p video file (although marginal). The results in Table 4.10 shows the results of solely using the BRISK detector for detection of the keypoints and extraction of a descriptor.

Table 4.10: Results for the experiment using solely the BRISK feature detector on a collection of high-definition video files.

Video	**Average FPS**	**Run Length** (s)
Square (540p)	14.29	630.36
Square (720p)	10.41	864.21
Square (1080p)	6.76	1331.37
Straight (540p)	15.05	598.59
Straight (720p)	10.65	846.46
Straight (1080p)	7.17	1255.58

Initially it can be seen that the sole use of the BRISK detector has a further impact upon the frame-rate of the video, with the 540p videos coming off significantly worse than the previous experimentation. The FPS achieved using BRISK for this resolution was 14.29 and 15.05 FPS, respectively; whereas with the previous experiment the FPS was 20.90 and 21.08, this is a delta of 6.61 and 6.03 frames. The difference in frame-rate was to be expected due to the previous experiments on static images whereby BRISK is slower for the detection of keypoints from the image. However, an improvement can be seen on the highest resolution file, 1080p; whereby a small increase could be seen on the frame-rate using solely just the BRISK detector. Although the increase was nominal, it does show that there is some advantage to just using a singular detector for extracting features opposed to using a hybrid. However, the benefits of using the hybrid far outweighs using a single detector as the overall FPS of the videos were considerably better to that achieved using the BRISK detector.

4.2 Results of Predicting a Pedestrians Movement

The first dataset used for the experiments consisted of $12,684$ pedestrians. Due to this large number of experiments that have been performed upon this dataset an expanse collection have been recorded. Therefore, some of the data can be discarded as they are considered to be redundant. There are a large amount of pedestrians that have a small number of steps recorded, whilst pedestrians with a larger number of steps are a rarity. The collection of data shown in Table 4.11 shows the mean accuracy of predictions for an imposed limit on the number of steps that have been observed.

Table 4.11: The mean accuracy of the dynamic stochastic model when applying a threshold to the number of steps that have been observed by the model.

Step Limit	**Mean Accuracy** (%)				
	Order One	**Order Two**	**Order Three**	**Order Four**	**Order Five**
4450	85.46	88.48	90.21	91.32	91.85
2225	85.45	88.48	90.21	91.32	91.85
1113	85.45	88.47	90.21	91.32	91.84
556	85.43	88.46	90.19	91.3	91.82
278	85.47	88.45	90.15	91.26	91.77
139	85.6	88.48	90.14	91.22	91.72

From the data in the table, it can be seen that by halving the number of steps each time sees a minimal decrease in the accuracy of each order. The accuracy range for a first order model drops from 85.46% to 85.43% ($4,450$ to 556 steps). However, when a further drop is made to 278 and 139 steps the accuracy increases (85.47% and 85.60%, respectively). The same increase occurs on a second order model, whereby decreasing the steps from 278 to 139 saw an increase of 0.03%. However, the higher-order models remain unaffected with a linear decrease being seen.

The mean accuracy of each step limit is shown in the graph in Figure 4.11. A linear decrease can be seen in the graph when decreasing the number of steps, although it is marginal. However, on the lower orders of the Markov model for a lower step limit an anomaly can be seen. An increase in the accuracy is observed and these anomalous results could be due to the number of previous movements that are collected to generate a prediction.

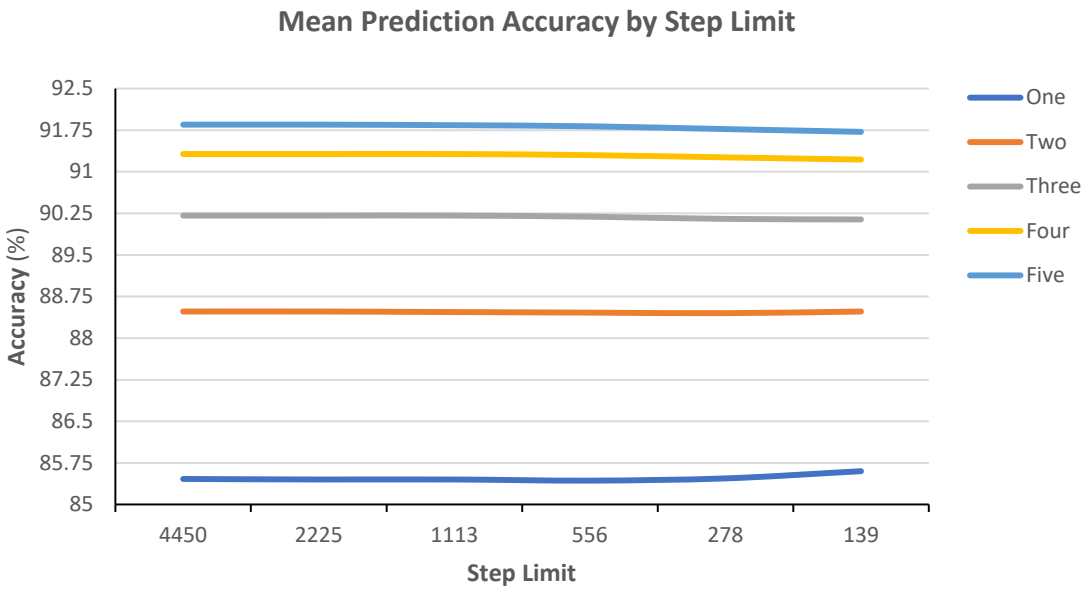


Figure 4.11: The mean accuracy of the various step thresholds used in Table 4.11.

A first order model only uses the single past movement in order to generate a prediction and therefore is most effective when trying to predict a simple path of movement. However, this will not be as accurate for paths that can be described as being complex, i.e numerous changes in direction. These results are considered to be anomalous as they are not in-keeping with the progression of change that is seen on the three higher orders. The third order model saw a decrease in the accuracy of the model, with a difference of 0.07% when applying a threshold from $4,450$ to 139 steps. Similar differences have been observed with the fourth and fifth model (-0.10% and -0.13%, respectively) and are considered not to impact the overall model significantly. The mean decrease between the three higher orders is seen to be less than 1% and therefore negligible. Due to the overall accuracy for each order being high it can be suggested that applying a conditional threshold of 305 steps will not impact the results detrimentally and can reduce the number of pedestrians for analysis from $12,684$ to $11,647$ (a reduction of 8.18%). To further back up the suggestion of applying a threshold, an analysis is performed on the pedestrians by categorising them into a collection of step ranges, as shown in Table 4.12.

By breaking down the pedestrians into their respective step ranges, it can provide a good indication where a majority of the pedestrians within the dataset are situated. It can be seen from the data in Table 4.12, a majority of the pedestrians are falling within the step ranges of $5-25$ and $25-45$, approximately 77.82% of the dataset. By including the pedestrians within ranges $45-105$ this suddenly increases to 95%. When increasing the order of the model, certain step ranges can reduce the amount of pedestrians, whereas others see an increase. This would be due to the number of initial steps that are taken at the beginning of the model to make the first prediction. These initial steps are trimmed from the file and therefore reduces the amount of steps used to generate predictions. For example, a fifth order model requires five previous steps to generate a prediction. Therefore, five steps are removed from the total number and this could bring a pedestrian from a category it once belonged in an order of one to a lower category in the fifth order model.

Table 4.12: The various categories of step ranges and the number of pedestrians that fall within each step category.

Step Range	Number of Pedestrians				
	Order One	**Order Two**	**Order Three**	**Order Four**	**Order Five**
$5-25$	3884	4110	4330	4581	4819
$25-45$	5180	5189	5166	5102	5052
$45-65$	1803	1580	1402	1230	1052
$65-85$	192	187	176	166	160
$85-105$	111	104	99	101	100
$105-125$	85	86	85	81	82
$125-145$	59	60	62	61	63
$145-165$	58	58	58	58	54
$165-185$	33	33	30	30	31
$185-205$	49	47	48	48	47
$205-225$	41	44	44	44	44
$225-245$	39	37	36	34	34
$245-265$	33	37	39	40	40
$265-285$	43	42	41	40	40

Although it can be seen that the step ranges that are greater than 105 steps consist of few pedestrians, they are important for the analysis of this work. Having as many pedestrians as possible is imperative in this study to determine whether the number of steps expressed by pedestrians can influence the accuracy of a model. By limiting the steps to 105 could restrict the analysis and conclusion of the study. A selection of various hardware have been used to determine whether the difference in the CPU and RAM could impact the time and accurateness of the model. Upon reviewing the data for the dynamic Markov model whilst reducing the amount of data for analysis it can be seen that the accuracy rates obtained from each device do not deviate significantly. Although as expected, the computation times for processing do increase and therefore it has been decided to normalise the prediction accuracies across the three devices to be a singular entity for analysis. With this further refinement occurring using the steps outlined above, a reduction was made to the number of steps within a range of 5 to 305 steps.

4.2.1 Dynamic Markov Model

The experiments for the dynamic Markov model have used two different methods for prediction: random and maximum-likelihood. The results for the random prediction method have been subjected to a statistical descriptive analysis to derive the minimum, mean and maximum of the prediction accuracy, alongside the standard deviation and delta between the minimum and maximum. The results of the test are shown in Table 4.13 and it can be seen that the model has a high accuracy rate for predictions, with the mean of the model ranging between 85.45% and 91.78%, depending upon the order of the model.

Table 4.13: The descriptive statistics of the dynamic stochastic model using the random prediction generation method for pedestrians within 5 to 305 steps.

Model Order	Accuracy (%)			Standard Deviation	Delta
	Minimum	**Mean**	**Maximum**		
One	41.26	99.56	85.45	10.43	58.31
Two	53.12	100.00	88.44	7.48	46.88
Three	65.74	99.43	90.16	5.83	33.69
Four	70.83	100.00	91.27	4.97	29.17
Five	66.67	99.47	91.78	4.74	32.80

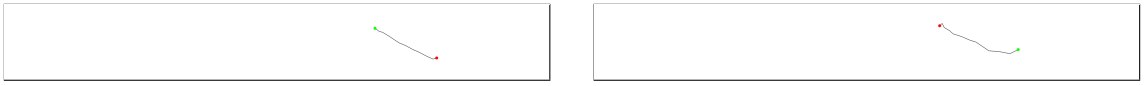
The statistics show that the minimum accuracy of the model ranges from 41.26% to 70.83% for all five orders. It is noted that between the fourth and fifth models the minimum accuracy decreases from 70.83% to 66.67% which goes against the linear increase that is seen in the mean accuracy of the model. To determine why this decrease has occurred, the pedestrians with the minimum accuracy gained for each order is extracted and shown in Table 4.14.

Table 4.14: A selection of pedestrians with a minimum accuracy (highlighted in bold) are analysed.

Pedestrian	**Accuracy** (%)					**# of Steps**				
	One	**Two**	**Three**	**Four**	**Five**	**One**	**Two**	**Three**	**Four**	**Five**
5860	**41.26**	64.19	82.87	94.02	96.61	122	121	120	119	118
6344	48.46	**53.12**	72.98	80.14	84.95	83	82	81	80	79
8382	79.37	79.49	**65.75**	77.78	92.22	14	13	12	11	10
6642	73.74	76.67	82.72	**70.83**	85.71	11	10	9	8	7
1618	84.44	77.78	79.17	77.78	**66.67**	10	9	8	7	6
2533	84.44	86.42	75	74.6	**66.67**	10	9	8	7	6
2988	70	80.25	77.78	71.43	**66.67**	10	9	8	7	6
3300	80	80.25	84.72	74.6	**66.67**	10	9	8	7	6
5559	83.33	82.72	81.94	74.6	**66.67**	10	9	8	7	6
6447	86.67	77.78	75	80.95	**66.67**	10	9	8	7	6
6475	80	77.78	84.72	79.37	**66.67**	10	9	8	7	6
7602	81.11	85.19	77.78	80.95	**66.67**	10	9	8	7	6
8192	87.78	81.48	83.33	74.6	**66.67**	10	9	8	7	6
8481	84.44	80.25	84.72	71.43	**66.67**	10	9	8	7	6
8602	80	81.48	84.72	71.43	**66.67**	10	9	8	7	6
9875	80	81.48	77.78	77.78	**66.67**	10	9	8	7	6

The table shows the range of accuracies gained for each order of the model and the number of steps that have been recorded. It can be seen from the table that pedestrian #5,860 gained the lowest accuracy for a first order model but had a substantial number of steps for analysis (118 to 122). However, the pedestrian did see a subsequent increase in the accuracy of predictions when the order of the model was increased. The same statistics can be seen seen with pedestrian #6,344. Although they gained the lowest accuracy for the second order model there is a linear increase in the accuracy of predictions. A common denominator can be seen with the fifth order model, and could explain why such a low accuracy was recorded; the number of steps for this model are all less than twelve.

Due to the number of steps decreasing when the order of the model is increased, it could theoretically be possible that the accuracy of the predictions will also decease as there are a fewer number of steps available to make a prediction on and let the model 'learn' as time progresses. However, the accuracy rate could also potentially decrease due to the complexity of movement that has been exhibited by the pedestrian. Therefore, to analyse whether the complexity of movement has impacted the results of the fifth order model, four pedestrians from Table 4.14 have been selected and their paths drawn and shown in Figure 4.12.



(a) Pedestrian #1,618 **(b)** Pedestrian #5,559

Figure 4.12: Two of the four pedestrians chosen for analysis to determine why they gained the lowest accuracy for a fifth order model.

From the figure of the pedestrians path it can be seen that the two pedestrians have a similar path. Pedestrian #1,618 is recorded to see a decrease in the accuracy of predictions made, going from 84.44% to 66.67% when the order of the model is increased. The path of the pedestrian is

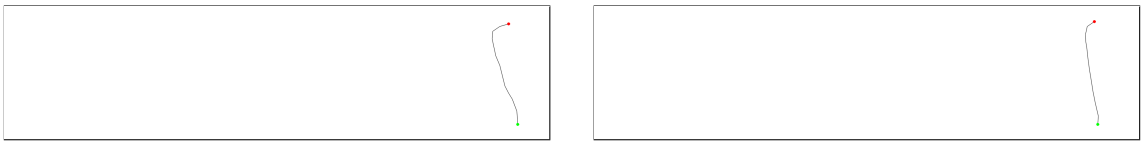
fairly linear, with it moving in a diagonal straight line from the north-west to south-east; therefore, it is expected to see the accuracy rate improve as order of the model increases. To understand why the accuracy rate decreases, the probability matrix for the pedestrian is analysed. It can be seen for the first order only a single row of the matrix was re-calculated, as shown in Equation 4.1, whereas the fifth order matrix had multiple rows adjusted, as shown in Equation 4.2.

$$\begin{array}{c} \\ \cdots \\ [(1,1)] \end{array} \begin{array}{c} \begin{array}{ccccccccc} [-1,-1] & [-1,0] & [-1,1] & [0,-1] & [0,0] & [0,1] & [1,-1] & [1,0] & [1,1] \end{array} \\ \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.1\dot{1} & 0 & 0.89 \end{bmatrix} \end{array} \tag{4.1}$$

$$\begin{array}{c} \\ \cdots \\ [(1,1)\cdot 5] \end{array} \begin{array}{c} \begin{array}{ccccccccc} [-1,-1] & [-1,0] & [-1,1] & [0,-1] & [0,0] & [0,1] & [1,-1] & [1,0] & [1,1] \end{array} \\ \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0.8 \end{bmatrix} \end{array} \tag{4.2}$$

Due to the larger selection of rows to chose from within a fifth order model, it is expected to see the accuracy to decrease; and due to the larger selection of states to transition. For example, the first order model has one of two states it could transition to: $(1, -1)$ or $(1, 1)$. Whereas in the fifth order model the pedestrian has one of two states that the pedestrian could transition to; with a slight variation in the probabilities. Because the past movements recorded only take into account the direction $(1, 1)$ it can be seen that the path does not always exhibit this type of movement; and therefore at the end when the pedestrian is moving east for a short while it will always think it is going south-east, as the probability of this transition is 80%.

Pedestrian $\#5,559$ has a similar path to $\#1,618$, with a small elongation at the beginning of the journey where they are momentarily going towards the south. However, the majority of their path is them moving towards the north west. A similar behaviour was exhibited with the model whereby the accuracy of the predictions were decreasing from relatively high (83.33%) to the lowest accuracy for the fifth order (66.67%), a difference of -16.66%. As aforementioned it is expected to see the accuracy rate of the predictions to drop due to the increased complexity of the matrices. For any pedestrian where a sudden change in direction is exhibited it is most likely that this change can be captured with the first order model. However, the fifth order model is unlikely to do so due to the number of past movements that have been taken into consideration, and is more likely to give the wrong prediction. The pedestrian paths shown in Figure 4.13 are for pedestrians $\#8,192$ and $\#9,875$ and similar to the previous pedestrians they both exhibit the same path and a low number of steps.



(a) Pedestrian $\#8,192$ **(b)** Pedestrian $\#9,875$

Figure 4.13: Two of the four pedestrians chosen for analysis to determine why they gained the lowest accuracy for a fifth order model.

The pedestrians are moving within a similar direction; with their path beginning in the south and moving towards a north-west direction, before a change in direction to the north-east. Pedestrian $\#8,192$ sees a decrease in the accuracy of predictions for an increase in the order of the model (similar to the previous pedestrians); however, the second pedestrian has a different pattern. It can be seen that pedestrian $\#9,875$ sees an increase in the accuracy from a first to second order model. The third and fourth models stagnated (the same accuracy was recorded) and the accuracy then decreased with the fifth. The stagnation of the prediction accuracy could be due to the change of direction towards the end of the pedestrian's path. It can be seen that pedestrian $\#9,875$ has a longer path going towards a north-east direction and therefore there was more time for the model

to adjust to this new change of direction. The lowest accuracy overall from the dynamic stochastic model was achieved in the first order, by pedestrian #5, 860. The accuracy rate for the first order model was 41.26% but increased rather significantly for the second order and subsequent models. To determine why a low accuracy was recorded for the first model and higher on the other models the path will be analysed, and is shown in Figure 4.14.

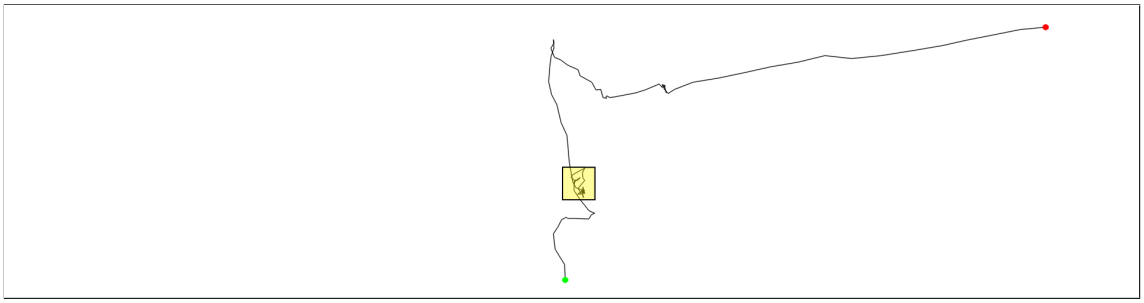


Figure 4.14: The path of pedestrian #5, 860 to be analysed and determine why a low accuracy rate was gained for the first order but a significant progress was made on the latter orders.

Between the first and second order and increase of 22.93% was achieved in accuracy for this model and by observing the path of movement by the pedestrian it can be seen that the path is fairly straight in parts but rather chaotic in others (as highlighted in the yellow). The chaotic movement can be quite difficult to predict due to the constant changes of direction that is being exhibited by the pedestrian and therefore would be inherently difficult for a first order model to predict. Due to the constant changes in direction and the model constantly being updated for each movement observed it can alter the probability matrix to a point where it would require significant movement in one direction to rectify. However, this problem is not present with the fifth order model due to the increased number of past movements that are taken into consideration to make a prediction. The higher order models are not adjusted as much as the first order model due to the larger selection of rows. Therefore, the probabilities will not need as much 'learning' in order to re-correct themselves and this would be the reason as to why the higher order models see a significant increase in the accuracy of the model.

The pedestrian shown in Figure 4.15 saw an anomalous result for their third order model. Pedestrian #8, 382 saw an increase between the first and second order (+0.12%) and a sudden decrease to 65.75% (−13.74%).

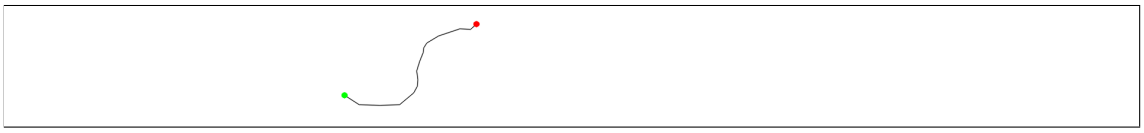


Figure 4.15: The path of pedestrian #8, 382 to be analysed and determine why a low accuracy rate was gained for the third order, when the previous and latter two orders saw a linear increase.

The path of the pedestrian is akin to the letter 's', however the overall length is relatively short with the pedestrian only moving between 10 to 14 steps. The range of movements made are within three different directions: left, down and up. The actual states of transition that were visited are: $(-1,-1),(1,-1)$ and $(1,1)$. The three types of movement collected could impact the accuracy for a first order mode as three rows within the matrix are adjusted to account for each direction. However, for a third order model the number of rows adjusted in the probability matrix are six, as shown in equation 4.3.

$$
\begin{array}{l}
 \\
\cdots \\
[(1,-1),(1,-1),(1,1)] \\
[(1,1),(1,-1),(1,-1)] \\
[(-1,-1),(1,-1),(1,-1)] \\
[(1,-1),(1,-1),(-1,-1)] \\
[(1,-1),(-1,-1),(1,-1)] \\
[(1,-1),(1,-1),(1,-1)]
\end{array}
\begin{array}{c}
\begin{array}{ccccccccc}
[-1,-1] & [-1,0] & [-1,1] & [0,-1] & [0,0] & [0,1] & [1,-1] & [1,0] & [1,1]
\end{array} \\
\left[\begin{array}{ccccccccc}
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0.167 & 0 & 0 & 0 & 0 & 0 & 0.667 & 0 & 0.167
\end{array}\right]
\end{array}
\tag{4.3}
$$

As it can be seen, there is a fair few more rows that the previous direction have movement has populated and therefore this would account for the dip in accuracy for the third order model. When analysing the matrix of the fourth and fifth model it can be seen that these two matrices saw an increase in the number of rows to select (7 and 8, respectively) and the probabilities were similar. With the larger number of past movements that are used to generate a prediction, it is expected to see a difference in the number of rows that are adjusted in the matrix. However, due to the lower number of steps that are exhibited by the pedestrian then the low accuracy gained is to be expected. However, if the number of steps observed were to grow, it is known that the model will be able to alter the probabilities suitably and therefore the accuracy will increase. It can be seen from the results in Table 4.14 that the models order can influence the accuracy of prediction, depending upon the type of movement that has been exhibited and the number of steps. For example, pedestrian #1,618 had a large number of steps observed and saw a significant linear increase in the accuracy of predictions when the order was adjusted respectively. Therefore, to determine whether there is a correlation between the number of steps and the accuracy of the model a correlation test was performed using the Pearson's (Pearson 1896) correlation test, shown in Table 4.15.

Table 4.15: The statistics for the Pearson's correlation test using the IBM SPSS statistics software.

Order	One	Two	Three	Four	Five	
One	−0.158	−0.040	0.089	0.216	0.340	**Steps**
Two	−0.158	−0.040	0.089	0.216	0.340	
Three	−0.158	−0.040	0.089	0.216	0.340	
Four	−0.158	−0.040	0.089	0.216	0.340	
Five	−0.158	−0.040	0.089	0.216	0.340	
	Accuracy					

The results from the correlation test are weak and show that the number of steps observed by the pedestrians do not impact the accuracy rates of the predictions generated. It is assumed that instead of the steps affecting the accuracy rate of the model it is the general path of the pedestrian that can influence the outcome of the model. As it has been mentioned previously with the several pedestrians, it can be seen their path have a direct influence on the accuracy of the model; with some orders being weaker at predicting a particular path opposed to the other. For example, a pedestrian with a simple, straight path will perform better when using a lower order model (typically a first order) whereas more complex paths work better with a higher order model. Overall, it can be seen that by increasing the accuracy of the model does have a positive impact on the accuracy as a linear growth can be seen across the mean accuracy of the model, as shown in Table 4.13. To determine whether the categories of step ranges have any influence upon the accuracy of the predictions, the graph in Figure 4.16 has been created to show how the mean accuracy can fluctuate depending upon the range of steps that have been exhibited.

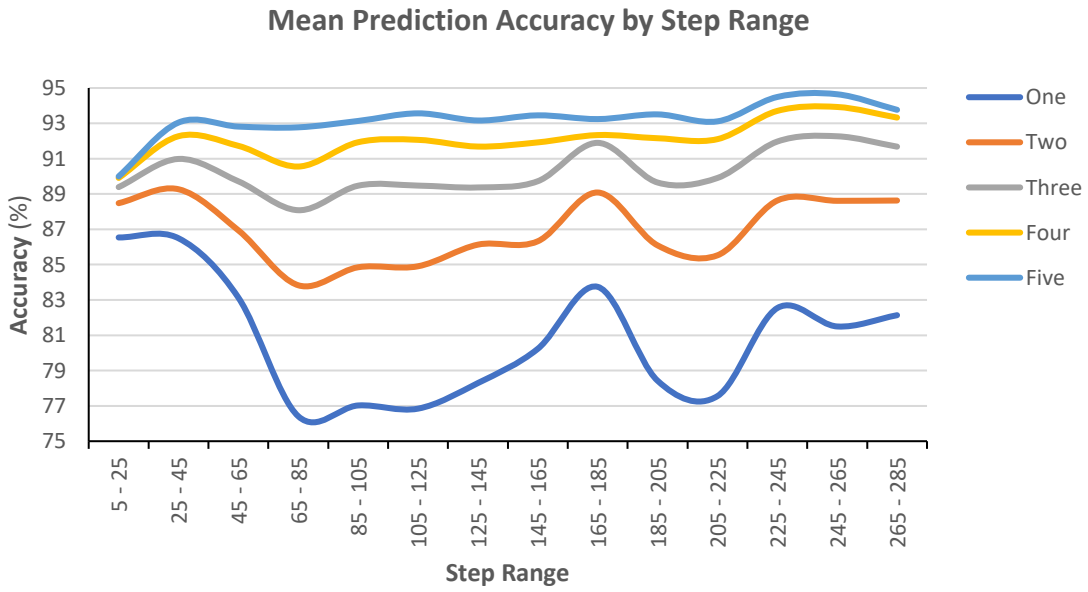


Figure 4.16: The mean accuracy for pedestrians within their respective category of step ranges for the dynamic stochastic model using the random prediction method.

It can be seen from the graph that there is no straight linear increase when the step ranges increase. There is a degree of instability with the model as there are numerous peaks and troughs within the lower orders. The first and second order models are significantly the worse for stability with the prediction rate dropping significantly between ranges 5 to 85, with it then increasing until it reaches 205 steps. The peaks and troughs within the graph would be attributed to the number of pedestrians within each category; with a lower number of pedestrians having a far greater influence on the mean accuracy in comparison to a step range that has a larger number of pedestrians. The instability of the predictions for orders one to three could be due the complexity of the paths that fall within the lower step-ranges. It can become less and the line becomes straighter and a small linear increase can be seen.

The results for the maximum-likelihood method have been subjected to a statistical descriptive analysis, similar to that was performed within the random prediction method section. The results of this test can be found in Table 4.16, and it can be seen that the maximum accuracy achieved was 100%, whilst the minimum accuracy rate saw an increase in comparison to the random prediction method.

Table 4.16: The descriptive statistics of the dynamic stochastic model using the maximum-likelihood prediction generation method for pedestrians within $5 - 305$ steps.

Model Order	**Accuracy** (%)			**Standard Deviation**	**Delta**
	Minimum	**Mean**	**Maximum**		
One	50.00	88.81	100.00	8.27	50.00
Two	58.82	90.76	100.00	6.29	41.18
Three	63.64	91.98	100.00	5.03	36.36
Four	62.50	92.73	100.00	4.52	37.50
Five	66.67	93.09	100.00	4.47	33.33

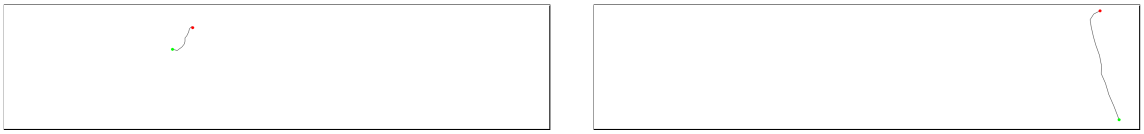
The minimum accuracy of the model was recorded to be between 50% and 66.67%. A linear increase can be seen across the five orders of the model with the minimum and mean. It can be seen that the delta (the difference between the mean and maximum) decreases between orders one and three, with a small increase on the fourth order model. To determine why this anomaly occurs

within the results, the pedestrians that gained the minimum accuracy recorded for the model are analysed; the pedestrians for this order are shown in Table 4.17.

Table 4.17: The pedestrians that gained the minimum accuracy for the fourth order model when using the maximum-likelihood prediction method.

Pedestrian	**Accuracy** (%)					**Number of Steps**				
	One	**Two**	**Three**	**Four**	**Five**	**One**	**Two**	**Three**	**Four**	**Five**
3098	72.73	70.0	77.78	62.5	85.71	11.0	10.0	9.0	8.0	7.0
8512	72.73	70.0	77.78	62.5	100.0	11.0	10.0	9.0	8.0	7.0

Two pedestrians gained the lowest accuracy for this order, and they both had the same number of steps observed. The paths for each pedestrian are shown in Figure 4.17 and it can be seen that they both have different paths; with the path by #3,098 moving north-east and pedestrian #8,512 moving north-west and then deviating towards the north-east.



(a) Pedestrian #3,098 **(b)** Pedestrian #8,512

Figure 4.17: The two pedestrians that gained the lowest accuracy for the fourth order model using the maximum-likelihood method.

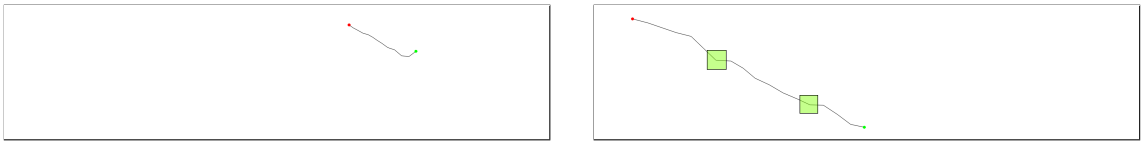
The number of steps exhibited by the pedestrian range between 7 and 11 steps, depending upon the order of the model. It can be seen that both of the pedestrians follow a similar path of progression for the accuracy rates with orders one to four having the same rate. The drop in accuracy from the third order to the fourth was measured to be -15.28% and this large drop in accuracy could be due to the matrices. Analysis of the matrices show that the third order model has had its probabilities adjusted on four rows, whereas the fourth order model had five rows adjusted for pedestrian #3,098. Pedestrian #8,512 on the other hand saw the same number of rows adjusted, respective to the order of the model. Therefore, the complexity of the path must have an implication on the accuracy of the order for the fourth order. It could be said that not enough data is present within the matrices for a fourth order to be able to make an accurate prediction. Although the fifth order is able to make a reliable (and strong) prediction with the same number of steps.

The highest accuracy gained from the model was 100% and it is expected to see this prediction method gain 100% accuracy due to it always choosing the most probable outcome and not randomly selecting a state of transition. The pedestrians that gained 100% accuracy across all orders of the stochastic model using the maximum-likelihood method. A total of 386 pedestrians gained this accuracy across all of the orders. Therefore, ten samples have been picked at random to analyse why these pedestrians gained 100% accuracy and are shown in Table 4.18.

Table 4.18: The pedestrians that gained the maximum accuracy for each stochastic model order.

Pedestrian	Accuracy (%)					Number of Steps				
	One	Two	Three	Four	Five	One	Two	Three	Four	Five
7043	100.0	100.0	100.0	100.0	100.0	23.0	22.0	21.0	20.0	19.0
2070	100.0	100.0	100.0	100.0	100.0	15.0	14.0	13.0	12.0	11.0
12555	100.0	100.0	100.0	100.0	100.0	14.0	13.0	12.0	11.0	10.0
12095	100.0	100.0	100.0	100.0	100.0	35.0	34.0	33.0	32.0	31.0
3308	100.0	100.0	100.0	100.0	100.0	12.0	11.0	10.0	9.0	8.0
6099	100.0	100.0	100.0	100.0	100.0	11.0	10.0	9.0	8.0	7.0
2217	100.0	100.0	100.0	100.0	100.0	26.0	25.0	24.0	23.0	22.0
7085	100.0	100.0	100.0	100.0	100.0	24.0	23.0	22.0	21.0	20.0
11269	100.0	100.0	100.0	100.0	100.0	34.0	33.0	32.0	31.0	30.0
10022	100.0	100.0	100.0	100.0	100.0	33.0	32.0	31.0	30.0	29.0

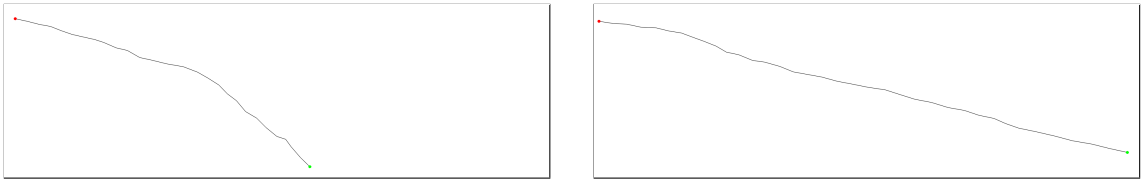
It can be seen that the selection of pedestrians that the number of steps achieved are all less than forty. To determine whether the complexity of their paths influence the accuracy of predictions. Four pedestrians have been chosen for their paths to be generated and analysed. The pedestrians chosen to have their paths analysed have been chosen based upon the number of steps that have been observed, ranging from the smallest to the largest number of steps. The first two pedestrians with the lower end of steps are shown in Figure 4.18.



(a) Pedestrian #6,099 **(b)** Pedestrian #2,070

Figure 4.18: Two of the four pedestrians chosen for analysis to determine why they gained the maximum accuracy of 100% for all orders of the dynamic stochastic model.

It can be seen from the images that the two pedestrians have a similar path whereby they are both heading in a direction towards the north-west. Pedestrian #2,070 has a slightly longer range when moving diagonally towards the north-east. This is to be expected as that pedestrian has 14 steps, opposed to the 11 steps of pedestrian #6,099. It can be seen that both paths are fairly straight, and can be seen as to why the accuracy rate recorded for these models were observed to be 100%. It can be seen with the second pedestrian there is a kink in the path of movement (highlighted in green). Although this kink is relatively different in the angle of movement, it is still considered to be the same course of direction and therefore is classed as the same direction of movement. The second batch of pedestrians are shown on Figure 4.19 and both of these pedestrians had the large collection of steps observed, ranging between 26 and 35 steps. The two pedestrians consist of a similar path with them both heading towards the north-west again from a south-east location. They both looked different due to the angle of change. As aforementioned, with how the model works they are both classified as moving towards the same direction.



(a) Pedestrian #2,217 **(b)** Pedestrian #12,905

Figure 4.19: Two of the four pedestrians chosen for analysis to determine why they gained the maximum accuracy of 100% for all orders of the dynamic stochastic model.

Due to the large number of steps and the direction of movement being the same throughout, it is easily seen as to why the maximum accuracy limit has been achieved. Therefore, it can be from these results that simple paths (such as moving in a diagonal line) can achieve a 100% prediction rate. This would be expected from the deterministic model as it is always choosing the most probable prediction and because the last few movements have been the same direction it is always going to re-select that as the next movement. Had there been a level of deviation and the method being used a random prediction then the 100% figure would have not been met.

The lowest accuracy gained for this method of prediction was achieved by pedestrian #7675 on the first order model, with an accuracy of 50%. The statistics of the pedestrians accuracy for the other orders and observed steps are shown in Table 4.19.

Table 4.19: The accuracy rates and number of steps for the pedestrian that gained the lowest accuracy in the order of five dynamic stochastic model using the maximum-likelihood prediction method.

Pedestrian	**Accuracy** (%)					**Number of Steps**				
	One	**Two**	**Three**	**Four**	**Five**	**One**	**Two**	**Three**	**Four**	**Five**
7675	50.0	59.46	80.56	94.29	94.12	38.0	37.0	36.0	35.0	34.0

It can be seen that although the pedestrian gained the lowest accuracy for the first order, a significant difference can be seen with orders two to four. Whereby the fifth order stagnates and sees only a small increase. To see why the lowest accuracy of the model was achieved on this pedestrian, the path is analysed and shown in Figure 4.20.

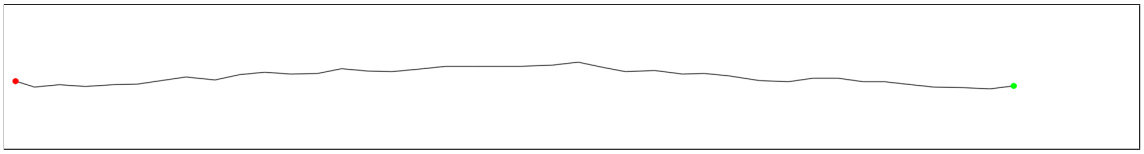


Figure 4.20: The path of pedestrian #7,675 that gained the lowest accuracy for the first order dynamic stochastic model using the maximum-likelihood prediction method.

The rather horizontal line of the pedestrian means that it would be expected a first order model will perform rather well. However, there are a few peaks and troughs within the path and these could easily be registered in the first-order model as a change in direction. To ensure that this was the case the matrix of the pedestrian was analysed for the first order model. The matrix shows that three different directions have been observed and used to update the probability matrix. The updated rows of the matrix are shown in Equation 4.4.

$$\begin{array}{c} \\ [(-1,-1)] \\ [(-1,0)] \\ [(-1,1)] \\ \cdots \end{array} \begin{array}{c} \begin{array}{ccccccccc} [-1,-1] & [-1,0] & [-1,1] & [0,-1] & [0,0] & [0,1] & [1,-1] & [1,0] & [1,1] \end{array} \\ \begin{bmatrix} 0.4375 & 0.125 & 0.4375 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0.4 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.0625 & 0.4375 & 0 & 0 & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \end{array} \tag{4.4}$$

It can be seen from the matrix that the path of movement is directly expressed within the matrix through the first three states. For example, $(-1,-1)$ is the object moving towards the north-west, $(-1,0)$ is the object moving west and $(-1,1)$ is the object moving south west. All of these movements can be seen in Figure 4.20 showing the pedestrians path. Because the probability values within the matrix are close to 50% for each direction. It is understandable why the first order model gained such a low accuracy rate.

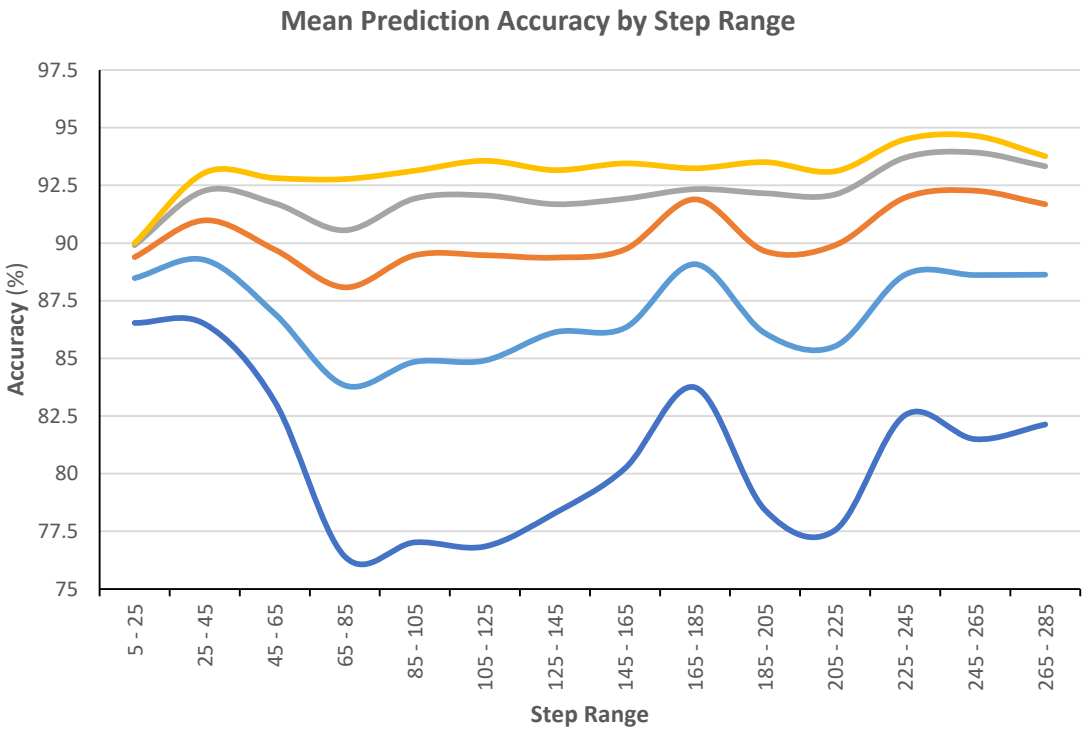
To understand why the third order gained such a high accuracy (80.56%) the matrix is also analysed, and it can be seen that the matrix had six rows that were adjusted. Therefore, this would

explain why the accuracy of the model is increased. The accuracy would be increased due to the past history of movements ensuring that the right state prediction is made, and as the path of the pedestrian is repeatable in some aspects it is understandable as to why the accuracy rate would be higher. In order to determine which of the two predictions works the best, the delta between the two methods are calculated for the mean, maximum and minimum results and is shown in Table 4.20.

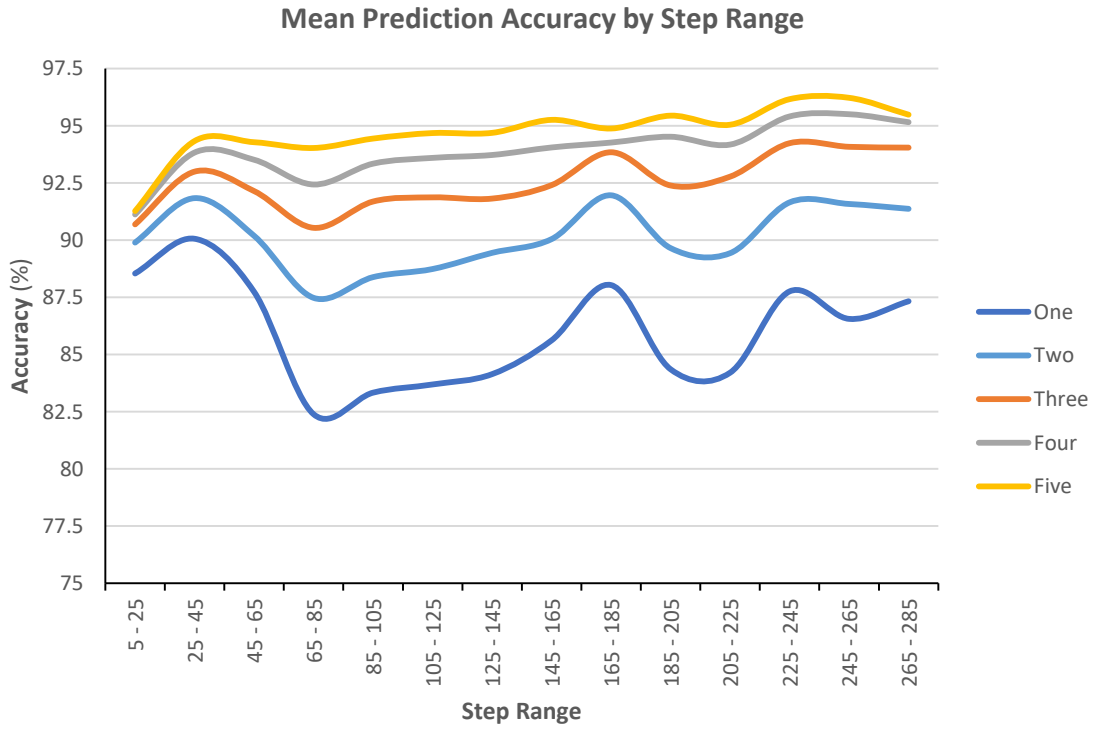
Table 4.20: The descriptive statistics of the random and maximum-likelihood methods, with the delta between the two shown.

Model Order	**Difference**		
	Minimum	**Mean**	**Maximum**
One	+8.74	+3.36	+0.44
Two	+5.7	+2.32	0
Three	−2.1	+1.82	+0.57
Four	−8.33	+1.46	0
Five	0	+1.31	+0.53

The results show a mixed collection of results with the fluctuation of positive and negative values in the column for the minimum accuracy. It can be seen that the maximum-likelihood method performs better for orders one and two. However, orders three and four were worse off with a decrease in the minimum accuracy that was obtained. However, it can be seen that overall the mean accuracy of the model is better using the maximum-likelihood method, especially when considering the first two orders whereby a +2% increase can be seen on the prediction accuracy. The graph in Figure 4.21 shows the mean accuracy for both the random and maximum-likelihood generation methods.



(a) Random Predictions



(b) Maximum-likelihood Predictions

Figure 4.21: The mean accuracy of the predictions generated using the dynamic Markov model with the two prediction methods.

The graph shows that the maximum-likelihood method is marginally better than that of the random prediction method and closely mimics the peaks and troughs of the method. However, by using the maximum-likelihood method it removes a condition which makes it a Markovian model. The key property of a Markov model is the generation of a prediction using the random number to select the next possible transition; by always selecting the highest probability from the matrix makes the model deterministic. Similar to the random prediction method analysis, a correlation test was performed upon the results of the maximum-likelihood method to determine whether the number of steps can influence the prediction accuracy of the model. The chosen correlation test was the Pearson's test, and the results are shown in Table 4.21.

Table 4.21: The statistics for the Pearson's correlation test using the IBM SPSS statistics software for the maximum-likelihood prediction method.

Order	One	Two	Three	Four	Five	
One	−0.088	0.002	0.095	0.171	0.225	**Steps**
Two	−0.088	0.002	0.095	0.171	0.225	
Three	−0.088	0.002	0.095	0.171	0.225	
Four	−0.088	0.002	0.095	0.171	0.225	
Five	−0.088	0.002	0.095	0.171	0.225	
	Accuracy					

As to be expected, the results of the correlation show that there is no correlation between the number of steps and the accuracy of the model. Therefore, it suggests that the although some of the pedestrians have a greater number of steps, due to the complexity of the path the accuracy could be affected. As there is no correlation between the number of steps and the accuracy, it is necessarily not a bad thing. The model has been experimented with data that has been collected from real pedestrians moving around a train station. If the same model is applied upon a simulation of pedestrians walking within a straight line and a fluctuation in the number of steps for the same path of movement then it is expected to possibly see a correlation between the two values.

4.2.2 Traditional Markov Model

The traditional Markov model uses the probability matrices that have been exported from the dynamic Markov model (random prediction method) to provide a level of historical data. This experiment has discarded the fifth order model due to the large redundancy the model had when loading the matrices into the model. Therefore, the analysis for this model will be undertaken on orders one to four only. The experiments performed used the random prediction method, and the results have been subjected to a statistical descriptives test. The results of the test are shown in Table 4.22.

Table 4.22: The descriptive statistics of the dynamic stochastic model using the random prediction generation method for pedestrians within $5 - 305$ steps.

Model Order	**Accuracy** (%)			**Standard Deviation**	**Delta**
	Minimum	**Mean**	**Maximum**		
One	30.25	83.72	100.00	14.16	69.75
Two	41.88	86.84	100.00	11.27	58.12
Three	44.44	88.99	100.00	9.14	55.56
Four	50.00	90.51	100.00	7.82	50.00

From the results, it can be seen that the minimum accuracy gained is far lower than what was registered when using the dynamic stochastic model (on both the random and maximum-likelihood prediction methods). The minimum accuracy sees a linear increase for an increase in the order

of the model with it ranging between 30.25% and 50%. The pedestrian that gained the lowest accuracy for the first order was #9, 842. The details of this pedestrian can be found in Table 4.23.

Table 4.23: The accuracy rates and number of steps for the pedestrian that gained the lowest accuracy in the first order of the traditional stochastic model.

Pedestrian	**Accuracy** (%)				**# of Steps**			
	One	**Two**	**Three**	**Four**	**One**	**Two**	**Three**	**Four**
9842	30.25	54.78	68.69	82.99	101.0	100.0	99.0	98.0

From the results, it can be seen that the accuracy of the model for this pedestrian jumps from 30.25% to 82.99% for a range of steps from $98 - 101$. To understand why the pedestrian gained such a low accuracy for the first order model a path of the pedestrian was formed and is shown in Figure 4.22. It can be seen from the image that the path of the image is rather chaotic with it moving around the train station and coming back upon itself and stopping within the centre.

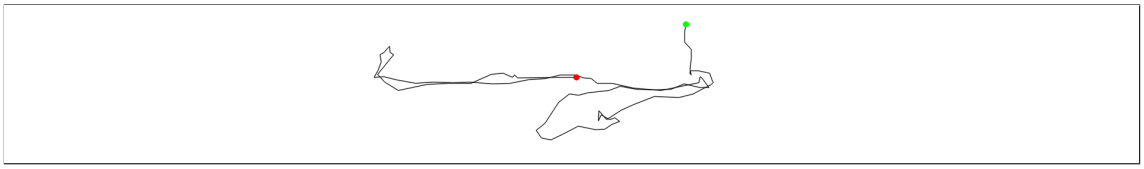


Figure 4.22: The path for pedestrian #9, 842 to analyse why such a low accuracy was gained using the traditional stochastic model for the first order.

By observing the path of the pedestrian, it can be seen why the accuracy rate for the first order is considerably low. As the probability matrix is a static vector the model is unable to learn the directional movements of the pedestrian and therefore a prediction is generated based upon the overall movements exhibited by the pedestrian based upon the dynamic stochastic model. To understand how this model fairs against the dynamic stochastic model the data in Table 4.24 displays a comparison of the accuracy gained for the first order model in each stochastic model that has been experimented with for the pedestrian.

Table 4.24: The accuracy of predictions using a first order model for the dynamic and traditional stochastic models. The pedestrian in question is #9, 842 and compares the low accuracy gained on the traditional model against the dynamic models.

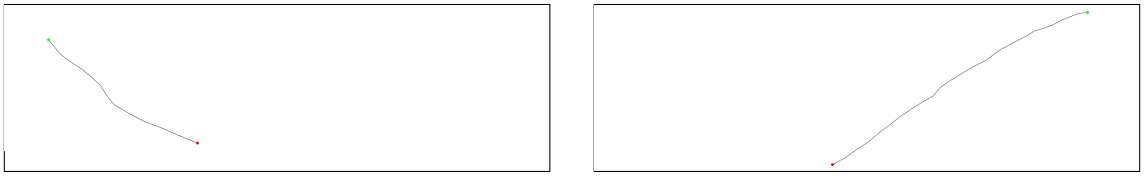
Model	**Order**	**Accuracy** (%)
Dynamic - Maximum	One	53.46
Dynamic - Random		48.62
Traditional		30.25

The dynamic stochastic model performed better with an increase in accuracy for the random and maximum-likelihood methods achieving an additional 18.37% and 23.21% in accuracy. This gain in accuracy is to be expected as the dynamic model is able to learn for each step made by the pedestrian and therefore the chance of making an incorrect prediction is lower. However, the traditional model does have some positives and it can be seen from the statistical descriptives that the maximum accuracy gained for each order was 100%. Table 4.25 shows the pedestrians that gained the maximum accuracy for orders one to four. The number of pedestrians that gained this maximum accuracy was 2, 332. A reduction is applied to the number of pedestrians for analysis with a random selection chosen.

Table 4.25: The accuracy rates and number of steps for the pedestrian that gained the lowest accuracy in the first order of the traditional stochastic model.

Pedestrian	Accuracy (%)				# of Steps			
	One	Two	Three	Four	One	Two	Three	Four
5482	100.0	100.0	100.0	100.0	26.0	25.0	24.0	23.0
7603	100.0	100.0	100.0	100.0	19.0	18.0	17.0	16.0
2051	100.0	100.0	100.0	100.0	12.0	11.0	10.0	9.0
12290	100.0	100.0	100.0	100.0	15.0	14.0	13.0	12.0
4663	100.0	100.0	100.0	100.0	10.0	9.0	8.0	7.0
7014	100.0	100.0	100.0	100.0	26.0	25.0	24.0	23.0
1218	100.0	100.0	100.0	100.0	19.0	18.0	17.0	16.0
10028	100.0	100.0	100.0	100.0	17.0	16.0	15.0	14.0
4109	100.0	100.0	100.0	100.0	20.0	19.0	18.0	17.0
11840	100.0	100.0	100.0	100.0	41.0	40.0	39.0	38.0

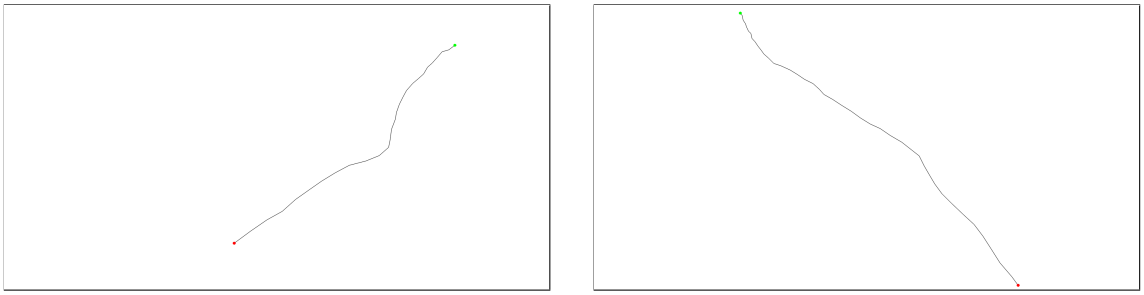
The collection of ten pedestrians show that the number of steps that were observed were relatively low and were between the range of 10 and 41 steps. To analyse as to why these pedestrians gained the maximum value of 100%. Four of them have been selected to draw their paths for analysis. The paths of the first two pedestrians with the lower number of steps are found in Figure 4.23. From the images, it can be seen that the two pedestrians have a path that is running from the south to the north in a diagonal. As mentioned beforehand, the diagonal line is a simple path and can be relatively simple to predict; therefore the 100% accuracy rate is to be expected.



(a) Pedestrian #2,051 **(b)** Pedestrian #7,014

Figure 4.23: Two of the four pedestrians chosen for analysis to determine why they gained the maximum accuracy of 100% for all orders of the traditional stochastic model.

The second two pedestrians have a higher number of steps; pedestrian #5,482 and #11,840 have 26 and 41 steps, respectively. The paths of the two pedestrians are shown in Figure 4.24. The two paths are similar to the previous collection of paths, the pedestrians are travelling from the south towards the north.



(a) Pedestrian #5,482 **(b)** Pedestrian #11,840

Figure 4.24: Two of the four pedestrians chosen for analysis to determine why they gained the maximum accuracy of 100% for all orders of the traditional stochastic model.

It can be seen that pedestrian #5, 482 has a small dent in its path, but as previously explained the direction of movement is still travelling within a north-east direction and therefore is classified as the same direction as the previous movements. To determine whether the number of steps can influence the accuracy of the predictions a correlation test is performed using the Pearson's technique. The results of the test are shown in Table 4.26 and the results show that the correlation is very weak between the two attributes.

Table 4.26: The statistics for the Pearson's correlation test using the IBM SPSS statistics software for the traditional stochastic model.

Order	One	Two	Three	Four	
One	−0.155	−0.105	−0.048	0.001	**Steps**
Two	−0.155	−0.105	−0.048	0.001	
Three	−0.155	−0.105	−0.048	0.001	
Four	−0.155	−0.105	−0.048	0.001	
	Accuracy				

The correlation is worse off in comparison to the dynamic stochastic model with the fourth order of the traditional model being negligible. The results are not surprising, as the correlation on the previous model with the two prediction methods is low. To determine whether there is any relationship within the results could potentially be obtained by categorising the types of movements together and then doing a correlation between the type of movement and the accuracy of the model. For example, do all straight paths see a linear increase when the order of the model increases? In order to do this, the matrices of the pedestrians will be scored to determine whether there is any similarity. This analysis is discussed in Section 4.2.3.

4.2.3 Pattern Analysis of the Dynamic Markov Model

Analysis of the matrices for the pedestrian dataset supplied by Yi et al. (2016*b*) uses the formulas proposed by Haralick (1979) to calculate the dissimilarity, entropy and homogeneity. The scoring function has been applied to the first-order model of the dynamic stochastic model. The different scores supplied by each calculation will be used to determine whether there is a particular selection of categories that can be created and used to organise the pedestrians based upon their movement. The scores that consisted of pedestrians over forty that were obtained for the dissimilarity, entropy and homogeneity are shown in Table 4.27.

Table 4.27: The different scores that have been obtained by using the Haralick (1979) formula's on the pedestrian matrices for the first order dynamic stochastic model.

Dissimilarity	**Count**	**Entropy**	**Count**	**Homogeneity**	**Count**
90.667	55	1.5360313058225554	679	3.2454624760507107	679
91.167	47	1.5360313058225603	551	3.1716163222045544	551
91.810	63	1.5360313058225572	980	3.171616322204556	980
97.333	1065	1.5360313058225599	386	3.2454624760507085	386
109.333	1552	1.995426246616357	41	3.8148761042878663	61
122.667	52	1.9803517391735688	60	2.9808992703110335	52
		2.123817970724675	54		

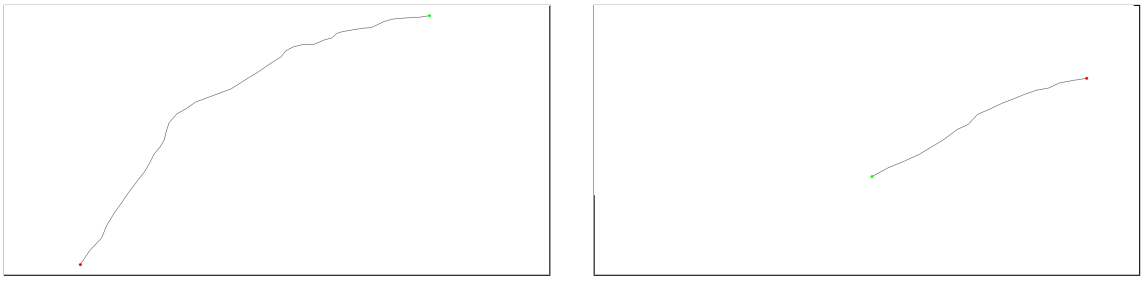
By applying a threshold to the number of pedestrians within each scoring function removes the outliers where not enough pedestrians are categorised to provide a meaningful data for analysis. In this section, each scoring function is split into a relative section and discussed in detail regarding what the scoring represents in terms of movement by the pedestrian and the type of path that has been observed. It can be seen from the table of scores that the six different dissimilarity scores have been recorded whereby the number of pedestrians with each score is over 40.

The dissimilarity scoring function is designed to determine the distance between the points within the matrix. The further the points lay away from each other then the higher the number becomes within the matrix. It can be seen from Table 4.27 that the majority of pedestrians fell within scores of 109.333 and 97.333. In order to determine the types of movement exhibited by the pedestrians with this scoring metric, the direction and path of movement are analysed. A sample of twenty pedestrians are taken for the first scoring metric, 109.333 and is shown in Table 4.28 and displays the accuracy and direction of movement that has been observed for the pedestrian.

Table 4.28: A sample of twenty pedestrians that scored 109.333 using the dissimilarity scoring function.

Pedestrian	**Accuracy** (%)	**Direction of Movement**	**Pedestrian**	**Accuracy** (%)	**Direction of Movement**
10866.0	98.19	$(-1\,1)$	242.0	95.68	$(1\ -1)$
2639.0	97.9	$(-1\,1)$	7344.0	94.12	$(1\ -1)$
8922.0	98.26	$(1\ -1)$	8065.0	96.11	$(1\ -1)$
11277.0	94.44	$(1\ -1)$	8879.0	93.75	$(1\ -1)$
2061.0	96.49	$(1\ -1)$	6301.0	95.65	$(1\ -1)$
11027.0	95.24	$(1\ -1)$	2238.0	97.04	$(1\ -1)$
3265.0	94.74	$(1\ -1)$	5204.0	97.95	$(-1\,1)$
1224.0	94.12	$(1\ -1)$	4228.0	94.44	$(1\ -1)$
7876.0	94.74	$(1\ -1)$	8143.0	93.75	$(1\ -1)$
2001.0	94.81	$(1\ -1)$	2674.0	98.06	$(-1\,1)$

From the table of pedestrians, it can be seen that the mean accuracy is 95.77% and that 80% of the pedestrians move in the direction of $(1, -1)$ which is a south-west direction. Two pedestrians have been selected from the table and their path of movement is shown in Figure 4.25.



(a) Pedestrian #10,866 **(b)** Pedestrian #8,879

Figure 4.25: The paths of pedestrian #10,866 and #8,879 that scored 109.333 using the dissimilarity score function.

It can be seen that the paths of the pedestrian generally follow the direction of the movement that has been recorded. For example, the movement $(1, -1)$ is a south-west direction as shown by pedestrian #8,879, whilst $(-1, 1)$ is a north-east direction of movement as exhibited by pedestrian #10,866. The reason these types of movements have gained a relatively high accuracy is due to the probability values that are found within the matrices of each pedestrian. It can be seen that each matrix of the pedestrians shown in the table have only their relative direction of movement updated and the probability of transitioning to the same state is 100%. The matrix of pedestrian #6,301 is shown:

$$
\begin{array}{c}
 \\
[-1,-1] \\
[-1,0] \\
[-1,1] \\
[0,-1] \\
[0,0] \\
[0,1] \\
[1,-1] \\
[1,0] \\
[1,1]
\end{array}
\begin{array}{ccccccccc}
[-1,-1] & [-1,0] & [-1,1] & [0,-1] & [0,0] & [0,1] & [1,-1] & [1,0] & [1,1] \\
0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} \\
0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} \\
0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} \\
0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} \\
0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} \\
0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} \\
0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1} & 0.\dot{1}
\end{array}
\tag{4.5}
$$

As it can be seen from the matrix, the pedestrian has a 100% probability of moving towards the same direction as previously. However, with this sort of probability it could be expected a 100% accuracy rate is gained for predictions; but this is not the case. The initial prediction is based upon a uniform distributed probability across the entire matrix; and therefore the chance of the first being correct is approximately 1.23%. This would account as to why the pedestrian gained an accuracy of 95.65%, as the number of steps observed for predictions was 23 and only 22 of these predictions were correct (if taking into account the first prediction being incorrect).

Based upon the initial prediction being a random selection from the matrix, it is expected (and seen) that with the traditional stochastic model that a 100% accuracy rate is gained. To see if a relationship exists between the number of steps and the overall accuracy of predictions a correlation test is performed upon the pedestrians and the results are shown in Table 4.29.

Table 4.29: The statistical test performed upon the pedestrians that gained a dissimilarity score of $109.33\dot{3}$.

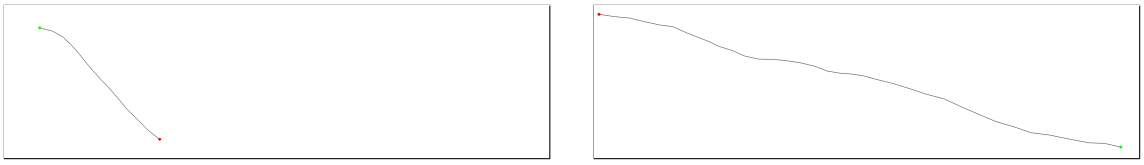
	Accuracy					
Order	One	Two	Three	Four	Five	
One	0.777	0.800	0.800	0.787	0.778	**Steps**
Two	0.777	0.800	0.800	0.787	0.778	
Three	0.777	0.800	0.800	0.787	0.778	
Four	0.777	0.800	0.800	0.787	0.778	
Five	0.777	0.800	0.800	0.787	0.778	

The relationship between the steps and accuracy is positive, however, mixed. It can be seen that the strongest relationship is with the second and third order model which was recorded as 0.800. The sample of pedestrians that gained the dissimilarity score 97.333 is shown in Table 4.30.

Table 4.30: A sample of twenty pedestrians that scored 97.333 using the dissimilarity scoring function.

Pedestrian	**Accuracy** (%)	**Direction of Movement**	**Pedestrian**	**Accuracy** (%)	**Direction of Movement**
11582	98.1	(1 1)	1973	98.1	(−1 − 1)
4552	93.52	(1 1)	8364	96.77	(1 1)
5308	90.0	(−1 − 1)	1142	95.83	(1 1)
10022	96.97	(−1 − 1)	11892	97.08	(1 1)
1483	95.42	(1 1)	4499	97.06	(1 1)
9613	95.83	(1 1)	326	98.23	(−1 − 1)
7388	90.0	(1 1)	10909	92.93	(1 1)
11018	97.83	(1 1)	6834	97.3	(−1 − 1)
7507	96.83	(−1 − 1)	3289	97.12	(−1 − 1)
5104	97.98	(−1 − 1)	11269	98.04	(−1 − 1)

It can be seen that the accuracy rates recorded for these pedestrians are above 90%. The main direction of movement exhibited by the pedestrian is $(1, 1)$ which accounts for 55% of the pedestrians in the table. The dissimilarity score of 97.333 represents movements of a pedestrian that is moving within a diagonal, either towards the south-east or north-west, represented by $(1, 1)$ and $(-1, -1)$ respectively. The mean accuracy of the pedestrians within this sample set is 96.05% and similar to the previous scoring metric the matrices show that the pedestrians have a 100% chance of transitioning to their next movement. Sample paths of two of the pedestrians from the table is shown in Figure 4.26 and it can be seen that their path of movement is a fairly straight diagonal. There is a small difference within the two lines of the pedestrians, whereby the angle of gradient is different between the two.



(a) Pedestrian $\#4,552$ **(b)** Pedestrian $\#11,269$

Figure 4.26: The paths of pedestrian $\#4,552$ and $\#11,269$ that scored 97.333 using the dissimilarity score function.

The difference of the two angles of the paths could affect the accuracy of the results had the movement of the pedestrian been calculated in a different method, rather than the pixel displacement; or had the pixel-displacement not been adjusted to a only a bipolar or binary difference. If the adjustment of the pixel displacement had not been taken into account then the results of the matrix calculations would have been different and in-turn the accuracy of the results may have been difference. The relationship between the steps and accuracy for this score are shown in Table 4.31 and a strong relationship between the two variables can be seen.

Table 4.31: The statistical test performed upon the pedestrians that gained a dissimilarity score of $97.33\dot{3}$.

	Accuracy					
Order	One	Two	Three	Four	Five	
One	0.840	0.836	0.838	0.827	0.829	**Steps**
Two	0.840	0.836	0.838	0.827	0.829	
Three	0.840	0.836	0.838	0.827	0.829	
Four	0.840	0.836	0.838	0.827	0.829	
Five	0.840	0.836	0.838	0.827	0.829	

The relationship between the two are stronger with this type of movement than it is with the previous dissimilarity score. This could be due to the smaller number of pedestrians that have been categorised into this scoring function; however the mean number of steps for the score 97.333 is 26 and 109.333 is 22. The difference in the number of steps could impact the overall accuracy of the predictions and this can be seen in the correlation results. With the relationship between the steps and accuracy being strong for the score 97.333 than it was for 109.333.

The third highest dissimilarity score suddenly drops from a collection of $1,065$ to 63 pedestrians; the score value achieved was 91.810 and the sample of pedestrians for this range are shown in Table 4.32. The mean accuracy gained for these sample of pedestrians was 83.89% and their direction of movement consisted of multiple types of paths

Table 4.32: A sample of ten pedestrians that scored 91.810 using the dissimilarity scoring function.

Pedestrian #	**Accuracy** (%)	**Direction of Movement**
9075	80.0	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
5264	81.82	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
4350	86.87	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
8579	90.28	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
6475	80.0	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
12056	83.84	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
1944	86.67	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
9227	81.82	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
5983	81.82	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
8192	87.78	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$

The collection of sample pedestrians are shown to exhibit only one type of potential paths the pedestrian may take; however, upon analysing the data individually it can be seen a second collection of paths also exist. The path shown in Table 4.32 is broken down to mean the following,

- $(-1\ -1)$ to $(-1\ -1)$: move from north-west to north-west
- $(-1\ -1)$ to $(1\ -1)$: move from north-west to north-east
- $(1\ -1)$ to $(1\ -1)$: move from north-east to north-east

This type of path is easily seen and understood when observing the path of pedestrian $\#9,075$ which is shown in Figure 4.27.

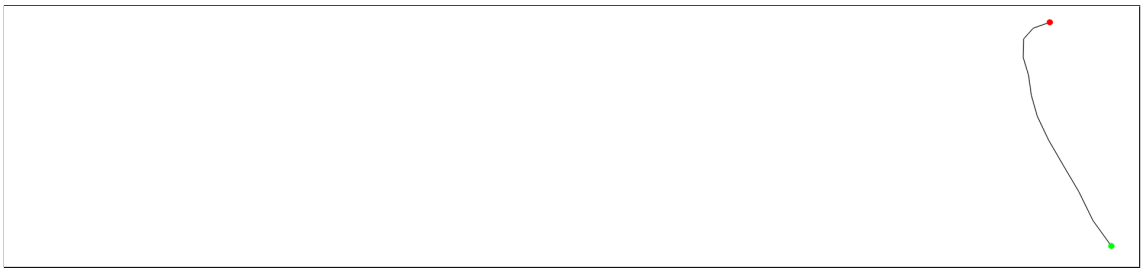


Figure 4.27: The path of pedestrian $\#9,075$ that shows the different directions the pedestrian was heading towards at points within their path.

It can be seen that the pedestrian exhibits the movements as shown in the path list. As explained in the methodology chapter, the dissimilarity score is relatively low due to a larger number of values changing within the matrix to represent the transitions between each state. This type of movement is fairly easy to predict when using the dynamic stochastic model due to a large majority of the steps that are heading towards the north-west. However, as soon as the change occurs in the direction of movement towards the north-east the model needs to re-learn the new type of continuous movement and therefore initially a number of incorrect predictions will be made; hence why the model only gained an 80% accuracy rate for this pedestrian. However, the second collection of paths are: $(-1\,1)$ to $(-1\,1)$ or $(1\,1)$ or $(1\,1)$ to $(1\,1)$, which when broken down is the following types of movement:

- (-1 1) to (-1 1): move from south-west to south-west
- (-1 1) to (1 1): move from south-west to south-east
- (1 -) to (1 1): move from south-east to south-east

This type of movement is the exact opposite to that of the previous collection of paths and an example if shown in the path of pedestrian #5,327 shown in Figure 4.28. The path of this pedestrian is similar to the one shown in Figure 4.27 but flipped so the pedestrian is walking in the opposite direction (towards the south-east).

Figure 4.28: The path of pedestrian #5,327, which is similar to the path of pedestrian #9,075 shown in Figure 4.27 but flipped.

The pedestrian within this figure gained an accuracy of 90.90% which was higher than that of the previous pedestrian due to the longer amount of time the pedestrian had spent walking in a south-west direction and then a further amount of time walking towards the south-east. The complexity of these paths are not that great and the prediction accuracy can increase if the duration of steps heading in that one direction is excessive. A correlation test has been performed upon the data to determine whether a positive relationship exists between the number of steps and the accuracy. The results are shown in Table 4.33.

Table 4.33: The statistical test performed upon the pedestrians that gained a dissimilarity score of 91.810.

	Accuracy					
Order	One	Two	Three	Four	Five	
One	0.772	0.829	0.741	0.786	0.687	**Steps**
Two	0.772	0.829	0.741	0.786	0.687	
Three	0.772	0.829	0.741	0.786	0.687	
Four	0.772	0.829	0.741	0.786	0.687	
Five	0.772	0.829	0.741	0.786	0.687	

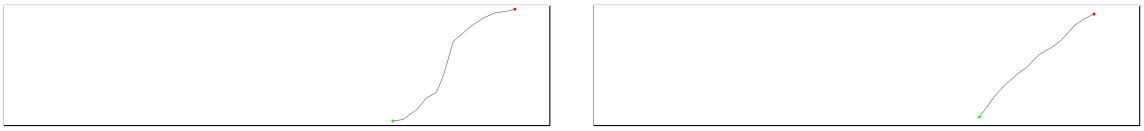
As to be expected, the relationship between the number of steps and the accuracy of the predictions for this type of movement is fairly weak due to the lower accuracy rate that was gained with the model for each pedestrian. Therefore, to determine whether any disorder have occurred within the matrices the entropy score is used. The larger the value then the elements within the matrix are considered to be the same. However, when the value is small the elements are unequal and could be considered to be chaotic. The various entropy scores extracted from the matrices whereby the number of pedestrians over 40 with the same score is shown in Table 4.27. It can be seen that the largest number of pedestrians grouped within a score of 1.5360313058225572 and a sample of five pedestrians are shown in Table 4.34.

The entropy score of 1.5360313058225572 is representative of pedestrians that are moving in a north-east direction, and solely within this direction. Although only five pedestrians are shown in the tablet, a further few more pedestrians were analysed by hand to determine whether this was the case. Upon viewing the matrices of a selected few more pedestrians it can be seen that this

Table 4.34: A sample of ten pedestrians that scored 91.810 using the dissimilarity scoring function.

Pedestrian	**Accuracy** (%)	**Direction of Movement**
4672	92.31	$(1 - 1)$
6146	90.91	$(1 - 1)$
4222	96.08	$(1 - 1)$
3652	95.45	$(1 - 1)$
4356	93.75	$(1 - 1)$

score is representative of pedestrians moving north-east. A couple of the pedestrians paths from Table 4.34 is shown in Figure 4.29 and it can be seen that the pedestrians do follow a direction heading towards the north-east.



(a) Pedestrian #4, 672 **(b)** Pedestrian #6, 146

Figure 4.29: The paths of pedestrian #4, 672 and #6, 146 that scored 1.5360313058225572 using the entropy score function.

The paths show that the pedestrians are moving towards the direction as shown in the table. However, it can be seen that pedestrian #4, 672 has a slight straight incline towards the north, but it is ever-so slightly leaning towards the east and therefore the direction of movement is classified as $(1, -1)$. To determine whether a relationship can be seen between the steps and accuracy of this score; the Pearson's correlation test has been applied to the data. The results of the test is shown in Table 4.35 and it can be seen that a correlation is seen between the two variables.

Table 4.35: The statistical test performed upon the pedestrians that gained an entropy core of 1.5360313058225572.

	Accuracy					
Order	One	Two	Three	Four	Five	
One	0.687	0.729	0.750	0.725	0.716	**Steps**
Two	0.687	0.729	0.750	0.725	0.716	
Three	0.687	0.729	0.750	0.725	0.716	
Four	0.687	0.729	0.750	0.725	0.716	
Five	0.687	0.729	0.750	0.725	0.716	

The relationship between the two variables are not strong, but it can be seen there is a relationship between the two. The strongest relationship can be seen in the third order model and can be seen in the results shown in Table 4.36 which shows the range of accuracies that were gained for each pedestrian.

Table 4.36: The accuracy of predictions for each pedestrian shown in Table 4.35. The accuracy for each order of the dynamic stochastic model is shown where the random prediction method was used.

Pedestrian	**Model Order Accuracy** (%)				
	One	**Two**	**Three**	**Four**	**Five**
4672	92.31	91.67	93.94	92.22	91.36
6146	90.91	93.33	88.89	87.5	85.71
4222	96.08	93.75	96.3	92.86	92.31
3652	95.45	96.3	95.0	94.74	94.44
4356	93.75	93.33	92.86	94.02	91.67

The table shows the accuracy of each pedestrian is different with no clear line of progression through the various orders of the model. For example, with pedestrian #3, 652 there was an observed drop in the accuracy between orders two and three, with it continuously beginning to drop. However, between the first and second order an increase was recorded. To see why a drop was recorded for this pedestrian after an order of two, the path is analysed and is shown in Figure 4.30.

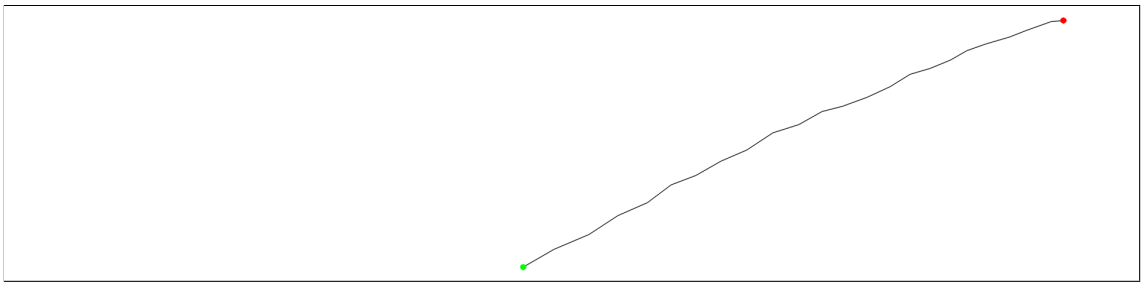


Figure 4.30: The path of pedestrian #3, 652 to determine why a drop was seen in the accuracy rates after a second order model.

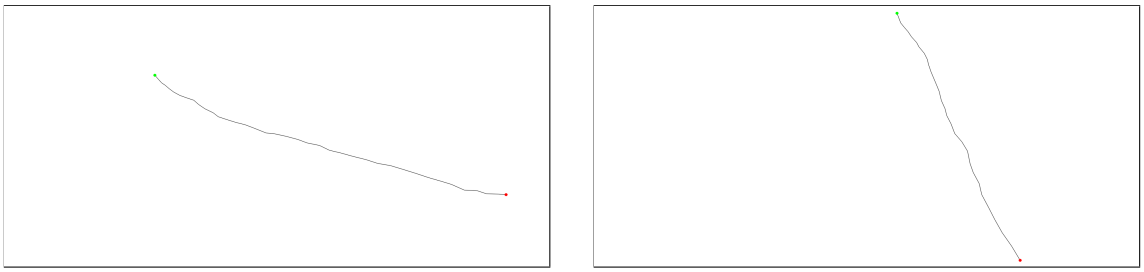
The path of the pedestrian is fairly straight and follows a direction of movement towards the north east; in-keeping with the results shown in Table 4.34. It would be expected with this type of movement that the prediction would be fairly high and follow a pattern where the first prediction is incorrect and all further predictions are correct due to the matrix having a 100% probability of transitioning to the same state. This is true for orders one, three, four and five; but the anomalous result of order two does not follow this method.

Upon analysing the individual hardware machines, it can be seen that the Beagle Bone and university server both follow the pattern of the first prediction being wrong and all subsequent predictions being correct. The laptop has an anomalous result with the second order, which drives up the accuracy for this model. The accuracy gained on the laptop was 96.82% and in order to achieve this number, then the number of correct predictions made would have to be 20.333. The number of iterations that was run on the laptop was three, and the average was taken across each of run of the experiment to gain one batch of results. This could account for the reason as to why 20.333 steps could have occurred as one run of the laptop may have obtained a 100% accuracy rate on the predictions for this order. This would explain why no progression is seen within Table 4.36. However, if the results were being analysed for the dynamic stochastic model with the maximum-likelihood prediction method then a linear progression would be seen through the results. A sample of pedestrians for the entropy score 1.5360313058225554 is shown in Table 4.37.

Table 4.37: The sample of five pedestrians that gained the entropy score of 1.5360313058225554 when analysing the first order matrices of the dynamic stochastic model with the random prediction method.

Pedestrian	**Accuracy** (%)	**Direction of Movement**
9361	97.9	(1 1)
1452	96.43	(1 1)
12192	96.77	(1 1)
7098	96.0	(1 1)
1653	97.44	(1 1)

The total number of pedestrians that fell within this score was 679 and it can be seen that their direction of movement was towards the north-east. The path of pedestrian #9, 361 and #1, 452 are shown in Figure 4.31 and it can be seen that each pedestrian follows a path heading towards the south-east although the angle of path is different.



(a) Pedestrian #9, 361 **(b)** Pedestrian #1, 452

Figure 4.31: The paths of pedestrian #9, 361 and #1, 452 that scored 1.5360313058225554 using the entropy score function.

The accuracy for this model for a first order is fairly high and is in-keeping with a pattern that is observed when only one direction of movement is present in the matrix. The pattern follows the convention that the first prediction is always wrong; and subsequent predictions are always correct due to the high probability of transition to the same state of direction. This observation has been made across the previous entropy score and the paths exhibited by the pedestrian and some of the dissimilarity scores whereby a single direction of movement has been observed. The sample of pedestrians with a score of 1.5360313058225603 is shown in Table 4.38 and it can be seen that their direction of movement is $(-1, 1)$ which is a south-west direction.

Table 4.38: The sample of five pedestrians that gained the entropy score of 1.5360313058225603 when analysing the first order matrices of the dynamic stochastic model with the random prediction method.

Pedestrian	**Accuracy** (%)	**Direction of Movement**
9816	97.44	(−1 1)
2598	95.91	(−1 1)
11618	97.01	(−1 1)
7501	96.62	(−1 1)
5097	97.56	(−1 1)

It can be seen from the various scores achieved with this formula that they all show a single direction of movement within the matrices. For example, the score 1.5360313058225603 depicts a pedestrians direction of movement is heading towards the south-west, whilst 1.5360313058225554

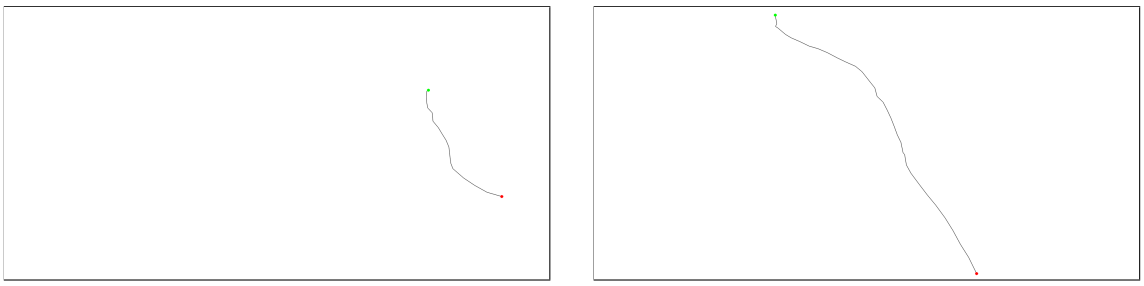
and 1.5360313058225572 show pedestrians moving towards the south-east and north-east, respectively. Similarly to the dissimilarity scores, there was far more scores that were calculated for the entropy; however, a selection of scores were chosen for analysis and it can be seen that each score can describe a direction of movement exhibited by the pedestrian.

The homogeneity scores are calculated to determine whether the elements of a matrix are situated near the diagonal of the matrix. For a first-order model, it can be seen that any number that sits on the diagonal of a matrix is a transition to the same state, i.e. a stationary to stationary movement is recorded. Therefore, if the scoring of this function is low it is expected to see a number is laying on the diagonal of the matrix which can represent a stationary pedestrian, or pedestrians walking within a straight line. The selection of homogeneity scores chosen for analysis are shown in Table 4.27. It can be seen the largest number of pedestrians fell within the score of 3.171616322204556, whereby a total of 980 pedestrians were allocated this score. It can be seen that this score has a similarity with the entropy score of 1.5360313058225572 whereby the direction of movement was recorded as $(1, -1)$. With the same number of pedestrians categorised with this score. Analysis of the score 2.9808992703110335 shows that only 52 pedestrians were allocated with this score, and a sample of the pedestrians are shown in Table 4.39.

Table 4.39: The sample of ten pedestrians that gained the homogeneity score of 2.9808992703110335 when analysing the first order matrices of the dynamic stochastic model with the random prediction method.

Pedestrian	**Accuracy** (%)	**Direction of Movement**
7712	93.75	$[(-1\,1)to(1\,1)]or[(1\,1)to(1\,1)]$
1158	97.22	$[(-1\,1)to(1\,1)]or[(1\,1)to(1\,1)]$
7266	93.52	$[(-1\,1)to(1\,1)]or[(1\,1)to(1\,1)]$
4450	96.3	$[(-1\,1)to(1\,1)]or[(1\,1)to(1\,1)]$
11641	93.94	$[(-1\,1)to(1\,1)]or[(1\,1)to(1\,1)]$
7904	96.08	$[(-1\,1)to(1\,1)]or[(1\,1)to(1\,1)]$
7027	97.22	$[(-1\,1)to(1\,1)]or[(1\,1)to(1\,1)]$
6775	97.57	$[(-1\,1)to(1\,1)]or[(1\,1)to(1\,1)]$
2391	98.04	$[(-1\,1)to(1\,1)]or[(1\,1)to(1\,1)]$
7020	96.97	$[(-1\,1)to(1\,1)]or[(1\,1)to(1\,1)]$

It can be seen that the path of this score is different to the previous scores that have been analysed for the dissimilarity and entropy. The path of movement can either be the pedestrian moving from south-west to south-east, to contentiously moving towards the south-east. It can be seen that the path shows that there is no transition back to the south-west state. Therefore, the pedestrian may have started moving towards the south-west before changing their mind and heading south-east. To determine whether this was the case, the paths of pedestrian #7, 712 and #2, 391 are shown in Figure 4.32.



(a) Pedestrian #7, 712 **(b)** Pedestrian #2, 391

Figure 4.32: The paths of pedestrian #7, 712 and #2, 391 that scored 2.9808992703110335 using the homogeneity score function.

It can be seen from the paths of the pedestrians that the direction of movement towards the south-west is relatively small; and may have only occurred once within the frequency matrix which is used to calculate the probabilities. Which is why there is no transition back into this direction of movement once the user begins to head towards the south-east direction. To see if this was the case, the frequency matrix of each pedestrian in the table is analysed, and an example of the matrix for pedestrian #2, 391 is shown,

$$\begin{array}{c} \\ [-1,-1] \\ [-1,0] \\ [-1,1] \\ [0,-1] \\ [0,0] \\ [0,1] \\ [1,-1] \\ [1,0] \\ [1,1] \end{array} \begin{array}{c} \begin{array}{ccccccccc} [-1,-1] & [-1,0] & [-1,1] & [0,-1] & [0,0] & [0,1] & [1,-1] & [1,0] & [1,1] \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 32 \end{bmatrix} \end{array} \tag{4.6}$$

From the matrix, it can be seen that the pedestrian was observed for one moment where they were heading in the south-west direction. A majority of the steps were taken when heading towards the south-east and this was seen in all the matrices for the pedestrians in Table 4.39. The third score for analysis is the homogeneity score 3.8148761042878663. whereby the pattern of movement exhibited is similar to the dissimilarity score of 91.810. However, it can be seen that the dissimilarity score had two more pedestrians within in when compared to the homogeneity category.

Table 4.40: The sample of ten pedestrians that gained the homogeneity score of 3.8148761042878663 when analysing the first order matrices of the dynamic stochastic model with the random prediction method.

Pedestrian	**Accuracy** (%)	**Direction of Movement**
1275	88.89	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
9274	84.62	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
4350	86.87	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
2582	81.82	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
10035	95.0	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
1003	81.82	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
2380	85.19	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
8192	87.78	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
8579	90.28	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$
12056	83.84	$[[(-1\ -1)to[(-1\ -1)or(1\ -1)]]or[(1\ -1)to(1\ -1)]]$

The two pedestrians that are missing from the collection of results are pedestrians #1, 225 and #5, 327. It can be seen by analysing their matrices that the path of movement is different to that expressed in Table 4.32. The path of movement for these two pedestrians are: $(-1, 1)$ to $(-1, 1)$ or $(1, 1)$, or $(1, 1)$ to $(1, 1)$. Therefore, the path is different to that which has been recorded for the homogeneity and the samples of pedestrians that were originally shown for the dissimilarity score. This is understandable, as it has been seen through the analysis of these scoring functions that the dissimilarity score shows the path of movements that are in the same direction, but also the flipped direction (so two types of movements that are essentially the same). Whereas, the entropy and homogeneity are functions that only select single paths of movement (and does not consider the flipped version).

4.2.4 Processing Times of the Markov Models

The accuracy of the predictions obtained with the stochastic models are fairly high and within the expectations. However, the accuracy of the predictions is not the only important aspect of this study; the time taken to make these predictions is also of significant importance. Three pieces of hardware have been used with contrasting specification between them to gain an understanding on the amount of computational resources required by the stochastic model. The recorded times from the experiments for the stochastic models are shown in Table 4.41. The results of the dynamic Markov model with the maximum-likelihood prediction method are denoted by the column with the heading 'Dynamic^1', whereas the model with the random method are denoted by the 'Dynamic^2' heading.

Table 4.41: The processing time of each hardware for the respective stochastic model. 'Dynamic^1' are the results of the dynamic Markov model with the maximum-likelihood predictions and 'Dynamic^2' are the times of the random prediction method.

		Hardware Processing Time (s)		
Type of Model	**Model Order**	**Embedded**	**Laptop**	**Server**
Dynamic^1	1	0.14	0.049	0.006
	2	0.153	0.07	0.008
	3	0.174	0.084	0.009
	4	0.197	0.096	0.01
	5	0.217	0.097	0.011
Dynamic^2	1	0.127	0.047	0.012
	2	0.154	0.068	0.015
	3	0.183	0.078	0.018
	4	0.195	0.087	0.02
	5	0.218	0.093	0.015
Traditional	1	0.028	0.003	0.000511
	2	0.02	0.007	0.000618
	3	0.038	0.008	0.000705
	4	0.031	0.007	0.000809
	5	-	-	-

From the collection of results it can be seen that the server gained the best results for the amount of processing time that was taken. The results of the experiment are less than half-a-second and these results are to be expected due to the hardware that is found within the server. It can be seen from the table that the traditional model experiment was only executed up to a fourth order model and this was due to limitations that were imposed from the embedded hardware. To ensure that a fair comparison is made between the three pieces of hardware for this model, the results of the fifth order have been omitted from the analysis for the server and laptop. However, it is noted that the hardware found within the server and laptop are capable of computing a fourth and fifth order model.

The results of the dynamic Markov model (using the random prediction method) on the university server show that the model is able to compute within 0.02 seconds. A linear growth is seen for an increase in the order of the model, with the difference between a second and first order model being observed to be 0.003 seconds. There seems to be a pattern with the increase across the first three orders whereby the increase between them is 0.003 seconds until the fourth order is reached and this decreases slightly to 0.002 seconds. Interestingly, it can be seen that with the fifth order model a decrease is observed when compared to the fourth order model. The amount of time taken decreases from 0.020 to 0.015 seconds. When comparing the random prediction method with the maximum-likelihood it can be seen that the times recorded with the latter are better, and this could be due to the methodology behind selecting a prediction. The random prediction method involves the generating of a number between zero and one, and then comparing this number against each probability value within the selected row of the matrix. This process can add time to the

prediction process. Unlike the maximum-likelihood method whereby a probability is chosen based upon the maximum value within the selected row. Therefore, the process is able to save some time on the comparison of two values for each index in the matrices row. The graph in Figure 4.33 shows the times plotted for each dynamic and traditional stochastic model.

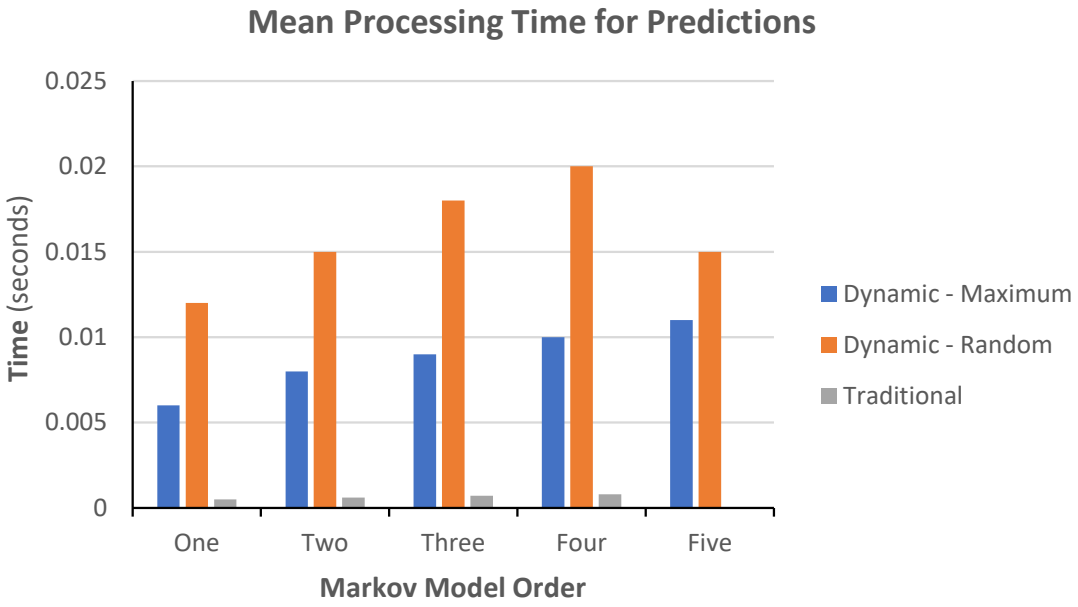


Figure 4.33: The mean processing time for each of the various models and prediction methods used on the university server.

The graph shows the difference that can be seen in the recorded times for each type of model that is used on the server, with the various prediction methods of the dynamic model. The fastest model for computation was the traditional stochastic model, and it can be seen that the growth in the time for each order of the model is fairly flat. The other two methods on the hand see a linear increase in the computation time, with the random prediction method growing faster when compared the maximum-likelihood method. The drop in time can be observed on the fifth order with the random prediction method; the reasoning behind this drop is unknown. The results of the laptop on the other hand faired well in comparison to the server, although it can be seen that the computation time was slower with the laptop. The recorded times of the laptop are all below a tenth of a second; and are therefore still well within the realms of being able to generate a prediction within a real-time constraint. Out of the three models, the traditional mode was observed to have the best time recorded; with the times coming in under 0.009 seconds. The lower time recorded for the traditional model is to be expected due to the nature of how the model is built. The model has a fixed probability matrix and therefore is not required to be adjusted each time a movement is made by the pedestrian. This saves having to search through the matrix (which can be time consuming for a larger model) and adjusting the probability values. Due to the lack of needing to update the probabilities it would explain as to why the low values were recorded. The graph in Figure 4.34 shows the times of the traditional stochastic model for each piece of hardware tested.

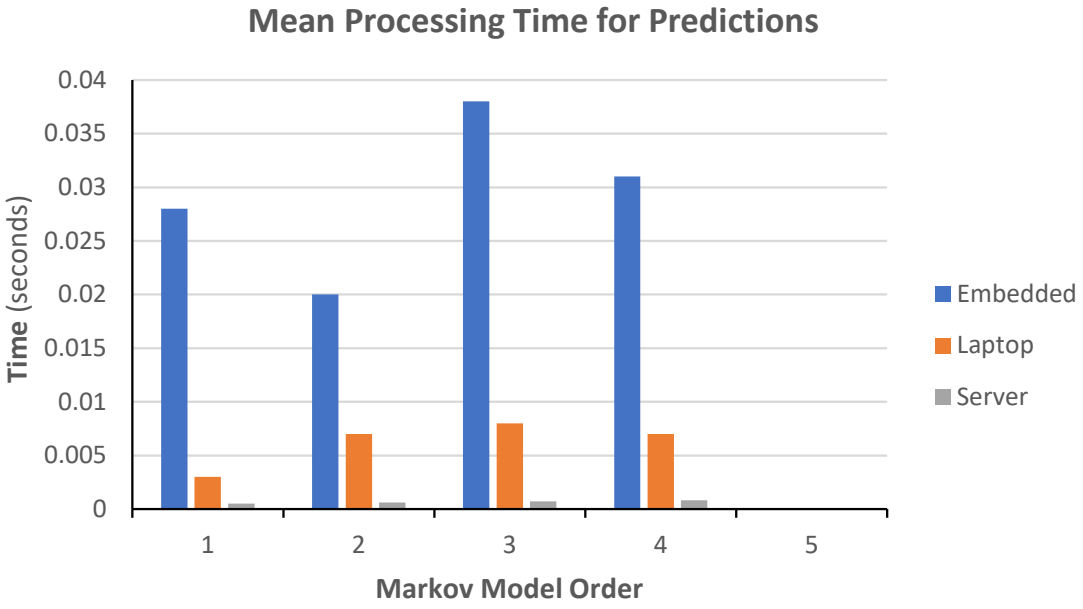


Figure 4.34: The mean processing time for the traditional Markov model on each of the devices used for the experiments.

It can be seen in the graph that the results of the embedded network are mixed, with a decrease in the times between the first and second order traditional model. The computation time then suddenly increases for a third order and then subsequently decreases once again for the fifth order. The mean time could be affected by the pedestrians and the hardware being unable to process pedestrians with a high step-count within a timely manner. To determine whether this was the case, two pedestrians have been chosen at random from the embedded data and their times are analysed against the number of steps exhibited. The data is shown in Table 4.42 and a selection of random pedestrians have been chosen for analysis.

Table 4.42: Two pedestrians from the collection of results for the embedded hardware and the traditional stochastic model to determine why a difference in the times for each order was recorded.

Pedestrian	**Time** (s)				**Number of Steps**			
	One	**Two**	**Three**	**Four**	**One**	**Two**	**Three**	**Four**
2485	0.01	0.01	0.03	0.01	17.0	16.0	15.0	14.0
8182	0.01	0.02	0.06	0.02	30.0	29.0	28.0	27.0

From the table it can be seen that the range of steps are relatively low for the two pedestrians and there does not seem to be a relationship between the number of steps and the total time taken for processing. For example, pedestrian $\#8,182$ was observed for 30 steps and the results for an order of one model and the time for processing was 0.01 seconds. However, pedestrian $\#2,485$ took almost half the amount of steps but was processed within the same amount of time. However, a similar pattern that is observed in the graph is present within the table. For example, pedestrian $\#8,182$ saw an increase in the computation time between orders one and three, then a decrease on the fourth order. There was a similar pattern that can be seen in pedestrian $\#2,485$. The pedestrian saw a stagnation on the processing time for orders one and two, with an increase for the third order before decreasing again on the fourth.

4.2.5 Neural Networks

The Neural Network experiments have been implemented to provide a comparative study against the performance of the dynamic stochastic model and the most popular method of prediction, neural networks. Two differing methods of a neural network have been implemented, a simple feed-forward network and a more complex recurrent neural network. The results of the neural networks have been obtained by using the server located at Coventry University due to the large amount of resources required to run the experiment. The experiment was attempted to be performed upon the Dell XPS laptop, but unfortunately due to hardware limitations the networks were unable to compute. Due to this, the experiments were not attempted to be performed upon the Beagle Bone Black. The results for the feed-forward and recurrent neural networks are discussed in their respective sections; and due to the limit of hardware resources, the models are constrained to a third order model whereby just three past movements are used to generate a prediction.

The feed-forward neural network is a simple network, whereby the data is pushed forward through the nodes of the network and the weights are adjusted accordingly. The outcome from the network is a prediction of the pedestrians intended next direction. The results of the network have been subjected to a statistical descriptives analysis for the accuracy of the neural network and is shown in Table 4.43.

Table 4.43: The descriptive statistical analysis of the feed-forward neural network for the accuracy of predictions made.

Model Order	**Accuracy** (%)			**Standard Deviation**	**Delta**
	Minimum	**Mean**	**Maximum**		
One	0.00	78.51	100.00	19.25	100.00
Two	0.00	69.43	100.00	24.45	100.00
Three	0.00	77.09	100.00	20.28	100.00

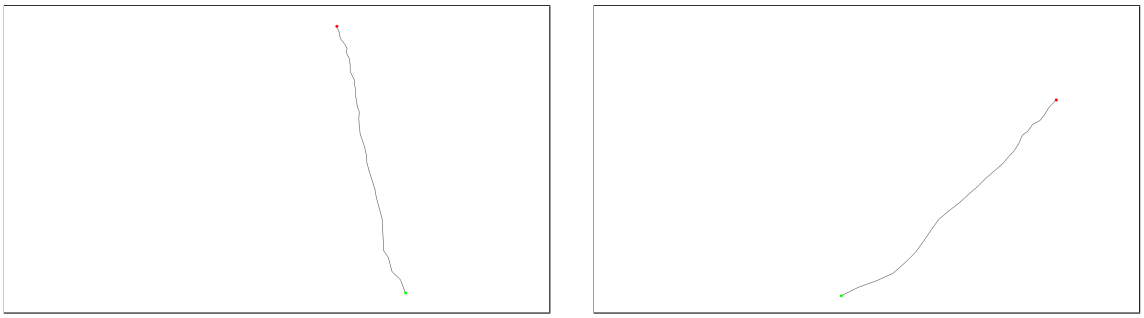
It can be seen from the table that the minimum accuracy gained was zero and due to this the delta (difference between the maximum and minimum) will be recorded as 100% and therefore, the minimum and delta are retracted from the analysis of the results. It can be seen from the descriptives that the results of the neural network are not in-line with the expectation. It was expected to see that a linear growth would occur with the accuracy rates for an increase in the number of past movements that are fed into the network. However, the mean accuracy of the results have no linearity and are a mixed collection of results with the first order gaining a mean accuracy of 78.51%, the second order 69.43% and the third 77.09%. To understand why these results are not as expected; a selection of pedestrians are chosen for analysis and are shown in Table 4.44.

Table 4.44: The accuracy rate and number of steps observed by the feed-forward neural network for generating predictions using the pedestrian dataset.

Pedestrian	**Accuracy** (%)			**# of Steps**		
	One	**Two**	**Three**	**One**	**Two**	**Three**
1896	67.57	80.56	71.43	37.0	36.0	35.0
2497	100.0	69.57	100.0	24.0	23.0	22.0
3774	40.74	88.46	64.0	27.0	26.0	25.0
5390	36.67	37.93	14.29	30.0	29.0	28.0

The results show that the accuracy rates for each order are fluctuating and that there is no linear growth in the accuracy for when the order of the network is increased. This sort of pattern is to be expected as the feed-forward neural networks do not necessarily need to have a collection of past movement introduced to affect the accuracy of the predictions. Instead, the network is trained over a length of epochs to ensure that the input can match the target variable as close as possible. To determine why the neural network behaved in this particular fashion, the paths of

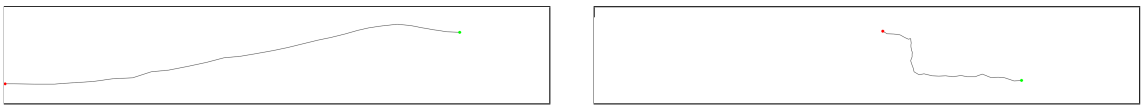
each pedestrian are examined. The first two pedestrian (#1, 896 and #2, 497) paths are shown in Figure 4.35 and it can be seen that the two paths are similar, with the pedestrian taking a path that is going towards the north. However, there is a slight difference in the path with them both going off in a different direction, one west and one east. The two paths are classified as moving within a diagonal, and therefore the movement for each pedestrian would be $(-1, -1)$ and $(1, -1)$, respectively.



(a) Pedestrian #1, 896 **(b)** Pedestrian #2, 497

Figure 4.35: Two of the four pedestrians chosen for analysis to determine why the difference in accuracy rates were achieved for each order of the feed-forward neural network.

By analysing the paths of the pedestrian, it cannot be seen as to why the feed-forward network would gain different rates of accuracy for each order and it could be attributed towards the training phase of the network. The training phase of the network is undertaken for one epoch which would hinder the impact of getting the correct target value. Therefore, it could be the fact that there was not enough 'training' allocated to ensure that the accuracy could follow a linear progression. The paths of pedestrians #3, 774 and #5, 390 is shown in Figure 4.36.



(a) Pedestrian #3, 774 **(b)** Pedestrian #5, 390

Figure 4.36: Two of the four pedestrians chosen for analysis to determine why the difference in accuracy rates were achieved for each order of the feed-forward neural network. From right to left, pedestrian #3774 and #5390.

It can be seen that the two paths for these pedestrians are rather different, with the path of pedestrian #3, 774 following an albeit straight line, and pedestrian #5, 390 is more of a 's-shape' path. The accuracy of pedestrian #3, 774 is relatively low with the order of one model seeing an accuracy rate of 40.74%, that is approximately a reduction of 27% when compared to pedestrian #1, 806 who exhibited a similar sort of path. The only difference between the two paths is that ones vertical and the other horizontal. It has been mentioned that the difference between these two pedestrians could be due to the amount of training that has been applied and also the number of steps that have been observed. The number of steps exhibited by pedestrian #1, 806 is between 35 and 37, depending on the order of the model. Whereas, pedestrian #3, 774 saw fewer steps observed with the range of steps being 25 to 27. The second pedestrian (#5, 390) faired the worse in terms of accuracy, with a low value of 14.29% being observed for the third order model. The low accuracy gained on this order could be due to the training mechanism and the pattern of movement not fitting well with the chosen activation function of the hidden neurons. It could also be due to the low number of training epochs that were present within the training phase and even the type of activation function that was used for the hidden neurons.

Based upon the accuracy achieved for each of the models, it can be seen that the feed-forward network is not best suited for dealing with predictions where the pedestrians movement could be

considered to be complex. There were also numerous instances where an accuracy rate of zero was achieved, and this was most likely due to not enough training epochs being allocated to ensure that a robust prediction is made. However, the single training epoch was allocated to ensure that the network closely resembles the process that a Markovian model follows. By only training the network by one epoch it ensures that the prediction is generated within the first instance and not based upon a method of further learning. The stochastic model is able to generate predictions on the fly based upon a single recalculation of the probabilities and therefore, a single epoch of training will recalculate the weightings of the neural network once and then a prediction is made. However, it can be seen that by following this method is not the most accurate method, and that feed-forward neural networks require a certain level of training before it could match the prediction aspect of the stochastic model.

Table 4.45: The times taken for each pedestrian when using the feed-forward neural network to make a prediction.

Pedestrian	**Times** (s)		
	One	**Two**	**Three**
1896	33.75	137.48	318.56
2497	15.79	81.21	276.3
3774	28.96	115.65	291.46
5390	11.5	73.81	212.15

Overall, it can be seen that by using a feed-forward network the accuracy rates are not what were expected; especially with the large amount of computation time that was taken. The time undertaken for generating the predictions are shown in Table 4.45 and it can be seen that the amount of time taken for each neural network is not the most efficient method for generating a prediction. Pedestrian #1,896 took almost 34 seconds to compute, that is almost a second per step that was taken and was achieved on a first order model. The recorded timings only got worse, with the order of two network taking 137.5 seconds and the third order network taking almost 320 seconds. The graph in Figure 4.37 displays the mean time for each order across every pedestrian within the dataset.

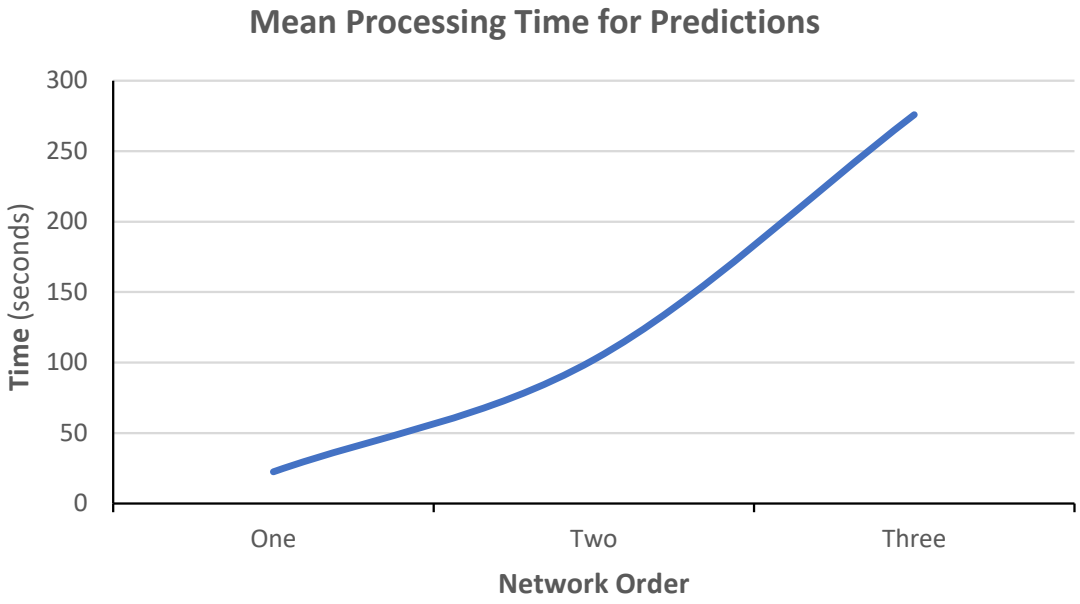


Figure 4.37: The mean time for each order of the feed-forward network for all pedestrians within the dataset.

As expected, there is a linear increase when the order of the network progresses. The growth in time can be considered to be exponential due to the steep increase in the line; with the difference between the first order and second order being 233 seconds, an increase of $1,344.19\%$. Based upon the analysis of the feed-forward network, it is expected that the recurrent neural network should perform better due to the nature of the output variable being fed back into the network. This could provide further machine learning to the model to ensure that the correct prediction is made on the next instance. The recurrent neural network model has been developed to provide a comparison between the dynamic stochastic model and the feed-forward neural network. It has been seen from the analysis of the feed-forward network that is particularly weak and in some cases is unable to provide a prediction at all. To determine how well the recurrent network performs against the other models and networks a descriptive analysis has been performed using the IBM SPSS software and is shown in Table 4.46.

Table 4.46: The descriptive statistical analysis of the feed-forward neural network for the accuracy of predictions made.

Model Order	**Accuracy** (%)			**Standard Deviation**	**Delta**
	Minimum	**Mean**	**Maximum**		
One	0.00	67.06	98.08	22.84	98.08
Two	0.00	72.82	100.00	19.44	100.00
Three	0.00	76.27	100.00	18.13	100.00

Similar to the feed-forward network the minimum accuracy was measured to be zero, this could be due to the level of training the network went under. For this network, it was decided to use fifteen epochs as the initial testing showed that using just one epoch garnered too many incorrect predictions and the accuracy rate was recorded as zero. However, unlike the feed-forward neural network, a linear increase can be seen across the mean accuracy of the network for an increase in the order. This is a positive sign that the recurrent neural network is an appropriate model for predicting a pedestrians next movement based upon the history of past movements. To analyse the performance of the recurrent neural network, four pedestrians have been chosen at random from the list and are shown in Table 4.47.

Table 4.47: The accuracy rate and number of steps observed by the recurrent neural network for generating predictions using the pedestrian dataset.

Pedestrian	**Accuracy** (%)			**# of Steps**		
	One	**Two**	**Three**	**One**	**Two**	**Three**
567	0.0	25.0	57.14	9.0	8.0	7.0
3815	58.82	63.64	50.0	34.0	33.0	32.0
4461	0.0	12.5	28.57	9.0	8.0	7.0
8580	50.0	57.14	33.33	8.0	7.0	6.0

Pedestrian #567 saw an accuracy rate of zero achieved for an order of one model. However, the pedestrian saw a linear increase across orders two and three. By observing the amount of steps that have been exhibited by the pedestrian it can be seen that the accuracy is not affected by the number of steps. This is because the higher accuracy rates were gained on fewer steps than previously. To see whether this was the case, and whether a relationship can be seen between the steps and accuracy a correlation test is performed upon the results of the dataset using the Pearson's method. The results of the correlation test can be seen in Table 4.48.

Table 4.48: The correlation test for the recurrent neural network to find a relationship between the steps and accuracy.

	Accuracy			
Order	One	Two	Three	
One	0.246	0.203	0.151	**Steps**
Two	0.246	0.203	0.151	
Three	0.246	0.203	0.151	

From the correlation test it can be seen that there is a weak relationship between the number of steps and the accuracy. These results are to be expected due to the inconsistency of some of the results that are shown in Table 4.47. For example, pedestrian #3, 815 saw an increase in the accuracy between the first and second order, and then it decreased again for the third order. To determine why the pedestrian saw a decrease in the accuracy rate for the third order model the path is displayed in Figure 4.38 and is analysed.

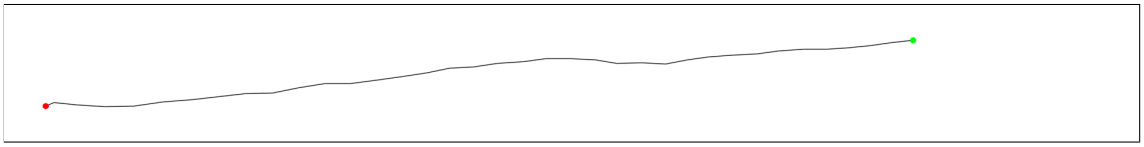
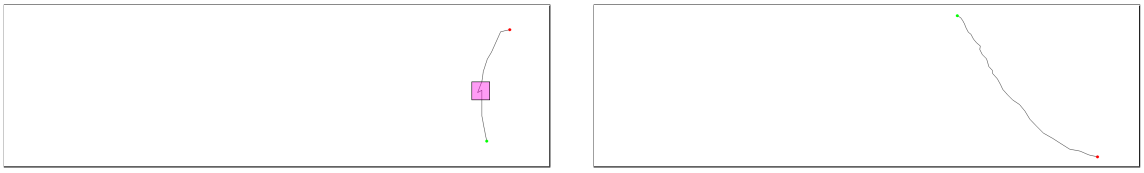


Figure 4.38: The path of pedestrian #3815 to determine as to why a lower accuracy was gained on the third order; when an increase could be seen on the first and second orders.

It can be seen that the path of the pedestrian is travelling within a fairly straight line, horizontally from east to west. The path has a linear progression towards the north and therefore the movement would be described as $[1, -1]$ within the network. To make a prediction, the network is trained upon the data of previous movement that is inserted into the model and the next move is used as a target in-order to train the weights of the network before a prediction is made. The reason for the drop in the accuracy of the third order model is unexplained due to the nature of the movement exhibited by the pedestrian. Therefore, it is assumed that the lack of training on the recurrent neural network inadvertently affected the accuracy of the overall predictions. To see whether this was a anomalous result, the paths of pedestrian #4, 461 and #8, 580 are analysed; and are shown in Figure 4.39.



(a) Pedestrian #4, 461 **(b)** Pedestrian #8, 850

Figure 4.39: Two of the four pedestrians chosen for analysis to determine why the difference in accuracy rates were achieved for each order of the recurrent neural network.

It can be seen from the images that the two paths are rather different in terms of the direction the pedestrian is heading and the length of the path. Pedestrian #4, 461 can be seen to have travelled from the east, towards the south-west and make a small change in direction (highlighted in pink) before resuming their journey towards the south. The type of movement that is highlighted in pink can be fairly difficult to predict, especially if the movement is only made for a couple of steps. This 'change-of-mind' in their direction of movement could contribute towards the low accuracy rates that were achieved with this pedestrian. It can be seen that on the first order model the accuracy of predictions was zero, whilst the second and third orders performed better (12.50% and 28.57%, respectively). The second pedestrian (#8, 850) faired better in the accuracy rates, which saw an increase between the first and second orders (50% and 57.14%). However, it

can be seen that on the third model the accuracy rate dropped to 33.33%. A pattern can be seen within these results, and that for a lower number of steps there is a possibility that the accuracy rate of a first order model could be zero and then subsequent low prediction accuracies obtained on the other orders. The times gained by these pedestrians are shown in Table 4.49 and it can be seen that the times are far worse in comparison to the feed-forward neural network (shown in Table 4.45).

Table 4.49: The times taken for each pedestrian when using the feed-forward neural network to make a prediction.

Pedestrian	**Times** (s)		
	One	**Two**	**Three**
567	84.91	284.29	701.13
3815	124.74	449.83	1144.1
4461	72.77	258.28	600.93
8580	71.34	333.98	711.0

The times of the feed-forward network and the recurrent network are displayed in Figure 4.40 and when comparing the times of the two networks together it can be seen that there is a significant difference between the feed-forward network times and the recurrent network.

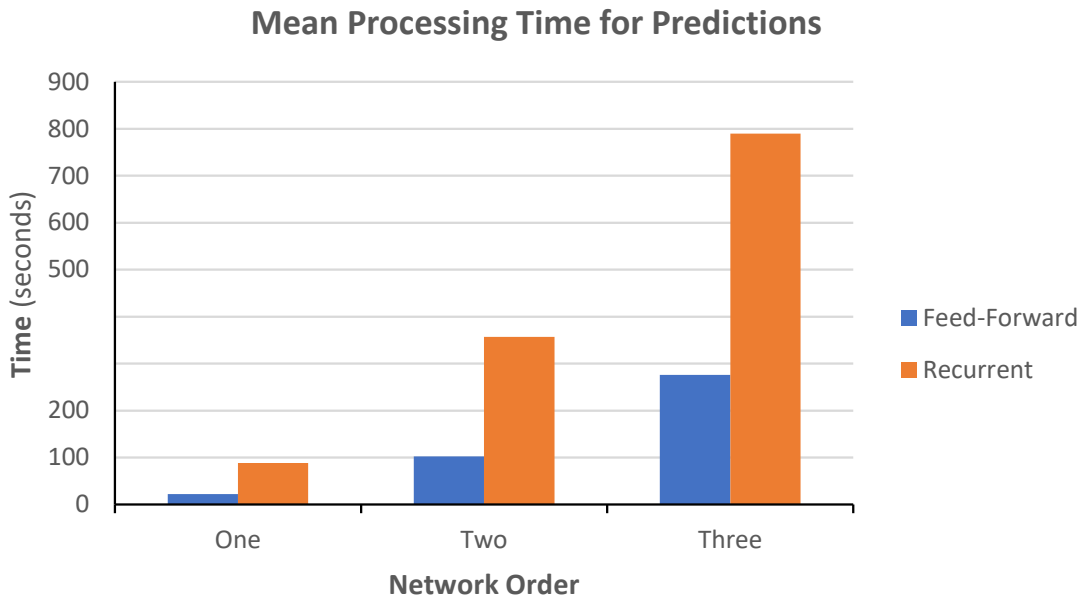
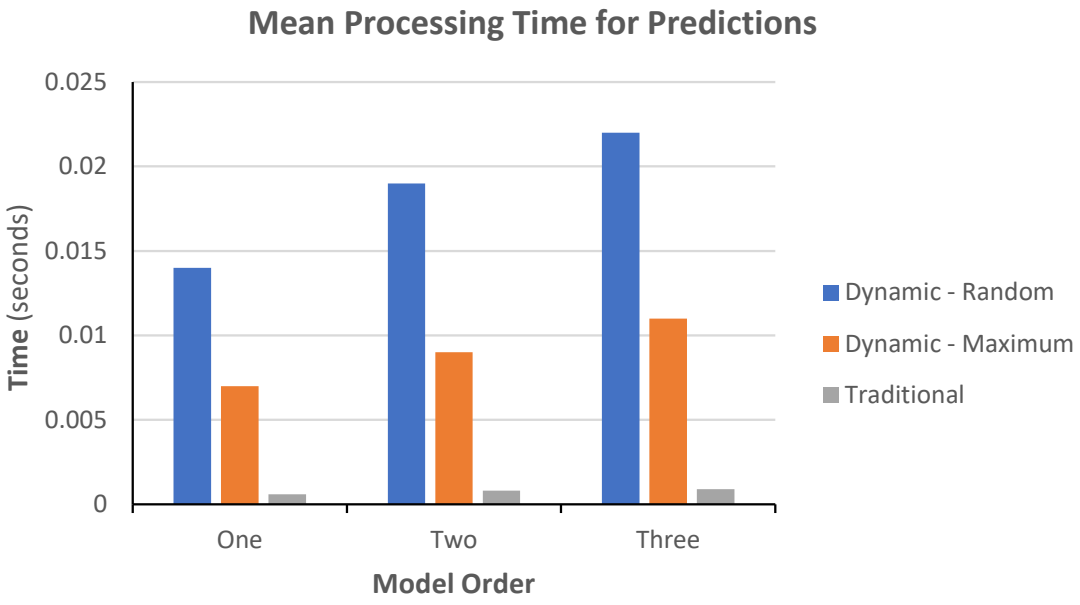


Figure 4.40: The mean accuracy time for the feed forward and recurrent networks compared against each other.

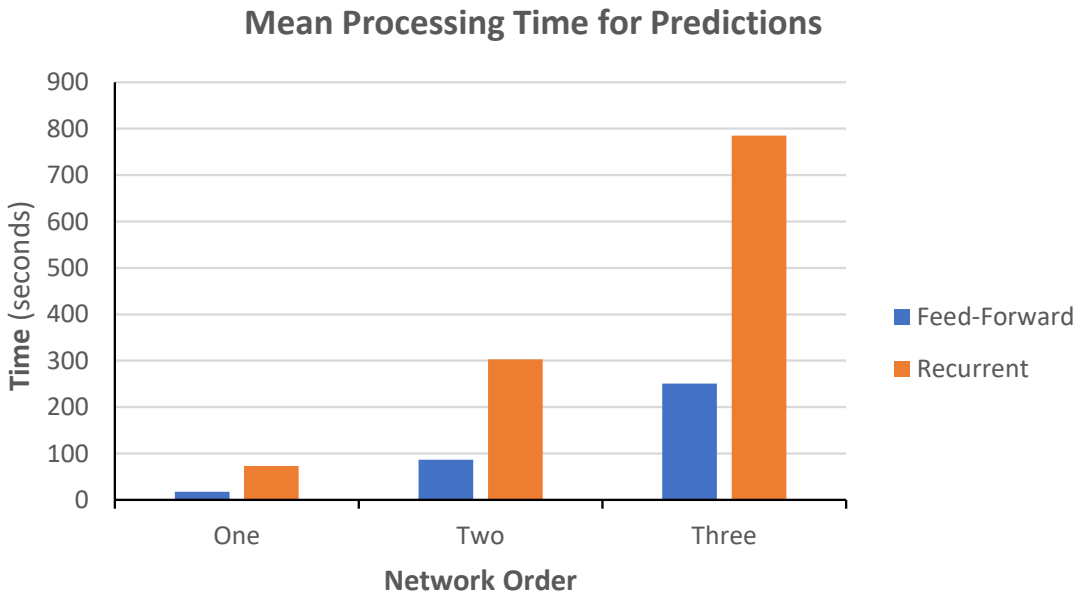
Based upon the significant difference in the computation times of the neural networks and a small difference between the two for prediction accuracy it can be considered that the recurrent network is redundant due to the amount of computational time taken. Both models have been executed on a server at the university which consists of a large number of cores and a significant amount of memory and if the hardware of this server takes a considerable amount of time then the the neural network is not the best option to be used within a in-car vehicle system, or an embedded system.

4.2.6 Comparison of the Neural Networks and Markov Models

The analysis of the neural network shows that the results are not favourable for use in a real-time constraint when compared to the results of the dynamic stochastic model with the random prediction or maximum-likelihood method. When comparing the mean processing time of the first order feed-forward neural network, it can be seen that the processing time was 17.38 seconds, whereas with a dynamic stochastic model with the random prediction method obtained a relatively low score of 0.0117 seconds (processing times are from the university server). The recurrent neural network came across the worse out of the two neural networks, with the mean time for a first order network being observed at 73.07 seconds. There is a reasonable explanation as to why the recurrent neural network took approximately four times as much time for processing, and that would be due to the number of epochs that have been selected to train the recurrent network. The number of epochs for training the feed-forward network was set to one; to ensure it was in-line with how the dynamic stochastic model works. However, the recurrent neural network was increased to 15 epochs due to the relatively low accuracy results that were gained when using only one epoch. The processing time for the first three orders of the Markov model and neural networks are shown in Figure 4.41.



(a) Markov Models



(b) Neural Networks

Figure 4.41: The comparison of mean processing times for the various neural networks and stochastic models that have been used in the experimentation for the pedestrian dataset.

From the graph it can be seen that the overall processing time of the neural networks is far greater in comparison to the all three of the stochastic models when combined. Although, it can be seen that the feed-forward neural network performs better than the recurrent network counterpart and this is most likely due to the simplicity of how the network performs and the number of epochs that have been chosen for the training phase. The graph shows that the best performing stochastic model was the traditional model; whereby the probability matrix has been pre-computed using the data that has been learnt by the dynamic stochastic model with the random prediction method.

The reason the traditional stochastic model is able to compute in a quicker manner is due to the reliance on not requiring to recalculate the probabilities each time a transition occurs within the model. This requires searching the entire matrix (which can be a costly process when a higher order is used) and then re-calculating the probabilities. This is then followed by performing the same search across the matrix rows for the prediction process. Therefore, the searching process is repeated twice which can add to the time of processing. Although this repetition exists on the dynamic stochastic model, it is seen that the two models performed within 0.02 seconds. Whilst the feed-forward network for a first order was not close to this figure with the network taking approximately 80 seconds. The data in Table 4.50 shows the mean accuracy for the stochastic models and the neural networks up-to a third order model.

Table 4.50: The mean accuracy of the stochastic models and neural networks for models of an order up to three. 'Dynamic^1' is the random prediction method and 'Dynamic^2' is the maximum-likelihood prediction method.

Order	**Stochastic Model Accuracy** (%)			**Neural Network Accuracy** (%)	
	Dynamic^1	**Dynamic^2**	**Traditional**	**Feed-Forward**	**Recurrent**
One	85.43	88.81	83.64	78.51	67.06
Two	88.48	90.76	86.79	69.43	72.82
Three	90.18	91.98	89.01	77.09	76.27

The table of results show that the stochastic model performed the best for the prediction accuracy, with the the range of results all ranging between 85% and 92%. On the other hand, the neural network did not perform as expected with the results ranging between 67% and 79%, approximately 6% less on the lower end of the range when compared to the stochastic models. The poor performance of the neural network could be attributed towards the lack of training that has been used to ensure that an accurate prediction is made; however, it was purposely chosen to have a low training rate to mimic the process of a stochastic model closely with an exception made on the recurrent network to stabilise the predictions. As initial testing of the network shown that a sufficient level of training was required in order to make predictions that were correct and not counting at 0%. A comparative graph of the results from Table 4.50 is shown in Figure 4.42.

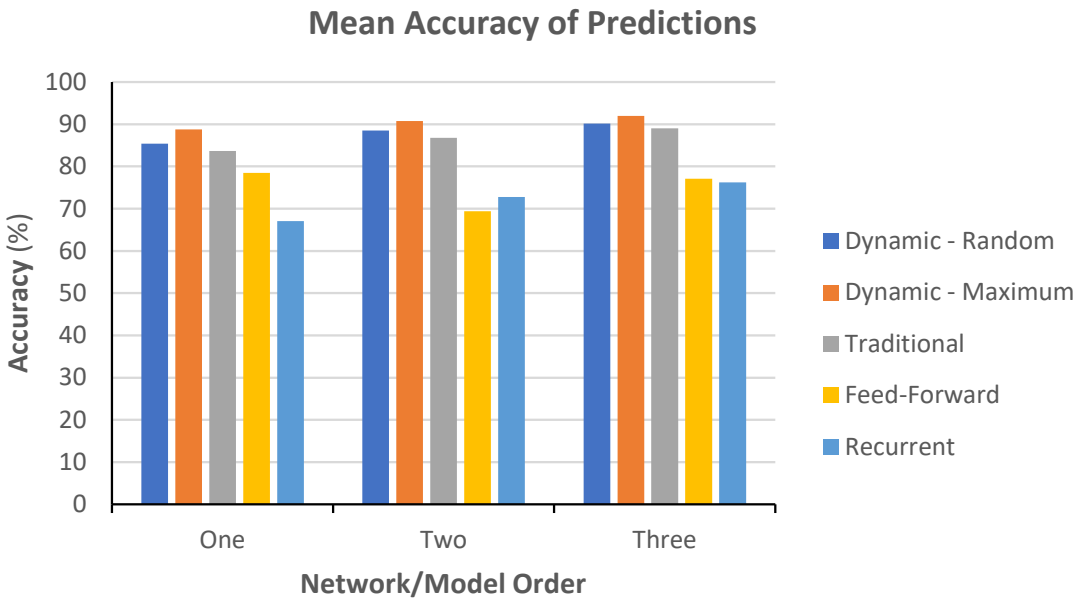


Figure 4.42: The comparison of mean processing times for the various neural networks and stochastic models that have been used in the experimentation for the pedestrian dataset.

From the graph it can be observed that there is a significant difference between the accuracy of the stochastic models and the neural networks. A delta of 21.75% is recorded between the recurrent neural network and the dynamic Markov model (with maximum-likelihood predictions). This shows that the implemented dynamic stochastic model is statistically better in comparison to the neural networks.

4.3 Results of Predicting an Objects Movement

With the ability to recognise an object from an image or video, the extracted features can be used to determine the position of the object within its environment. Using this data can assist in the prediction of an objects next direction of movement. In this section, the results of the various Markovian models and neural networks to make these predictions are discussed and analysed. The section will cover the various different methodologies employed to make a prediction and determine which of the methodologies are best suited for making a prediction within a real-time constraint. This section has been split into three key areas: the discussion on the dynamic and traditional Markovian models, and a third section on the processing time of the stochastic models.

4.3.1 Dynamic Markov Model

The results of the random prediction method for the video dataset have been subjected to a statistical descriptives test. The test will show the minimum, mean and maximum accuracy that was gained for the model and method. The data also includes the range (the difference between the minimum and maximum) and the standard deviation of the results. The statistical data is shown in Table 4.51.

Table 4.51: The descriptive statistics of the dynamic stochastic model using the random prediction generation method for the various videos.

Model Order	**Accuracy** (%)			**Standard Deviation**	**Delta**
	Minimum	**Mean**	**Maximum**		
One	62.91	86.34	96.11	12.38	33.20
Two	73.35	90.81	97.52	9.18	24.17
Three	80.49	92.87	98.14	6.74	17.65
Four	82.21	93.60	98.34	6.21	16.13
Five	84.66	94.42	98.61	5.42	13.96

From the results it can be seen that the minimum accuracy ranges from 62.91% for the first order, up to 84.66% for the fifth order. The progression through each order is linear, with no anomalous results in comparison to the stochastic models of the pedestrian dataset. It can be seen from the descriptives that the range of results are in-line to the outcomes that were expected of the model with a linear increase observed in the minimum, mean and maximum values for an increase in the order. Although it can be seen that for an increase in the order the difference between the minimum and maximum becomes smaller in size. The minimum, mean and maximum values are plotted and shown in Figure 4.43.

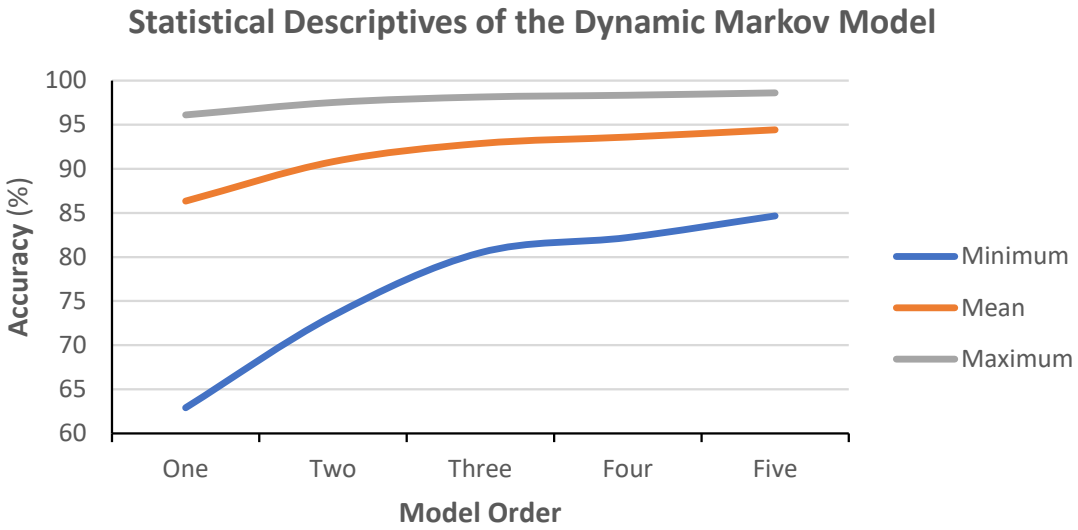


Figure 4.43: The minimum, maximum and mean values for each order of the dynamic stochastic model using the random method. The values are based upon the video dataset and shows the difference between the range of results.

It can be seen from the graph that the mean and maximum results are fairly close within proximity, especially as the order of the model increases towards the higher end. The minimum range of the accuracy saw a greater increase, especially in th range of orders one to three where the growth in prediction accuracy was 18.42%. The growth in accuracy then plateaus slightly between orders four and five, before a a slightly larger increase is made between orders four and five. It can be seen from minimum accuracy that the videos gained a fairly high accuracy across the various resolutions and different paths. The data in Table 4.52 shows the results of the experiment for the video of an object moving in the shape of a square.

Table 4.52: The results of the video where the object is moving within the shape of a square for the dynamic stochastic model with the random prediction method.

		Accuracy (%)			**FPS**		
		540p	**720p**	**1080p**	**540p**	**720p**	**1080p**
Order	One	62.91	82.93	93.04	36.60	21.86	12.99
	Two	73.35	88.31	96.58	35.98	21.78	12.82
	Three	80.49	90.16	97.45	33.51	22.84	13.02
	Four	82.21	91.12	97.93	33.70	23.11	12.98
	Five	84.66	91.79	98.13	32.80	23.01	12.88

It can be seen from the results that the minimum accuracy gained for an order of one model from both sets of videos was achieved in this video. It can be seen that by increasing the resolution of the video the accuracy of the predictions become better. However, there is a trade off with the FPS of the video decreasing. The decrease in the FPS for an increase in the resolution of the video is to be expected due to the larger collection of pixels that are within each frame that requires analysis. However, it can be seen that for a 540p (960 x 540) resolution, the videos can run obtained the desired frame rate. The desired frame rate is the frame rate at which the video was recorded in, for this instance the desired rate is thirty. Although some of the recorded FPS is over the desired number, the frame rate can be adjusted to ensure it does not overshoot the recorded FPS; but this is outside the scope of the research. Interestingly, it can be seen on the 720p resolution that the frame-rate of the video increases linearly for an increase in the order of the model. This goes against the recorded FPS of the 540p video where a decrease was seen. The

graph in Figure 4.44 plots the decrease of the FPS against the increase of the accuracy for a second order model.

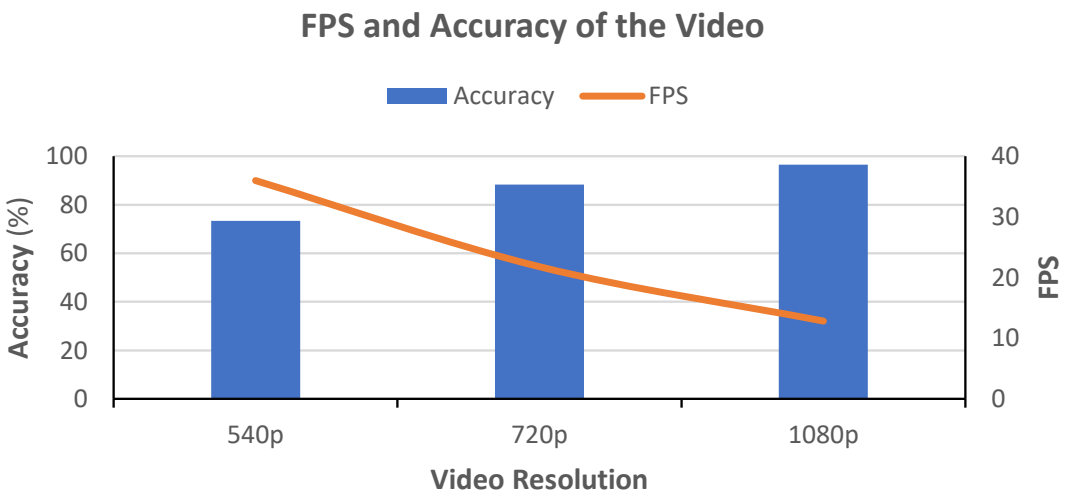


Figure 4.44: The graph shows a plot for the accuracy against the FPS of the video with an object moving in a trajectory shape of a rectangle.

To determine whether there is any relationship between the resolution of the video and the accuracy a correlation test is performed upon the data of the video using the Pearson's correlation method. The results of the test are shown in Table 4.53 and it can be seen that there is a relationship between the resolution of the video file and the accuracy of the predictions made.

Table 4.53: The statistics for the Pearson's correlation test using the IBM SPSS statistics software for the dynamic stochastic model using the random prediction method on the square videos.

	Accuracy				
Order	One	Two	Three	Four	Five
Resolution	0.929	0.938	0.963	0.965	0.975

This can be seen clearly within the descriptive statistics that are shown in Table 4.51. It can be seen on the fifth order that it gained the strongest relationship; and this can be seen in the accuracy that was gained for these videos, just over 98%. It is expected to see a similar result with the video that consists of the object moving within a straight line, but with a minor small increase in the prediction accuracy due to the simple path of movement. The results of the individual experiment on the straight line videos are shown in Table 4.54.

Table 4.54: The results of the video where the object is moving along a straight line for the dynamic stochastic model with the random prediction method.

		Accuracy (%)			**FPS**		
		540p	**720p**	**1080p**	**540p**	**720p**	**1080p**
Order	One	89.06	93.99	96.11	36.75	23.28	13.27
	Two	93.21	95.88	97.52	36.36	23.42	13.20
	Three	94.07	96.88	98.14	36.66	23.48	13.30
	Four	94.59	97.43	98.34	35.43	23.44	13.15
	Five	95.30	97.99	98.61	35.97	23.49	13.22

It can be seen from the results that they are significantly higher than that of the square video, as expected. The reason for the results being higher than that of the square video is due to the

simple nature of the movement that is being exhibited. The object is moving between the left and right and this is being repeated with intervals of pausing (stationary) movement on the extremes of the straight line. It can be seen across the minimum, maximum and mean values that there is linear increase across the different orders of the model. Similar to the square video, the increase in the order of a model sees a decrease in the FPS of the 540p video, but the 720p video was observed to see an increase in the accuracy. The 1080p video on the other hand saw a decrease between the first two orders of the model and then an increase in FPS for the third order, whilst again it decreased and then increased for the fourth and fifth orders. The alternating FPS of the videos can be seen in the graph in Figure 4.45.

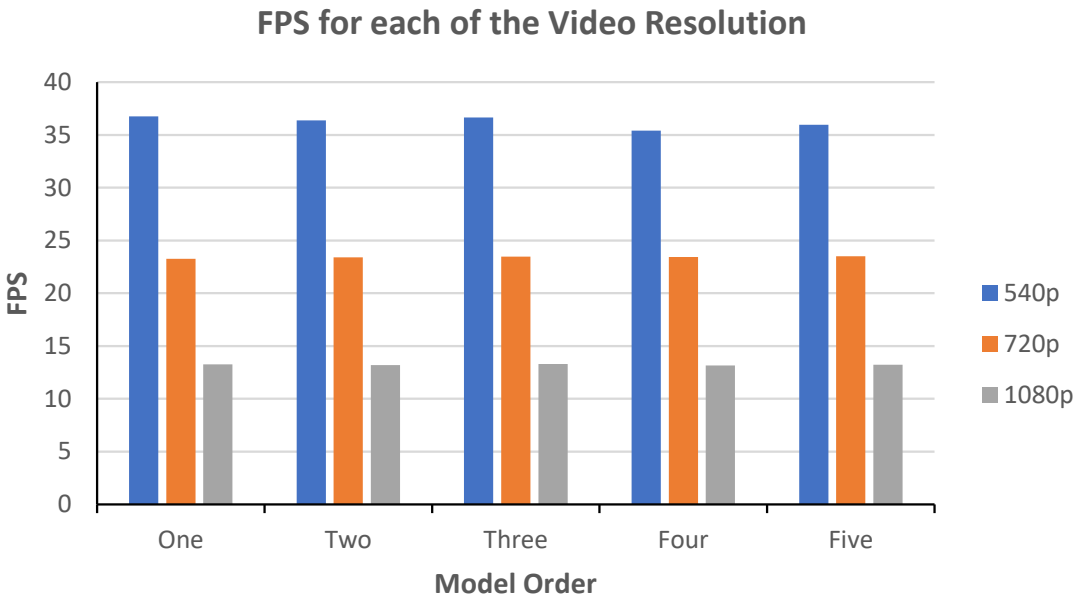


Figure 4.45: The graph shows how the FPS can decrease for each order of the dynamic stochastic model depending upon the resolution of the video.

It can be seen from the graph that between the 540p and 1080p video the frame rate of the video drops by over half the value that was achieved in the 540p video. It can also be seen on the 720p video the increase in the FPS for each order. Although they are all around a similar range (albeit the first order). To determine whether there is a relationship between the video resolution and the frame-rate a correlation test is performed upon the data using the Pearson's correlation test. The results of the text are found in Table 4.55 and it can be seen that there is a negative correlation between the resolution of the images and the frame-rate. The negative correlation is to be expected as the frame-rate decreases for an increase in the resolution of the video. It can be seen with the two videos that the random prediction method gains a fairly high accuracy between a simple movement such as moving in a straight line and something a little more complex such as travelling around in a square.

Table 4.55: The relationship values between the resolution of the video and the frame-rate recorded.

	FPS				
Order	One	Two	Three	Four	Five
Resolution	−0.962	−0.967	−0.965	−0.973	−0.970

From the results of the pedestrian dataset it is expected that the maximum-likelihood prediction method will also see an increase in the accuracy rate for the two videos in comparison to the random prediction method. The results of the experiment have been subjected to statistical descriptive

analysis; and the results are shown in Table 4.56.

Table 4.56: The descriptive statistics of the dynamic stochastic model using the maximum-likelihood prediction generation method for the various videos.

Model Order	**Accuracy** (%)			**Standard Deviation**	**Delta**
	Minimum	**Mean**	**Maximum**		
One	66.67	89.18	97.29	11.56	30.62
Two	78.73	93.04	98.13	7.39	19.41
Three	84.55	94.58	98.61	5.34	14.06
Four	85.80	95.14	98.83	4.99	13.03
Five	87.89	95.79	99.00	4.28	11.11

From the initial analysis of the results, it can be seen that the minimum accuracy rate from the first order sees an increase of approximately 3% in comparison to the random prediction method. The progression in the accuracy can be seen across all the orders of the stochastic model and a linear increase can be seen for the minimum, maximum and mean values. The range of difference between the minimum and maximum values start off to be fairly high at 30.62% but minimises rather drastically to 11.11% on the fifth order. The minimum accuracy achieved in the maximum-likelihood was 66.6% and was achieved on the video of the object moving within a square shape. The resolution of the video this was achieved on was 540p (the lowest out of the three resolutions used). The results of the square video and its varying orders are shown in Table 4.57.

Table 4.57: The results of the video where the object is moving in a square shape for the dynamic stochastic model with the maximum-likelihood prediction method.

		Accuracy (%)			**FPS**		
		540p	**720p**	**1080p**	**540p**	**720p**	**1080p**
Order	One	66.67	87.47	95.07	41.91	24.84	13.96
	Two	78.73	91.71	97.40	41.51	24.77	13.89
	Three	84.55	92.91	98.04	39.04	25.71	13.84
	Four	85.80	93.51	98.41	39.14	26.11	13.78
	Five	87.89	94.08	98.61	38.83	26.03	13.84

From the results it can be seen that the video sees a linear increase for when both the resolution of the video and order of the dynamic stochastic model increases. It is also noted that by using this method a higher frame rate can also be achieved with the FPS of the 540p video on a first order being recorded at 41.91%, whereas the same video on the random prediction method gained a frame rate of 36.60% a difference of 5.31%. It can be seen from the results that the FPS for the 540p image is over-running the desired frame rate between a range of 8.83% to 11.91% depending upon the order of the image. Processes can be put into place to control the over-running of the frame-rate for latter work. However, it is noted by previous experimentation that when using live-feed data such as web-cameras they do not suffer from this issue. The graph in Figure 4.46 shows the comparison between the mean accuracy of the random and maximum-likelihood methods.

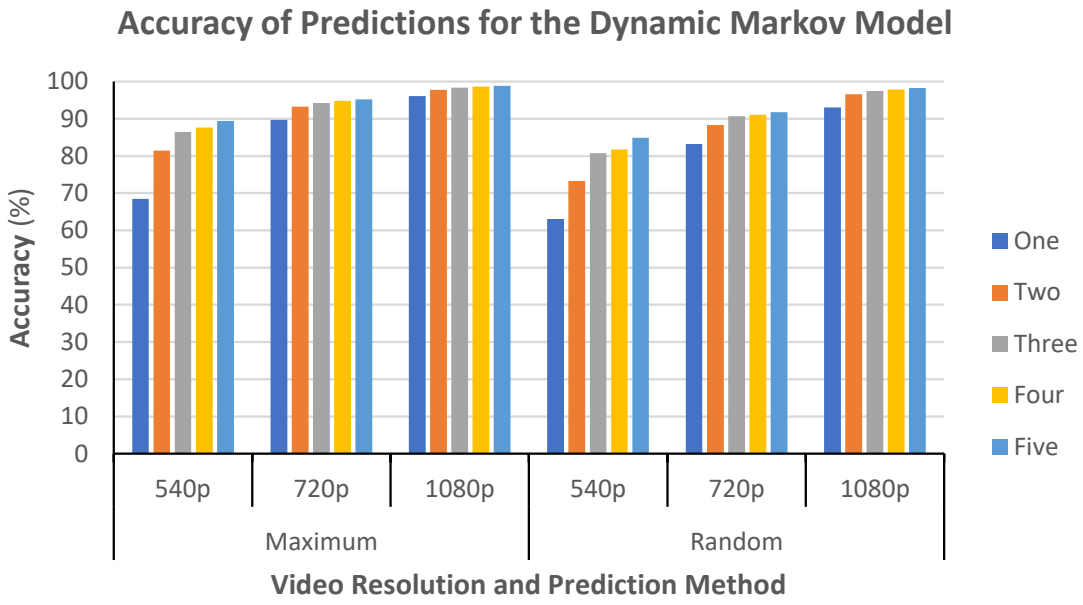


Figure 4.46: The difference between the random and maximum-likelihood prediction methods for the dynamic stochastic model on the square video.

It can be seen from the graph that the hypothesis of seeing an increase in the accuracy using the maximum-likelihood method was correct; although the increase was marginal, as displayed by the close gap between the two lines on the graph. Observing the graph it can be seen that the delta between the two methods becomes smaller as the order of the model is increased. With the difference becoming negligible when reaching the fifth order. Overall a linear increase can be seen between the two models with the increase being fairly smaller. However, this small increase is expected due to the nature of how the videos were generated. As the videos were digitally created using the Blender application, they have been generated that the paths are followed precisely by the object with no deviation of the object moving from the path. To determine whether there was a relationship between the accuracy and resolution of the square video a correlation test is performed using the Pearon's method and the results are shown in Table 4.58.

Table 4.58: The results of the Pearon's correlation test for a relationship between the resolution and accuracy of the dynamic stochastic model using the maximum-likelihood method.

	Accuracy				
Order	One	Two	Three	Four	Five
Resolution	0.899	0.916	0.947	0.950	0.961

From the correlation test it can be seen that there is a strong correlation between the resolution and the accuracy of the model. However, the correlation is not as strong in comparison to the random prediction method. To determine whether there is any difference with the FPS relationship with the resolution the Pearson's correlation test was performed, and the results are shown in Table 4.59.

Table 4.59: The correlation results of the Pearson's test showing how the resolution has a relationship with the FPS of the video.

	FPS				
Order	One	Two	Three	Four	Five
Resolution	-0.950	-0.952	-0.975	-0.979	-0.979

The results show that there is a similar relationship between the fourth and fifth order with the same value achieved. There is a negative relationship between the two values though. This is to be expected due to the nature of the decreasing FPS in relation to the size of the video. To check whether the FPS has any relationship to the accuracy rate of the model, a final correlation test is performed on the data, the results of this test is shown in Table 4.60. From the correlation test, it can be seen that the frame rate of the video can have a negative relationship with the accuracy.

Table 4.60: The results of the correlation test between the FPS and the accuracy of the dynamic stochastic model using the maximum-likelihood method.

	Accuracy					
Order	One	Two	Three	Four	Five	
One	−0.991	−0.996	−1.000	−1.000	−0.999	
Two	−0.990	−0.995	−1.000	−1.000	−0.999	
Three	−0.974	−0.982	−0.995	−0.996	−0.998	FPS
Four	−0.970	−0.979	−0.993	−0.994	−0.997	
Five	−0.959	−0.978	−0.992	−0.993	−0.997	

4.3.2 Traditional Markov Model

The traditional stochastic model uses the data exported from the dynamic model to provide a level of pre-training to the model to adjust the probability values. The purpose of this experiment was to determine whether the accuracy of the predictions made differ greatly to a model that can make learn the probability values whilst analysing the movements of an object. The results of the traditional stochastic model are subjected to a descriptives statistical analysis and the results are shown in Table 4.61.

Table 4.61: The descriptive statistics of the dynamic stochastic model using the random prediction generation method for pedestrians within 5 to 305 steps.

Model Order	**Accuracy** (%)			**Standard Deviation**	**Delta**
	Minimum	**Mean**	**Maximum**		
One	63.71	87.10	96.38	12.23	32.67
Two	74.54	91.33	97.69	8.77	23.15
Three	81.45	93.24	98.22	6.41	16.77
Four	83.01	93.93	98.49	5.95	15.48
Five	85.33	94.69	98.68	5.17	13.35

From the results it can be seen that the minimum accuracy of the first order model was 63.71%. It can be seen through the minimum accuracy rates a linear growth, with a delta between an order of one and five model being +11.52%. The results of the traditional model are similar to those of the dynamic stochastic model with a small deviation between the two results. This is to be expected as the probability matrices from the dynamic stochastic model has been used as historical data as prior 'learning'. With the results of the two models being close, it could be assumed that the dynamic stochastic model has no value or use when the data is simulated. The mean accuracy of the models are high and in-line with the expected results of the model. The mean accuracy for each order is above 87% for the first order model, whereas the fifth order model was recorded as 94.68%. The mean accuracy for the traditional stochastic model is plotted and shown in Figure 4.47 and shows the mean accuracy for the traditional model for both the square and straight video.

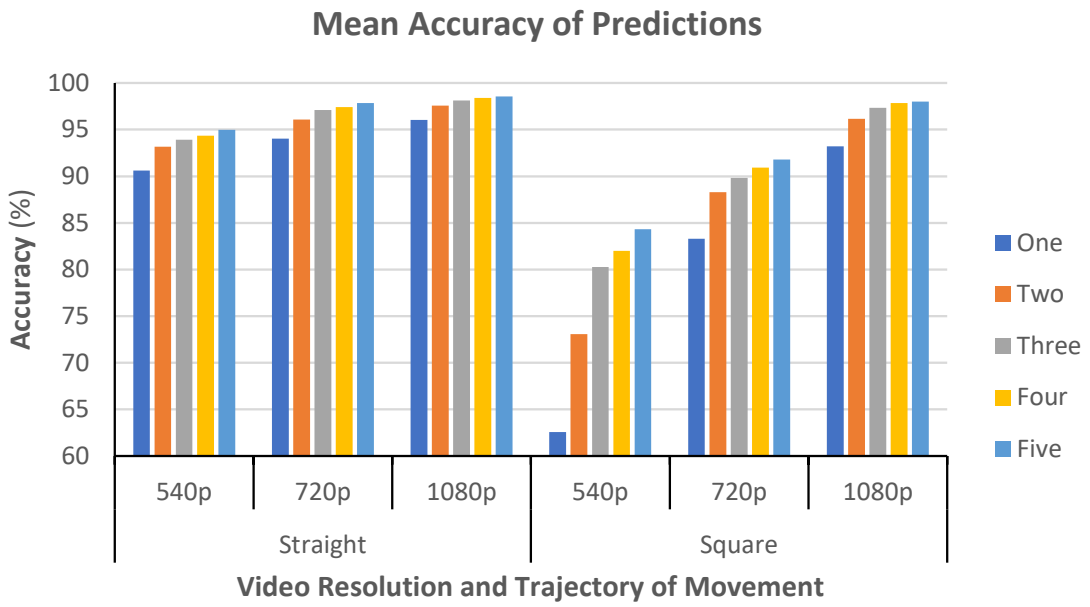


Figure 4.47: The mean accuracy plotted for the straight and square videos using the traditional stochastic model.

From the graph it can be seen that the square path video on the 720p resolution for orders three to five are very close within the accuracy rate of each other. The fifth order models shows a fairly straight growth in the accuracy of the model. However, the video with the straight line sees a similar growth, but plateaus between the 720p and 1080p resolution as it can be seen on the fifth order that the gain in accuracy is not as high. It is noticeable from the graph that the first order model is significantly lower than that of the other models that have been used in the model. This could be due to the number of previous movements that are being used for generating a prediction. Due to the first order model only using one single movement to make a prediction then the likelihood of selecting the incorrect transition state is more likely to handsome. This is more likely to happen when using the random prediction method too. The results of the object travelling within a straight line is shown in Table 4.62.

Table 4.62: Results of the experiment using the traditional stochastic model of an object travelling within a straight line for various resolutions.

		Accuracy (%)			**FPS**		
		540p	**720p**	**1080p**	**540p**	**720p**	**1080p**
Order	One	90.11	94.53	96.38	41.91	26.56	14.05
	Two	93.61	96.32	97.69	41.82	26.65	13.94
	Three	94.37	97.14	98.22	41.19	26.37	14.28
	Four	94.87	97.64	98.49	41.86	26.15	13.94
	Five	95.53	98.08	98.68	42.70	26.05	14.05

From the table of results, it can be seen that the accuracy rates of each resolution and order was greater than 90%. Due to the simplicity of the path of motion that was chosen for implementation it is expected to see such a high accuracy rate be gained using the traditional model. The object can move in one of three movements (left, right and stationary) and therefore this limits predictions that can be made. It can be seen by looking at the probability matrix of this video that only three states are ever visited for the first order, as shown in matrix 4.7.

$$
\begin{array}{c}
 \\
[(0,0)] \\
[(-1,1)] \\
[(1,1)]
\end{array}
\begin{array}{c}
\begin{array}{ccccccccc}
[-1,-1] & [-1,0] & [-1,1] & [0,-1] & [0,0] & [0,1] & [1,-1] & [1,0] & [1,1]
\end{array} \\
\begin{bmatrix}
0 & 0 & 0.3\dot{3} & 0 & 0 & 0.3\dot{3} & 0 & 0 & 0.3\dot{3} \\
0 & 0 & 0.999 & 0 & 0 & 0.001 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.001 & 0 & 0 & 0.999
\end{bmatrix}
\end{array}
\tag{4.7}
$$

It can be seen from the matrices that the object moved within one of three directions; and that the chance of moving left or right had the highest probability. This would make sense as the amount of time that the object spent stationary within the video was minimal. It can also be seen from the matrix that there is an equal chance of transitioning from a stationary movement to either left, right or being stationary again. To determine whether there is any difference between the matrices for the accuracy rate to be higher in the third order. By observing the matrix of a third order model, it can be seen from the third order matrix that nine rows have been adjusted out of 729. It can be seen that the rows of the matrix that have been adjusted follow the same movement path as the first order but adjusted to include the past history of movement. For example, an object may follow the path whereby its last three transitions were stationary for two and then moving left would look like: $[(0,0),(0,0),(-1,0)]$. The third order matrix shows that 25 rows were adjusted when tracking the movement of the object. The number of rows have increased due to the various states that the objects previous movement has been classified as. For example, one state would be stationary with three past movements, whilst a second state would consist of two stationary movements and a movement to the left. This will continue where there is only one stationary movement and then two movements to the left. However, with this increase in the number of rows being updated; it in-turn will provide a more accurate prediction.

4.3.3 Processing Times of the Markov Models

Previous experimentation with feature detectors in the methodology chapter shows that the amount of time taken for detecting an object from a video for higher resolutions greater than 540p takes a significant amount of time. This results in the video taking a longer time to compute than the intended run-length. The results in Table 4.63 are the mean times for the object travelling within a straight line for the stochastic models.

Table 4.63: The processing time of each hardware for the respective stochastic model for the video of an object travelling within a straight line for the various resolutions. 'Dynamic[1]' are the results of the dynamic stochastic model with random predictions and 'Dynamic[2]' are the times of the maximum-likelihood prediction method.

Straight Video: 540p Resolution						
Model Order	**Server**			**Laptop**		
	Dynamic[1]	**Dynamic[2]**	**Traditional**	**Dynamic[1]**	**Dynamic[2]**	**Traditional**
One	642.87	488.83	488.83	413.27	403.67	417.57
Two	649.37	489.33	489.33	419.03	402.73	419.57
Three	641.93	490.93	490.93	417.1	402.5	436.73
Four	648.53	488.8	488.8	445.2	402.93	421.17
Five	645.93	485.83	485.83	429.27	405.33	401.47

Straight Video: 720p Resolution						
Model Order	**Server**			**Laptop**		
	Dynamic[1]	**Dynamic[2]**	**Traditional**	**Dynamic[1]**	**Dynamic[2]**	**Traditional**
One	942.7	770.43	770.43	713.37	655.2	664.37
Two	948.3	755.87	755.87	697.07	657.3	668.93
Three	937.0	766.6	766.6	702.8	664.8	676.83
Four	956.13	768.8	768.8	689.07	660.93	689.27
Five	957.27	775.37	775.37	687.37	663.9	694.73

Straight Video: 1080p Resolution						
Model Order	**Server**			**Laptop**		
	Dynamic[1]	**Dynamic[2]**	**Traditional**	**Dynamic[1]**	**Dynamic[2]**	**Traditional**
One	1621.2	1437.03	1437.03	1330.73	1326.1	1340.8
Two	1657.87	1459.53	1459.53	1329.53	1306.27	1338.1
Three	1658.63	1427.23	1427.23	1336.17	1278.7	1312.2
Four	1661.4	1451.63	1451.63	1353.43	1287.0	1337.87
Five	1662.0	1474.67	1474.67	1324.5	1313.93	1329.5

From the results, it can be seen that the laptop performed the best whilst processing the video across the three different media resolutions that have been used. For example, on the 540p video, the order of one model for the dynamic stochastic model with random predictions was able to compute the video in 413.27 seconds (approximately six minutes), which was a reduction of 230 seconds in comparison to the server model. The difference can be seen across the various models for the same resolution and the increase in the amount of time taken on the server could be due to the lack of libraries required to ensure that the video processing libraries of OpenCV are not available and thus the libraries have not been optimised. However, it can be seen that this large difference is only seen on the model with random predictions. All the other models and the varying resolutions see a comparable time when compared to each other. For example, it can be see on the server and laptop that the times of processing for the dynamic (with maximum-likelihood prediction method) and the traditional stochastic models are both comparable (within the same time stamp) of processing for each order of the models, i.e an order of two model was $1,459.53$ and $1,459.53$ seconds, for both the dynamic and traditional model.

Excluding the results of the dynamic model with random predictions from the results of the 540p video; it can be seen that the resolution performed the best for computation time. The recorded times are within two minutes of the original run-length of the video; whereby the original length of the video was five minutes. On the laptop the best performing model was the dynamic stochastic mode with maximum-likelihood prediction method of which computed within 403.67 seconds (approximately seven minutes). The worse performance was the random prediction method which was recorded at 642.76 seconds for a first order model. However, a significant increase was not observed when the orders of the model was increase. This can be seen across the collection of the result and the graph in Figure 4.48 shows the times of the video for the traditional stochastic model for each resolution on the laptop.

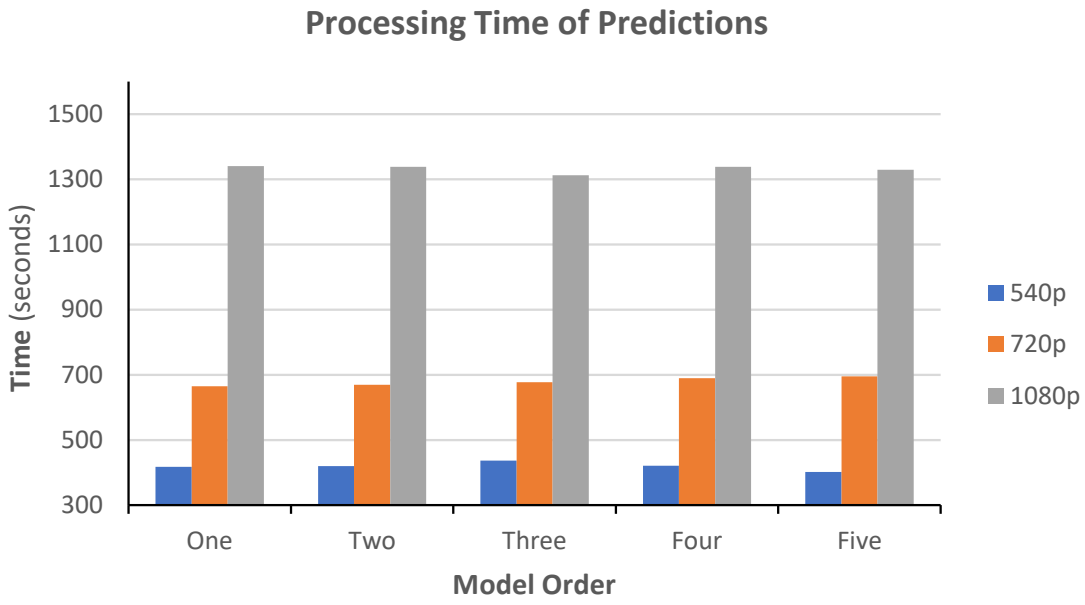


Figure 4.48: The observed computation times for the an object moving in a straight line for the traditional stochastic model on a Dell XPS laptop.

It can be seen from the graph that the recorded times for the 540p and 720p vary between the orders of the model. However, it is seen that the 720p follows a linear progression, whereby the time increases in-line with the order of the model. The varying times that were recorded for the different orders of the model for the same resolution could be due to other processes running in the background that is taking a larger portion of the system resources and therefore affecting the times of the model. The results of the square video processing time is shown in Table 4.64. The results of the object moving within the contours of a square are similar to those of the object travelling within a straight line. This is to be expected as the run-length of the videos were the same, and the feature detection method has been fine-tuned to ensure that the same parameters are used across both videos to provide a robust detection method. Using a hybrid of feature detectors to recognise the object from the scene has provided results that can ensure the stochastic models and feature detection process can be computed within real-time for a resolution of 960-by-540 pixels, with an achieved FPS of 41.98. However, it can be seen that with the higher resolutions that the prospects of achieving a real-time recognition and prediction is unfavourable. The FPS for the 720p video on the Dell XPS laptop averaged to be 26.12, whilst the 1080p FPS was recorded to be 14.29. It can be clearly seen that the 720p was within the realms of possibly meeting the real-time constraint. On the other hand, the 1080p video was far from it, with the frame-rate being over half of the intended rate.

Table 4.64: The processing time of each hardware for the respective stochastic model for the video of an object travelling within the shape of a square for the various resolutions. 'Dynamic[1]' are the results of the dynamic stochastic model with random predictions and 'Dynamic[2]' are the times of the maximum-likelihood prediction method.

Square Video: 540p Resolution						
Model Order	**Server**			**Laptop**		
	Dynamic[1]	**Dynamic[2]**	**Traditional**	**Dynamic[1]**	**Dynamic[2]**	**Traditional**
One	646.6	497.33	497.33	414.37	408.63	432.03
Two	654.33	506.93	506.93	424.97	409.8	431.97
Three	656.57	584.33	584.33	502.3	411.3	424.33
Four	648.73	576.67	576.67	496.4	411.47	417.57
Five	654.7	593.47	593.47	524.6	413.13	421.7

Square Video: 720p Resolution						
Model Order	**Server**			**Laptop**		
	Dynamic[1]	**Dynamic[2]**	**Traditional**	**Dynamic[1]**	**Dynamic[2]**	**Traditional**
One	967.5	915.27	915.27	818.4	664.27	725.23
Two	968.87	929.7	929.7	833.33	664.67	727.0
Three	962.67	833.13	833.13	734.87	665.83	698.97
Four	964.93	802.63	802.63	712.0	666.73	705.3
Five	973.43	801.2	801.2	712.43	672.83	704.13

Square Video: 1080p Resolution						
Model Order	**Server**			**Laptop**		
	Dynamic[1]	**Dynamic[2]**	**Traditional**	**Dynamic[1]**	**Dynamic[2]**	**Traditional**
One	1667.63	1505.5	1505.5	1369.43	1293.6	1333.0
Two	1678.5	1499.33	1499.33	1416.63	1313.07	1356.87
Three	1666.5	1460.83	1460.83	1337.33	1316.5	1322.4
Four	1672.07	1481.13	1481.13	1340.33	1317.57	1360.27
Five	1684.57	1498.37	1498.37	1366.6	1297.4	1361.67

The recorded frame-rate of the server did not match that of the Dell XPS laptop, which is surprising considering the power of the hardware that is found within the unit. The mean frame-rate of the server for the 540p video was recorded to be 28.76 which was a reduction of 13.22 to the figure that was observed on the laptop. The drop in FPS for the server could be due to a limitation with the correct codecs required for the video libraries of the OpenCV framework. Therefore, it affected the overall processing of the video.

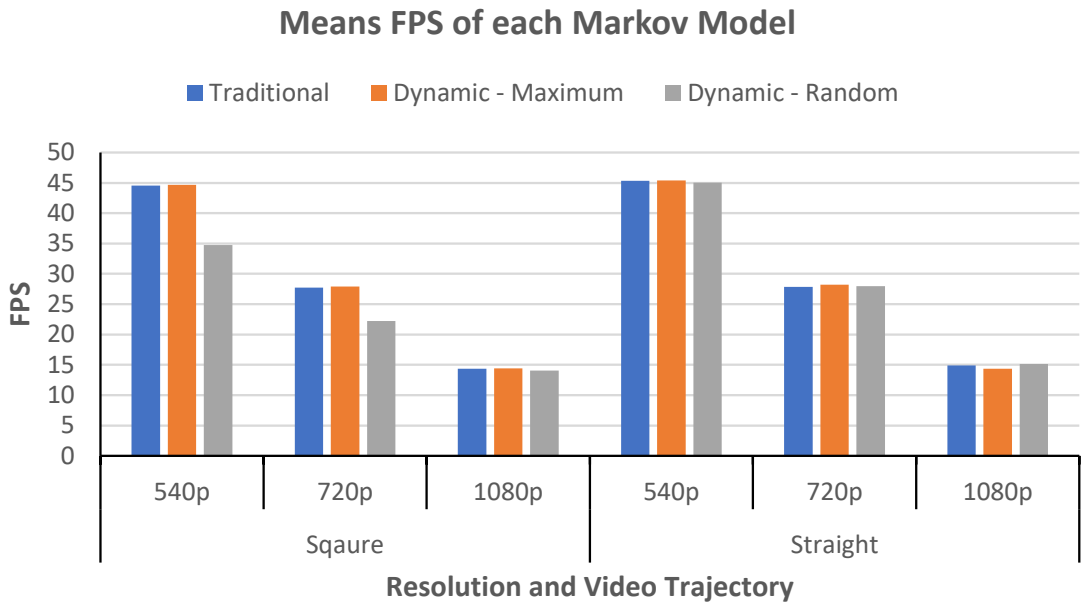


Figure 4.49: The mean FPS of the two videos for each order of the varying stochastic models used for the experimentation

.

The mean FPS of the two videos are shown in Figure 4.49 and it can be seen that there is a steep drop with the FPS of each video between the 540p resolution and 720p, with the same drop also recorded for the 720p and 1080p resolutions. The mean FPS of the random and maximum-likelihood prediction methods were consistent between the two videos, with a similar value recorded. This can be seen in Table 4.64, where the same times were recorded for the dynamic (with maximum predictions) and traditional stochastic models.

4.4 Conclusion of the Object Recognition Experiment

Based upon the experiments performed, it can be seen that although the BRISK detector is computationally efficient and able to detect an object from its environment it is unable to do so within a real-time constraint. However, when using the detector in a hybrid form with the SURF detector the possibility of this constraint is within reach. Based upon the experiments performed, it can be seen that there are limitations to using the BRISK and SURF detectors:

- **misdetection whilst moving**: the detector can often incorrectly detect the object from the environment when its moving too fast within its environment
- **misdetection based upon colour**: the detector can incorrectly detect the object when the wrong training image is used, especially when the only property changed is the colour
- **frame-rate fluctuations**: the frame-rate of a video can easily fluctuate to either over-processing a video (the FPS is higher than the recorded FPS) or where the FPS is below that of the desired number
- **poor performance on high-resolutions**: the frame-rate for videos where a resolution exceeds 540p is below the desired frame-rate

To overcome these limitations several methods have been proposed and can be implemented to reduce the number of misdetection that occur and ensure that a steady frame-rate can be maintained. Although these methodologies do not remove the limitations entirely, it can minimise them to a degree where the accuracy of predictions made are not affected inadvertently, and the processing of videos or live-feed data does not fall below a certain threshold. In order to minimise the misdetection of an object when it is moving in an environment, adjust the parameters of BRISK can be applied that will in-turn adjust the sampling pattern of the detector to ensure that a more robust detection can be made. From the experiments it can be seen that BRISK is a robust detector and maintains a fairly adequate frame-rate when processing a high-definition video. However, by adjusting the parameters of BRISK it can impact the frame-rate of a video and therefore by adjusting them too much it could severely impact the FPS of the recorded video, which is an important constraint when trying to predict a direction of movement in real-time. The data in Table 4.65 shows how adjusting just one parameter of BRISK can affect the frame-rate of the video rather significantly.

Table 4.65: The data in the table shows how by adjusting the threshold of the BRISK detector can increase the number of keypoints detected but can also inadvertently affect the frame-rate rather significantly.

Parameter			**# of**	**Average**
Threshold	**# of Octaves**	**Pattern Scale**	**Keypoints Detected**	**Frame-Rate**
50	5	0.5	715	33.19
60			528	42.62
70			376	52.32

Due to the influence of changing a threshold that it can have on the frame-rate of a video and the number of keypoints detected, a medium is required to be found. Having a large number of keypoints could provide an ample fix to the misdetection of an object whilst it is moving through the scene, but it is fundamental that the frame-rate is not negatively impacted. However, adjusting the threshold too much to increase the frame-rate could increase the chances of a misdetection happening and thus affecting the calculation of the boundaries for the detected object within the environment as shown in Figure 4.50. Based upon the findings in Table 4.65 it can be seen that the optimum threshold value is 50, whilst retaining the previous number of octaves and pattern scale of the experimentation performed. However, it can be seen that by using this threshold the frame-rate of the video is over the recorded frame rate of the video (approximately +9.6%). However, when using live-feed data (such as web-cams) the frame-rate will not go over the desired frame-rate, and therefore this would not be a major impact when using live-feed data for real-time analysis. It could be an issue when the frame-rate of the video is below the desired number, whereby the data being predicted is based upon previous movement and not the current.

Figure 4.50: The miscalculated boundaries of a detected object. This can occur due to the lower number of keypoints detected when the object is moving; or when the incorrect training image is used.

The chance of a misdetection still arising is slimmer than previously, but it could still happened and therefore a secondary process is applied to the calculation of the central point. The secondary process will ensure that the central point does not deviate too sporadically and is achieved by using the exponentially weighted moving average (EWMA) formulae,

$$S_t = \begin{cases} Y_1, & t = 1 \\ \alpha \cdot Y_t + (1 - \alpha) \cdot S_{t-1}, & t < 1 \end{cases} \tag{4.8}$$

where α is a user-defined weighting between the values of zero and one, Y_t the objects location for a given time period and S_t the smoothed value for the given time period t. The formulae applied to the central point should ensure that a smoothed value is given if a large difference between the current value and the next is present due to a misdetection. Using the EWMA value should ensure that the accuracy of the model remaining high for any misdetection that may occur due to the limitation of the BRISK detector.

The second issue with the BRISK detector is the misdetection of an object when the incorrect training image has been used. Although this is a relatively small issue and could be easily fixed by ensuring the correct training image is used, it is something that could be inadvertently overlooked. BRISK works by localises keypoints of interesting upon the corners of an image and therefore does not take into account any underlying properties such as colour. Similar objects, whereby only the colour changes, can pose an interesting issue when using the BRISK detector, i.e. identification cards. Coventry University employ the use of identification cards to identify students based upon their profile within the university: student, postgraduate student or staff member; and each card has a different colour associated to it (as shown in Figure 4.51.

(a) Research ID **(b)** Staff ID

Figure 4.51: Two of the identification cards that are used at Coventry University with the two cards being very similar apart from a major difference in the colour.

It can be seen from the image shown in Figure 4.51 that the two cards are very similar with only a subtle differences being the text that is present on the card, but a major difference being the colour of each card (one being red and the other blue). The BRISK detector will localise keypoints upon the text of the identification cards and therefore disposing the detail of the backgrounds colour; this could provide a ramification upon the detection of the object as the two cards have a major similarity in the background pattern, logo and placement of the students photograph. Due to the high similarity between the cards a false-positive could occur in the matching process and therefore lead to the misdetection of the object and its boundaries (as shown in Figure 4.50. A process could be put in place to overcome this issue by using the k-means clustering algorithm to split an image into its base colours, as discussed by Chen et al. (2008). The simplification of an image to its base colours will enable a comparison to be a made between the two images using the RGB colour-space. In order to ensure that this process works, the colour model of both images are required to be the same; otherwise the base colour values for each image would differ. If this instance was to occur, then a check could be made to ensure that the two values fall within a defined range. However, the process of segmenting the image by its colour is a process that relies heavily upon computational resources and could increase the overall processing time of the video (in-turn affecting the frame-rate of the video). To ensure that this method does not affect the frame-rate a check could be performed every n^{th} frame, or the scene of the objects environment could be cropped to a small subsection of an area where the object was detected to compare the colours.

Another issue that arose during the experimentation with the videos on the Dell XPS laptop was with the inconsistent fluctuation of the frame-rate (along with the poor performance of the high-resolution videos). The inconsistency of the frame-rate on the videos with the BRISK detection could hinder the applicability of being able to predict the direction of movement for an object within the real-time constraint. However, as the purpose of this study is to evaluate the stochastic model with a newly implemented quirk it is decided that trying to implement a feature to overcome this limitation is not required. However, it is proposed that to ensure that videos of a high-resolution (more notably greater than 960x540 pixels) are able to be computed within an appropriate time, they are downscaled to 540p. Based upon the experimentation it can be seen that the BRISK detector is highly favourable when computing high-resolution images, a decrease was recorded of -80% when comparing against SIFT on the 1080p video. Although previous work has shown that SIFT is a highly-robust feature detector when considering photo-geometric transformations it unfortunately has a large redundancy on computation resources. Therefore, the high computational cost of the detector out-weighed the benefits. The overall time of execution for the detectors on the high-definition videos is shown in Table 4.66.

Table 4.66: The times taken for localising keypoints and extracting descriptors using the three feature detectors for the high-definition video dataset.

Image Resolution	**Descriptor Extraction Time** (ms)		
	BRISK	**SIFT**	**SURF**
540p	0.030	0.115	0.160
720p	0.047	0.187	0.285
1080p	0.080	0.405	0.489

From the table, it can be seen that the SIFT detector was recorded to take 0.405 milliseconds to process the 540p image, whereas BRISK took significantly less time and computed the image within 0.080 seconds for the experiment performed upon the Dell XPS laptop. The SURF detector on the other hand performed the worse with the results unexpected when compared to the performance of the detector on the dataset by Lowe (2004). Overall, it can be seen that the results show that SIFT and SURF (the traditional descriptors) are not optimised for the task of recognising an object within a real-time constraint. Although BRISK was unable to meed the criteria of performing within this constraint due to fluctuations being recorded in the frame-rate it is a much better contender when compared to the other detectors. This could be attributed towards the fact that BRISK uses binary descriptors and is based upon a comparison of the light intensity within the images and is far quicker to computer compared to a conventional descriptor.

4.5 Conclusion of the Prediction Experiments

From the analysis of the results it can be seen that the traditional stochastic model performs the best with regards of prediction accuracy and the amount of time that was observed for generating the predictions. However, this type of model requires the prior learning of data to re-adjust the probability matrix of the model and it is not always readily available for use. However, in a comparable process the introduction of a second matrix to count the transitions between states and just the probabilities based upon the transitions can result in an accuracy that is accurate with a traditional stochastic model. However, a trade off with the dynamic stochastic model is the method of recalculating the probabilities which can add time to the overall processing and generating of the prediction method. It has been touched upon that in order to update the probabilities the algorithm searches the matrix for the specific row that is allocated to the tracked movement and then this row is updated with the new probability value. The searching process is then repeated to find the appropriate row to make a prediction. Therefore, this could add a redundancy to the dynamic model and in comparison to the traditional model. This is why it can be seen that the traditional model has a far lower computation time in comparison to the dynamic model with a random prediction method, as shown in Figure 4.52.

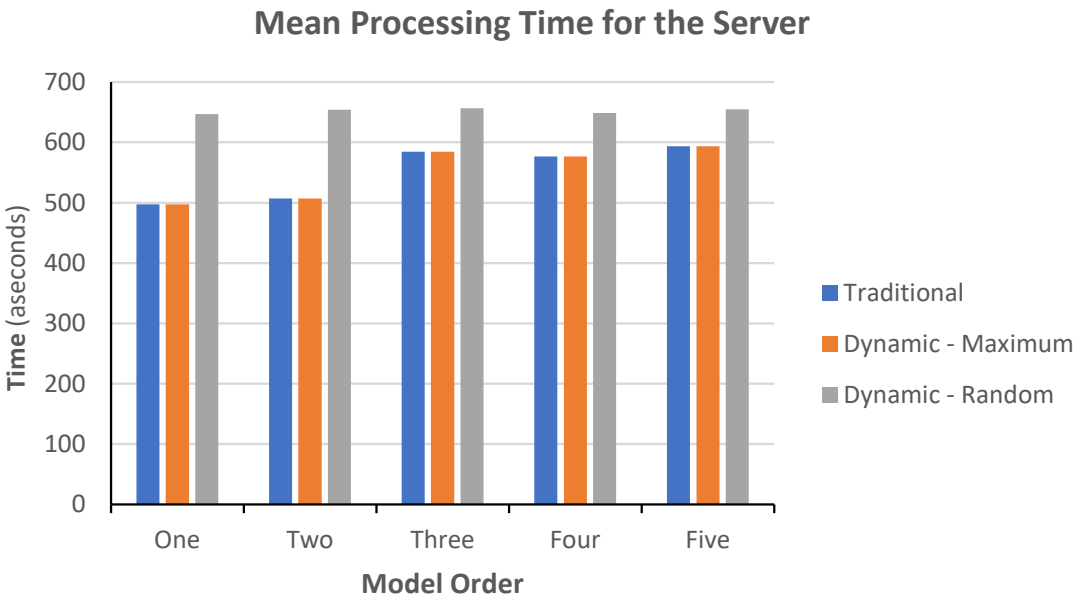


Figure 4.52: The mean processing time of the 540p video of an object moving in a square trajectory.

It can be seen from the graph that the growth in the computation time between a traditional model and the dynamic mode with the random prediction method grows largely. For example the difference for the first order model sees an increase of 130%, whilst the third order model sees a growth of 112.36%. However, the traditional model requires the model to be pre-loaded with the data that was obtained from the learning that was undertaken in the dynamic stochastic model. Therefore, it does not have this method of updating matrices 'on-the-fly'. It can be seen that the accuracy of the dynamic stochastic model is similar to the traditional model but the traditional made sees a larger abundance of pedestrians that gain an accuracy of 100%. There was a total of $2,332$ pedestrians that gained this accuracy rate with the traditional model. Whereas the dynamic stochastic model with the random and maximum-likelihood prediction methods saw none and 386 pedestrians, respectively. The results of the random prediction method are not surprising, to not see any pedestrians gain a 100% accuracy is expected due to the nature of prediction method works.

Overall, it can be seen that the dynamic stochastic models can provide an accuracy that is accurate and can be computed within a time that is respectable for the use in the machine vision systems that require predictions to be computed within a real-time constraint. However, the traditional model would be the ideal selection if a hybrid was to be created between the two models that have been implemented; as discussed in Section 5.2. It can be seen that the dynamic stochastic model provides a good balance between the accuracy of the predictions and the amount of processing time that is required. Based upon the analysis of results, it can be seen that the stochastic model works best when the pedestrian is travelling within a path that is not too complex. However, the complexer paths do obtain a relatively high accuracy in comparison to a traditional stochastic model that requires a priori data.

The outcome of the neural network were relatively weak with the accuracy of the predictions that were gained and the amount of time that was taken for processing. It can be seen from the graph in Figure 4.53 that the amount of time taken for the recurrent neural network is significantly greater than that of the feed-forward neural network.

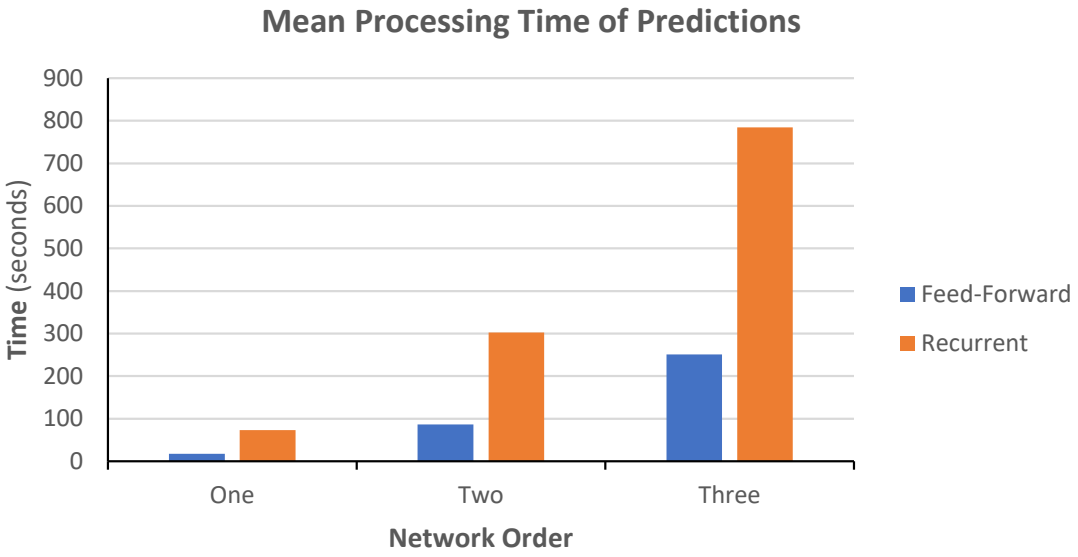


Figure 4.53: The mean processing time of the pedestrian dataset on the university server using a neural network.

As explained earlier, this is due to the amount of epochs that have been chosen for the model to learn the data that is being inserted into the network. It can be seen for an increase in the number of past movements that are feed into the network an increase is observed in the recorded times. The worst performing models were the third order, whereby the feed-forward and recurrent neural network were both recorded to have a mean processing time of 251 and 785 seconds, respectively. Although the feed-forward network performed better due to the simplicity of the networks nature. The mean accuracy for each order of the recurrent network were better and saw a linear progression. The details of the mean accuracy for the neural networks can be seen in Table 4.67 and it can be seen that although the feed-forward neural network performed the best it did not have any linearity in the progression when the number of past movements were increased. The highest accuracy gained for the feed-forward network was 78.50% whereas the highest on the recurrent network was 76.27%. Both of which were obtained on a different order model.

Table 4.67: The mean accuracy that was achieved for each of the neural networks that were used within the study.

Order	**Neural Network Prediction Accuracy** (%)	
	Feed-forward	**Recurrent**
One	78.50	67.06
Two	69.43	72.81
Three	77.09	76.27

The accuracy of the neural networks are similar to that of the stochastic model, but take far more computational resources that it does not make it applicable for the purpose of this study. It is worth noting that the neural network was implemented to provide a comparative study with the stochastic model due to the popularity that neural networks have in literature for predicting time-series events.

Chapter 5

Conclusion

The work undertaken in this thesis is vast and can be split into two key areas:

- a hybrid approach to the recognition of an object using two feature detectors
- the adaptation of a pre-existing stochastic model to ensure highly accurate predictions in real-time

The first half of the work has been developed to ensure that a robust recognition of an object can be performed on mobile devices without impacting the flow of data. The work has primarily been designed to take into account the evolving nature of hardware that can be found within mobile devices. It can be seen in literature that traditional feature detectors such as SIFT and SURF are computationally expensive to run and require a large amount of processing power (Mikolajczyk & Schmid 2005, Leutenegger et al. 2011). The introduction of BRISK has enabled developers to extract feature representations of an image within a time constraint that can be deemed to be run in a real-time constraint (Leutenegger et al. 2011). Therefore, the work undertaken in this thesis has taken each individual feature detector to determine the amount of time that is taken to detect an object from high-resolution images and video datasets. The results have shown that the traditional detectors, as expected, faired the worse with the average computation time of SIFT and SURF polling at xx and xx seconds, respectively. Using a mixture of SURF and BRISK has shown that this method of detection can be performed within 0.048 seconds on a 540p video of an object travelling in a straight line, whilst the sole use of BRISK on a static image in the same resolution was computed within 0.214 seconds. From the collection of results, it can be seen that the hybrid approach to feature detection not only keeps the robustness of a traditional detector, but maintains a speedy method of detection from the binary detector.

The new approach that has been introduced in this thesis is useful for applications that require a recognition to be computed in a real-time constraint. This can ideally be used on applications that are built on mobile hardware such as mobile phones and tablets, or used on embedded systems that are naturally limited/constrained on the hardware that is used, most notably in-vehicle systems. The work undertaken in this area has been used in the second body of work that has been studied, the adaptation of a stochastic model to generate predictions in a real-time constraint. The latter half of the thesis is considered to be the main contribution of the thesis with the use of hybrid detection framework previously being used within the works by Coltin et al. (2016). The stochastic model of choice, a Markov Model (Markov 1906), has been adapted to account for the transitions that occur between states. This has been implemented by including a secondary vector that takes into account these transitions that occur and is termed as a 'frequency' vector. The recorded state transitions are used to refine the values that are stored in the probability vector and are continuously updated for the run-length of the simulation/modelling. Results of the study have shown that the prediction accuracy for this new method of modelling is between 65% and 95%,

dependant upon the type of prediction method that is utilised and the type of data that is being inserted. To ensure that the model is fairly compared with the traditional implementation of a Markov model, two types of prediction methods have been used: randomly generated, and the maximum-likelihood method. It can be seen that the two methods have performed exceptionally well, with a respective accuracy ranging between $85.45\% - 91.78\%$ and $88.81\% - 93.09\%$ for the pedestrian dataset by Yi et al. (2016*a*). Comparing the results of the DMM with the traditional model employed by Krumm (2008) and Nižetić et al. (2009) it can be seen that the implemented model provides a higher accuracy overall.

This process does not require a high-level of machine learning or priori data to gain the level of accuracy that is often required for the likes of neural networks and the traditional Markov models. It is often seen with these methods that a lot of the time is spent on training the model to ensure that a high accuracy can be achieved on the predictions. Whilst these methods have been proven to be useful in previous applications, their use for models that require to be computed within real-time is not suitable. The results of the neural networks show that amount of computation time that has been used to ensure an accurate prediction can impact the real-time applicability of the network. It can be seen that a dynamic Markov model is able to be computed $31,600$ times in substitute of the neural network. Therefore, it can be seen that the model introduced in the body of this work is better suited for applicability in models that require a prediction to be generated in real-time.

However, there are some caveats to the work that have been proposed in this thesis. For example, the movement of an object has been calculated in the pixel difference between the last recorded and the current occupied location by an object. This difference has further been conditionally adjusted to a bipolar/binary integer, ± 1. Although this method works well for the body of this work and the various datasets that have been used, the applicability of this method would potentially not be suitable for applications that would prefer to take into account at which the speed the object is travelling. Therefore, there are improvements and amendments that could be made to the work to ensure that predictions can take into account the speed at which an object is moving along with the angular change in direction. Further amendments and improvements to this project have been discussed in Section 5.1.

5.1 Further Improvements to the Study

This section will discuss the various improvements that can be made to this study to increase the prediction accuracy of the stochastic models and neural networks, alongside the different methodologies that can be imposted to track the movement of an object. The various improvements can range from the alternative methods of tracking an object as it moves through its scene by measuring its velocity and rotation, or the prospect of finding a new detection method to ensure that the predictions can be made in a real-time constraint for higher-resolution files such as 720p or Ultra High Definition (UHD) video footage.

5.1.1 Object Recognition and Tracking

The recognition and tracking of an object with the computer-generated videos uses a hybrid of the SURF and BRISK algorithms. It can be seen this method provides a reliable recognition rate and the desired frame-rate of 30 frames-per-second can be achieved on a 540p resolution. However, with the popularity of cameras within mobile devices and vehicles able to shoot footage with a resolution up to 4K or UHD (4096x3160). The prospects of analysing video data at a 540p resolution is pretty redundant and a new method of being able to extract features from videos with a ultra high-definition (UHD) resolution needs to be sought in order to progress with the advancement of technology. It has been proposed in this study that downscaling the recorded

video to a resolution of 540p and then up-scaling the boundaries to be shown on the UHD frame can be implemented. However, by resizing the video to a 540p scale, it could discard some of the key data that may be useful in the UHD image and therefore is not a very practical solution.

Further improvements can be made to the tracking of an object whilst it is moving through the frames of a video The chosen method for this study is the pixel displacement which is then simplified to a binary/bipolar value. However, it can be seen that this type of method does not provide enough information, especially when considering the movement of a pedestrian. The simplification of a pedestrians movement to one of eight compass directions could hinder the the information such as the angle of movement and the velocity of movement that is exhibited by the pedestrian. Although the pixel displacement process can be used to determine the speed at which an object is moving; the values are discarded and limited to a difference of one or zero; therefore no indication of speed is present. Using this information could influence the accuracy of the predictions; they could improve or hinder performance depending upon the complexity of the movement. Finally, the calculation of the objects movement can be influence based upon the placement of the camera and the distance at which the object is located from the object. For example; an object that is closer towards the lens of a camera then the difference in movement between locations **A** and **B** will be smaller, than it would have been if the pedestrian is further in the background. Therefore, a process of normalising this type of difference would be required.

5.1.2 Stochastic Model

The stochastic model is not perfect, and improvements can be made to the model to increase the performance and also the tracking of a pedestrian or object through its scene. The first method of improving the model would be the consolidation of the process that is used to update the matrices when a transition occurs and the prediction method. By consolidating these two methods within one function (and albeit one search of a matrix) it could provide a reduction in the amount of time it takes to make a prediction using the dynamic stochastic model. The processing time of the traditional model is relatively low due to the searching of the matrix to find an appropriate row for generating a prediction only occurring once as the probabilities have been pre-adjusted at the beginning of the process; and is not updated for each transition seen. Therefore by applying this method to the dynamic stochastic model could provide a reduction within the amount of time it takes to generate a prediction; especially when considering the higher-order models whereby a large portion of the time is spent searching.

A further improvement could be made to the calculation of an objects movement that is used to update the stochastic model. In this work, the pixel difference is computed between two locations and then a conditional threshold is applied to a binary/bipolar value of ± 1. There are various other solutions that can be utilised to measure the movement of an object, such as the velocity of movement which would in turn also consider the angular difference of an object's movement. Using these metrics could increase the complexity of the overall model, but with the previous suggested improvement it could be seen that a decrease in computation processing is achievable.

5.1.3 Neural Network

The neural network was implemented using a pre-built Python module; and it can be see that the outcome of the results and processing times were significantly worse off in comparison to the stochastic model. The stochastic model is far simpler in complexity of structure in comparison to a neural network that functions with multiple neurons and layers opposed to a handful of states that are used in the stochastic model. However, with that being said the number of states can grow exponentially with a stochastic model when increasing the order of the model and taking into consideration a number of previous movements. To reduce the computation time of the neural network it could be implemented by hand (similar to the stochastic model) and this would

ensure that full control can be taken over the processes of a neural network, such as updating the weightings or the feedback mechanism of the recurrent neural network.

5.2 Future Work and Expansion of the Study

The foundations that have been laid by this study is the ability to be able to make predictions that are highly accurate using a variant of a traditional stochastic model and adjusting the probability values dynamically by counting the number of transitions that occur between states. There are several extensions that can be implemented to this underlying work:

- **hybrid modelling**
- **future location prediction**
- **usability for different datasets**

5.2.1 Hybrid Stochastic Modelling

The amalgamation of two feature detectors has been discussed in Chapter 3 and it can be seen that the combined use of the two algorithms can provide a stronger recognition of the object from its environment instead of using a single detector. Therefore, it is proposed that the same methodology should be applied to the traditional and dynamic stochastic models; otherwise known as a hybrid model. The proposed methodology of the hybrid model would follow the introduced dynamic stochastic model by counting the transitions that occur between states; and when the probabilities of the model are recalculated a scoring function is applied; the same process used for the pattern analysis of the pedestrian matrices. An activity diagram for the outline of the proposed method is shown in Figure 5.1.

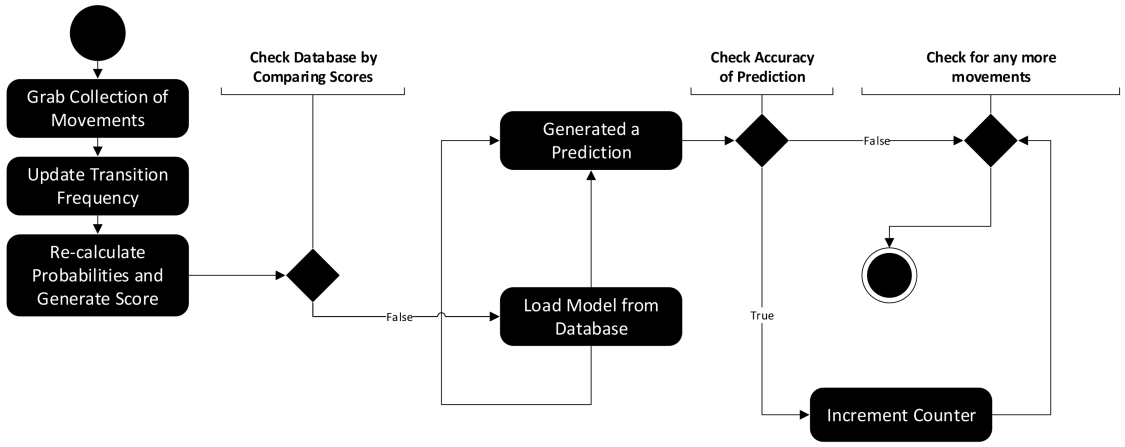


Figure 5.1: The proposed future work of using a hybrid stochastic model to generate predictions.

As it can be seen, the proposed method follows the beginning stages of the dynamic stochastic model but once the probabilities of the model have been recalculated it is proposed that a score is generated and a check is made against a database to load a pre-defined model that has a similar score but with model that is more accurate for predicting the types of movements that have already been observed.

5.2.2 Future Location Prediction

The purpose of this study was to determine a method that could be used to generate a prediction for an objects intended next direction of movement. The type of model that has been developed is able to accurately determine the direction of movement for an object and return a prediction with a model accuracy rating over 80%. The model could be used to determine the future location of an object or pedestrian based upon the last direction of movement that has been observed and could be most useful for when the object is occluded (hidden from view). This could be useful if the pedestrian was walking towards the vicinity where suddenly the camera is obscured by a wall or tree. The algorithm could continue to predict that the pedestrian was going in x direction and continue tracking as they move behind the wall and come back within the vicinity of the camera (or another camera). However, there is the chance that with this method the pedestrian could suddenly change direction and fool the algorithm and therefore the expected location of the pedestrian would become false and the pedestrian has been 'lost'. If the pedestrian becomes lost then it is proposed that the model returns to a state of equal probability whereby a prediction is made in any of the eight directions until the pedestrian has been found again.

5.2.3 Applicability to Different Data Sources

The experiments within this study has been applied to a text file of pedestrians moving within a train station, and the prediction of an object moving within a computer-generated video. The study shows that the model has a high accuracy for prediction on both datasets, with the accuracy getting better for the various orders that have been experimented with. To determine whether the same results could be achieved on a different dataset, then future work could look at the use of different use-cases to determine how the dynamic stochastic model works. For example, a dataset consisting of animals that are roaming freely within a field could be introduced; due to the sporadic and sometimes limited movements the animals would exhibit. Another study could look at the prediction of the football from clips of football matches to determine whether a prediction could be made on who will score the next goal based upon the placement of the football within the scene.

Appendix A

Thesis Results

The results from the experiments are too large to be able to include within the appendix of the thesis. Therefore, the results have been shared via the GitHub repository web-site for easy access and sharing.

A.1 Yi et al.'s Pedestrian Dataset

The collection of files are stored in a CSV file-format and consist of the results from the stochastic models and neural networks that have been used in this study.

URL: https://bit.ly/2M5R4Wx

A.2 Thesis Video Dataset

The collection of files are stored in a CSV file-format and consist of the results from the stochastic models that have been used in this study.

URL: https://bit.ly/2O38W5k

Appendix B

Feature Detection: Source Code

The full collection of source-code can be found within three repositories on the GitHub website. Each repository is for an experiment that has been performed within the thesis and can be accessed by e-mailing the author: cornelii@uni.coventry.ac.uk

Access will be provided upon receipt of the e-mail.

B.1 Linux Experiment

The experiment performed on the Linux operating system for detecting an object from its scene using a mixture of images and videos.

URL: https://bit.ly/2w3Rjvo

B.2 Android Experiment

The experiment performed on the Android operating system using a mixture of images for detecting an object from its environment.

URL: https://bit.ly/2MKXLB0

Appendix C

Thesis Study: Source Code

The full collection of the source code used for the study within the thesis for the prediction of an objects movement is split within two repositories. Each repository is for an experiment that has been performed within the thesis and can be accessed by e-mailing the author: cornelii@uni.coventry.ac.uk

Access will be provided upon receipt of the e-mail.

C.1 Pedestrian Dataset

The source-code for the experiment that was performed upon the dataset that was sourced from Yi et al. (2016*b*).

URL: https://bit.ly/2nZfVB2

C.2 Video Dataset

The source-code for the experiment that was performed upon the dataset that was newly created by the author to determine whether predictions can be made in real-time using high-resolution video files.

URL: https://bit.ly/2wkT6vg

Bibliography

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L. & Savarese, S. (2016), Social lstm: Human trajectory prediction in crowded spaces, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 961–971.

Awad, A. I. & Hassaballah, M. (2016), *Image Feature Detectors and Descriptors: Foundations and Applications*, Vol. 630, Springer.

Bay, H., Ess, A., Tuytelaars, T. & Van Gool, L. (2008), 'Speeded-up robust features (surf)', *Computer vision and image understanding* **110**(3), 346–359.

Bear, M., Connors, B. & Paradiso, M. (2007), *Neuroscience:*, Neuroscience: Exploring the Brain, Lippincott Williams & Wilkins.
URL: *https://books.google.co.uk/books?id=75NgwLzueikC*

Bekele, D., Teutsch, M. & Schuchert, T. (2013), Evaluation of binary keypoint descriptors, *in* 'Image Processing (ICIP), 2013 20th IEEE International Conference on', IEEE, pp. 3652–3656.

Bera, A., Kim, S., Randhavane, T., Pratapa, S. & Manocha, D. (2016), Glmp-realtime pedestrian path prediction using global and local movement patterns, *in* 'Robotics and Automation (ICRA), 2016 IEEE International Conference on', IEEE, pp. 5528–5535.

Bradski, G. (2000), *Dr. Dobb's Journal of Software Tools* .

Cancela, B., Iglesias, A., Ortega, M. & Penedo, M. G. (2014), Unsupervised trajectory modelling using temporal information via minimal paths, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 2553–2560.

Chen, T.-W., Chen, Y.-L. & Chien, S.-Y. (2008), Fast image segmentation based on k-means clustering with histograms in hsv color space, *in* 'Multimedia Signal Processing, 2008 IEEE 10th Workshop on', IEEE, pp. 322–325.

Chen, T. Y.-H., Ravindranath, L., Deng, S., Bahl, P. & Balakrishnan, H. (2015), Glimpse: Continuous, real-time object recognition on mobile devices, *in* 'Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems', ACM, pp. 155–168.

Coltin, B., Fusco, J., Moratto, Z., Alexandrov, O. & Nakamura, R. (2016), Localization from visual landmarks on a free-flying robot, *in* 'Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on', IEEE, pp. 4377–4382.

Cornelius, I. (2014), 'Evaluation of sift on android-based mobile devices and personal computers'. Thesis (MScRes) Coventry University; ID: COVALMA2164249370002011.

Crow, F. C. (1984), 'Summed-area tables for texture mapping', *SIGGRAPH Comput. Graph.* **18**(3), 207–212.
URL: *http://doi.acm.org/10.1145/964965.808600*

DiCarlo, J. J., Zoccolan, D. & Rust, N. C. (2012), 'How does the brain solve visual object recognition?', *Neuron* **73**(3), 415–434.

Fabre-Thorpe, M., Richard, G. & Thorpe, S. J. (1998), 'Rapid categorization of natural images by rhesus monkeys', *Neuroreport* **9**(2), 303–308.

Ghahramani, Z. (2001), 'An introduction to hidden markov models and bayesian networks', *International journal of pattern recognition and artificial intelligence* **15**(01), 9–42.

Haralick, R. M. (1979), 'Statistical and structural approaches to texture', *Proceedings of the IEEE* **67**(5), 786–804.

Hariyono, J., Hoang, V.-D. & Jo, K.-H. (2014), 'Moving object localization using optical flow for pedestrian detection from a moving vehicle', *The Scientific World Journal* **2014**.

Helbing, D. & Molnar, P. (1995), 'Social force model for pedestrian dynamics', *Physical review E* **51**(5), 4282.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Jain, R., Kasturi, R. & Schunck, B. (1995), *Machine Vision*, Computer science series, McGraw-Hill.

Kahn, J. (2016), 'Bmw will finally launch support for apple's carplay later this year'.
URL: *https://9to5mac.com/2016/05/20/bmw-apple-carplay-x5-x6-m/*

Karpushin, M., Valenzise, G. & Dufaux, F. (2015), Improving distinctiveness of brisk features using depth maps, *in* 'Image Processing (ICIP), 2015 IEEE International Conference on', IEEE, pp. 2399–2403.

Kim, S., Guy, S. J., Liu, W., Wilkie, D., Lau, R. W., Lin, M. C. & Manocha, D. (2015), 'Brvo: Predicting pedestrian trajectories using velocity-space reasoning', *The International Journal of Robotics Research* **34**(2), 201–217.

Koppula, H. S., Gupta, R. & Saxena, A. (2013), 'Learning human activities and object affordances from rgb-d videos', *The International Journal of Robotics Research* **32**(8), 951–970.

Krumm, J. (2008), A markov model for driver turn prediction, Technical report, SAE Technical Paper.

Krumm, J., Horvitz, E. & Letchner, J. (2007), Map matching with travel time constraints, Technical report, SAE Technical Paper.

Leutenegger, S., Chli, M. & Siegwart, Y. (2011), Brisk: Binary robust invariant scalable keypoints, *in* 'In Computer Vision (ICCV), 2011 IEEE International Conference on', pp. 2548–2555.

Lowe, D. G. (1999), Object recognition from local scale-invariant features, *in* 'Computer vision, 1999. The proceedings of the seventh IEEE international conference on', Vol. 2, Ieee, pp. 1150–1157.

Lowe, D. G. (2004), 'Distinctive image features from scale-invariant keypoints', *International journal of computer vision* **60**(2), 91–110.

Markov, A. A. (1906), 'Extension of the law of large numbers to dependent quantities', *Izv. Fiz.-Matem. Obsch. Kazan Univ.(2nd Ser)* **15**, 135–156.

Meyn, S. & Tweedie, R. L. (2009), *Markov Chains and Stochastic Stability*, 2nd edn, Cambridge University Press, New York, NY, USA.

Microsoft.com (2006), 'Blue&Me: Endless Infotainment Possibilities While on the Road'.
URL: *https://news.microsoft.com/2006/02/28/blueme-endless-infotainment-possibilities-while-on-the-road/*

Mikolajczyk, K. & Schmid, C. (2001), Indexing based on scale invariant interest points, *in* 'Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on', Vol. 1, IEEE, pp. 525–531.

Mikolajczyk, K. & Schmid, C. (2004), 'Scale & affine invariant interest point detectors', *International journal of computer vision* **60**(1), 63–86.

Mikolajczyk, K. & Schmid, C. (2005), 'A performance evaluation of local descriptors', *IEEE transactions on pattern analysis and machine intelligence* **27**(10), 1615–1630.

Miksik, O. & Mikolajczyk, K. (2012), Evaluation of local detectors and descriptors for fast feature matching, *in* 'Pattern Recognition (ICPR), 2012 21st International Conference on', IEEE, pp. 2681–2684.

Nižetić, I. & Fertalj, K. (2010), 'Automation of the moving objects movement prediction process independent of the application area', *Computer Science and Information Systems* **7**(4), 931–945.

Nižetić, I., Fertalj, K. & Kalpic, D. (2009), A prototype for the short-term prediction of moving object's movement using markov chains, *in* 'Information Technology Interfaces, 2009. ITI'09. Proceedings of the ITI 2009 31st International Conference on', IEEE, pp. 559–564.

Pearson, K. (1896), 'Mathematical contributions to the theory of evolution. iii. regression, heredity, and panmixia', *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character* **187**, 253–318.

Pellegrini, S., Ess, A., Schindler, K. & Van Gool, L. (2009), You'll never walk alone: Modeling social behavior for multi-target tracking, *in* 'Computer Vision, 2009 IEEE 12th International Conference on', IEEE, pp. 261–268.

Ross, S. (1996), *Stochastic processes*, Wiley series in probability and statistics: Probability and statistics, Wiley.

Rosten, E. & Drummond, T. (2005), Fusing points and lines for high performance tracking, *in* 'Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1', Vol. 2, IEEE, pp. 1508–1515.

Rosten, E. & Drummond, T. (2006), Machine learning for high-speed corner detection, *in* 'European conference on computer vision', Springer, pp. 430–443.

Shao, J., Kang, K., Change Loy, C. & Wang, X. (2015), Deeply learned attributes for crowded scene understanding, *in* 'The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Thorpe, S., Fize, D. & Marlot, C. (1996), 'Speed of processing in the human visual system', *nature* **381**(6582), 520.

Tomasi, C. & Kanade, T. (1991), 'Detection and tracking of point features'.

Van Den Berg, J., Guy, S. J., Lin, M. & Manocha, D. (2011), Reciprocal n-body collision avoidance, *in* 'Robotics research', Springer, pp. 3–19.

Viola, P. & Jones, M. (2001), Rapid object detection using a boosted cascade of simple features, *in* 'Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on', Vol. 1, IEEE, pp. I–I.

Wakim, C. F., Capperon, S. & Oksman, J. (2004), A markovian model of pedestrian behavior, *in* 'Systems, Man and Cybernetics, 2004 IEEE International Conference on', Vol. 4, IEEE, pp. 4028–4033.

Xia, J. C., Zeephongsekul, P. & Packer, D. (2011), 'Spatial and temporal modelling of tourist movements using semi-markov processes', *Tourism Management* **32**(4), 844–851.

Yi, S., Li, H. & Wang, X. (2016*a*), Pedestrian behavior understanding and prediction with deep neural networks, *in* 'European Conference on Computer Vision', Springer, pp. 263–279.

Yi, S., Li, H. & Wang, X. (2016*b*), Pedestrian behavior understanding and prediction with deep neural networks, *in* 'European Conference on Computer Vision', Springer, pp. 263–279.

Zou, J., Han, Y. & So, S.-S. (2008), 'Overview of artificial neural networks', **458**.