EDINBURGH NAPIER UNIVERSITY

Tracking, Analysis and Measurement of Pedestrian Trajectories.

by

Sarah Elisabeth Clayton

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Engineering, Computing and Creative Industries
School of Computing

May 2016

Declaration of Authorship

I, Sarah Elisabeth Clayton, declare that this thesis titled, "Tracking, Analysis and Measurement of Pedestrian Trajectories" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ____________________________

Date: ____________________________

"I can calculate the motion of heavenly bodies, but not the madness of people."

Isaac Newton

EDINBURGH NAPIER UNIVERSITY

Abstract

Faculty of Engineering, Computing and Creative Industries
School of Computing

Doctor of Philosophy

by Sarah Elisabeth Clayton

Pedestrian movement is unconstrained. For this reason it is not amenable to mathematical modelling in the same way as road traffic. Individual pedestrians are notoriously difficult to monitor at a microscopic level. This has led to a lack of primary data that can be used to develop reliable models.

Although video surveillance is cheap to install and operate, video data is extremely expensive to process for this purpose. An alternative approach is to use passive infrared detectors that are able to track individuals unobtrusively. This thesis describes the use of a low cost infrared sensor for use in tracking pedestrians. The sensor itself, manufactured by a British company, is designed to count people crossing an arbitrary datum line. However, with the development of additional software, the functionality of these sensors can be extended beyond their original design specification. This allows the trajectories of individual pedestrians to be tracked.

Although the field of view of each sensor is relatively small (4×4 m), five were deployed in a busy indoor corridor, covering most of its length. In this research, the technical challenges involved in using the sensors in this way are addressed. Statistics derived from the data collected are then compared to other studies at this scale.

Acknowledgements

I would like to acknowledge the help, support and patience of my supervisors, Dr. Neil Urquhart and Prof. Jon Kerridge. I would like to thank the technicians at the School of Computing. Riccardo Lazzarini and Felix Jarvis, for their ongoing help in installing and maintaining the Irisys sensors used during this research, as well as Bryden Ritchie for his help in measuring the corridor under observation.

Particular thanks go to Martyn Chapman for the loan of his LEGO® train set, which was invaluable.

I would especially like to thank my family for their unstinting support and encouragement, particularly my parents, but also my little sister, Dr. Sophie Clayton, for her occasional hectoring.

Contents

List of Figures

List of Tables

List of Equations

Dedicated to

Dr. Karin Van West
for her love and support

Dr. Sophie Clayton
for showing me how it's done

Dr. Pat Clayton
for the dumb idea in the first place!

In memory of

My grandparents
Bertha Severn Frank Severn
Ann Clayton Hubert Clayton

Charlie
(1st June 1996-4th June 2013)

He who binds to himself a joy
Does the winged life destroy;
But he who kisses the joy as it flies
Lives in eternity's sun rise.
– William Blake.

Sam
for the long walks on the beach

Chapter 1

Introduction

1.1 Introduction

The study of pedestrian dynamics is concerned both with pedestrians in normal situations and in evacuation situations. While the latter has always been subject to intense study, governments across the world are now placing pedestrians at the heart of their transport policies. The aim is not only to facilitate pedestrian movement at transport interfaces, such as rail stations, but also to encourage this as a mode of transport in its own right. An example is the development of the London Pedestrian Model. This is the result of collaboration between Transport for London (TfL) and the Central London Partnership (CLP), who commissioned the consultancy, Intelligent Space Partnership (ISP), "to develop a model of total walking volumes for every node on the street network in Central London" [1]. It arose from the Mayor's Transport Strategy [2], where a key aim was "to make London one of the most walking friendly cities for pedestrians by 2015."

Travel on foot is ubiquitous, and encompasses all human activities. However, our understanding of this form of transport has not advanced much since the work of Fruin [3] on levels of service in the 1970s. The reasons for this are two fold. Pedestrian movement is unconstrained, unlike road traffic, with a great deal of variability

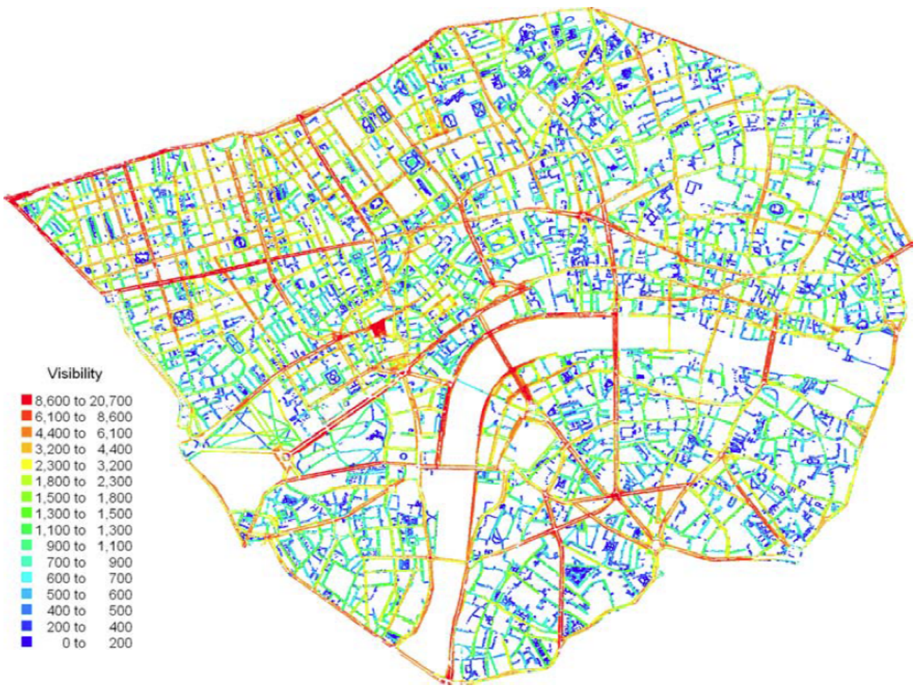


FIGURE 1.1: Visibility analysis for Central London. Taken from [1].

in acceleration and direction. This is very difficult to model mathematically. The second difficulty is in being able to gather data about people's behaviour as they travel through a space. Although a few studies have published models based on such data, the data sets used are small.

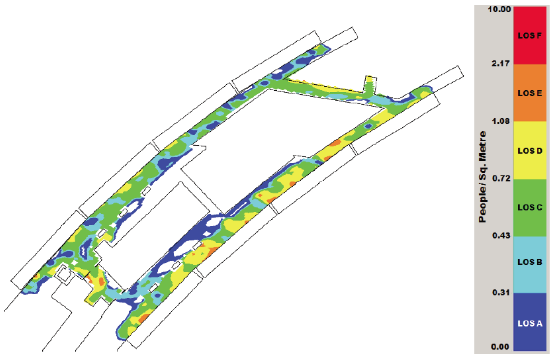
These difficulties are, however, increasingly being overcome due to the decreasing cost of computing resources and the emergence of new technologies allowing the gathering of primary data. These technologies range from the ubiquitous, such as video surveillance, to extremely sophisticated and expensive laser based scanners.

1.2 Motivation

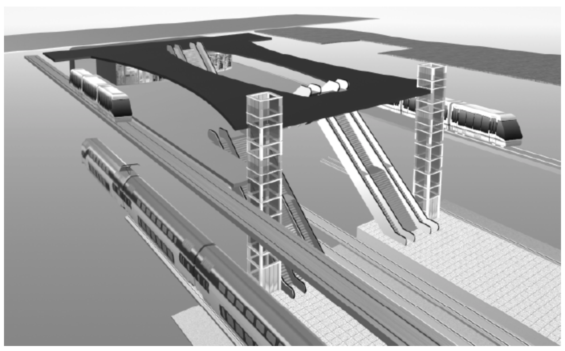
Pedestrian studies are divided into three categories in scale: macroscopic, mesoscopic and microscopic. Macroscopic models are concerned with aggregate behaviours, and predictors for these. In the case of the London Pedestrian Model, mentioned above, metrics such as pavement width, proximity to London Underground stations and visibility analysis, as shown in Figure 1.1, across the whole area of study are used to predict pedestrian behaviour and densities. Studies are usually not focused on such a large area. Transport hubs, such as train stations and airports, are usually considered.

The microscopic level is at the other end of the spectrum, in terms of scale, and is concerned with behaviour of the individual. Increasingly simulations model the behaviour of individual pedestrians at the microscopic level, to see what effects crowds have at the macroscopic level. An example is shown in Figure 1.2. Real crowds lead to emergent phenomena such as lane formation, at the macroscopic level, which is an area of study in its own right.

At the microscopic level, Hoogendoorn and Bovy [4] make three further distinctions: the strategic, tactical and operational levels at which individuals make choices. Broadly speaking, the strategic level concerns activity choice and choice of destination, the tactical level concerns route choice, and the operational level concerns walking behaviour. As they state: “In this hierarchy, expected utilities at lower levels influence choices at higher levels. Choices at higher levels condition choice sets at lower levels.” [4]



(a) Simulated cumulative mean density on the train platforms of a British suburban train station, using the STEPS simulation tool. Taken from [5]. The legend on the right shows levels of services (number of people per square metre).



(b) Overview of Delft station, used in the SimPed simulation tool. Taken from [6]

Figure 1.2: Example of train stations being studied.

In this research, the operational level is considered. Unhindered walking and pedestrian interactions in uncrowded situations have mostly not been addressed to date. As Versluis [7] and Daamen et al. [8] point out, this has consequences for the accuracy and validity of simulation models at these levels (microscopic and operational).

In addition to pedestrian dynamics research, there are a number of further applications for pedestrian data. The detection of anomalous behaviour using video surveillance is a well established area of research. Vishwakarma et al. provide a comprehensive review in [9]. Another, emerging, area of research is in robotics. In [10], Guy explains how robot navigation and collision avoidance can be developed using data from human pedestrian interactions.

1.3 Research questions

The overall research question here is: can meaningful pedestrian data be collected in an automated and unattended way? This leads on to three separate areas: the sensor hardware used and its capabilities, how the data is processed, and can the statistics derived be compared with other studies.

A number of scenarios were specified. These were to collect data on:

1. Single pedestrians;
2. Pairs of pedestrians;
3. Pedestrians crossing each others' path;
4. One pedestrian following another;
5. Groups (three or more) of pedestrians and their interactions with others;
6. Crowds passing through the corridor.

FIGURE 1.3: Irisys MkII sensor.

Given the long association with the manufacturer and a heavy investment in the Irisys products, the hardware used is the Irisys MkII People Counter, shown in Figure 1.3. Their ability to collect data in this way was untested.

If data could be collected reliably, the problem then was one of accurate and efficient storage. This is a problem that has come to the fore relatively recently in the field of geographic information systems (GIS) with the widening of access to global navigation satellite system (GNSS) and mobile phone tracking data. The data sets that are considered in recent research in GIS are far larger, and over much greater scales, than in this research. Solutions that are adapted to their purposes generally do not have the accuracy required for this application.

A consequence of using a commercial product that is designed to respect the privacy of the pedestrians under observation is that all demographic data is lost. At the same time, data protection laws give identifiable individuals certain rights, which are enforced by the Information Commissioner's Office (ICO), under the terms of the Data Protection Act. Working outside the laboratory can create ethical problems. These are entirely avoided by not collecting demographic data. In the framework of a much larger research study, such as the one described by Willis et al. in [11], the necessary resources would be available to handle the technical as well as ethical issues involved.

This leads to the final research question: if the first two challenges are overcome, are the statistics derived from the data collected consistent with literature?

In summary, the research questions are as follows:

1. Can an automated and unattended pedestrian data collection system be created using the Irisys MkII sensors? If so, which of the scenarios mentioned above can they provide data for?
2. Can the data returned be processed in an accurate and efficient way?

   In order for the data to be meaningful, the following need to be recorded:

   (a) the x, y, t coordinates of the pedestrian;

   (b) their speed;

   (c) the origin and destination of their trajectory;

   (d) the length of their trajectory;

   (e) which of the scenarios listed above is relevant.
3. Can the statistics derived from the data be validated against other studies?

The danger in implementation based research is however two fold. The first is that the system fails, leading to few insights into the motivating purposes of the system. The second is that it is often difficult to generalise from a specific system to generic principles. This is addressed in the concluding chapter, Chapter 8.

1.4 Structure of thesis

The structure of this thesis is as follows. In Chapter 2, a literature review of tracking technologies, data processing methods as well as data analyses is provided. This provides an introduction to the concepts used in the following chapters. The thesis is then in three parts: the implementation of the tracking systems, challenges and methods for data processing, followed by an analysis of the data.

Chapters 3 and 4 are concerned with the implementation of the tracking system. In Chapter 3, infra-red sensor technology is discussed in detail, leading onto a description of the Irisys MkII sensors themselves and their capabilities. In Chapter 4, the location where the sensors were deployed is described, as well as how the sensors were deployed and an overall infrastructure was implemented.

Chapters 5 and 6 deal with the issues faced in processing the data collected. Many of these challenges are specific to an Irisys MkII based tracking system. However, many are not and are generic to data returned by many tracking systems, examples of which are discussed in Chapter 2, the Literature Review. The problems faced in this relatively small-scale study are two fold: data reduction and data queries. First data reduction was achieved using B-splines (or basis splines). In Chapter 5, the use of splines for similar purposes are discussed, the mathematical bases of B-splines are explained, and their use in this thesis is described. In Chapter 6, the processing of the data, from raw tracking data to database queries, is explained.

In Chapter 7, the processed data is analysed statistically according to a number of scenarios and compared with the results of experimental studies on similar scenarios.

Finally, Chapter 8 provides conclusions, recommendations and suggestions for further work. Here the relative success or failure of the efforts described in the previous chapters is assessed.

Chapter 2

Literature Review

2.1 Introduction

The purpose of this chapter is to present a review of the literature in the subject areas covered by this research. These are indoor tracking and the derivation of statistics derived from this data.

The PERMEATE project at Edinburgh Napier University, detailed in [12, 13], presented the first efforts to evaluate the use of Irisys sensors in pedestrian dynamics research. In [13], a comparison is made between video technology available at the time and the capabilities of the Irisys sensors. Kerridge et al. conclude that given the identified weaknesses of both technologies, the use of both provides the best compromise.

In [12], the results of the PERMEATE corridor experiments (see Figure 2.1), using Irisys sensors are reported in detail. In these experiments, groups were sent down a narrow corridor, either in the same direction or in contraflow. An obstacle was placed in the corridor to see the effect on pedestrian movements. The obstacle is shown as a yellow "Caution Slippery" sign placed in the middle of the corridor in Figure 2.1. In evaluating the Irisys sensors, it was found that matching trajectories between the two sensors used was problematic. However, the authors were more confident about the ability to derive statistics from the sensors.

Since this study was carried out, many new technologies have become available, which are examined in this chapter.

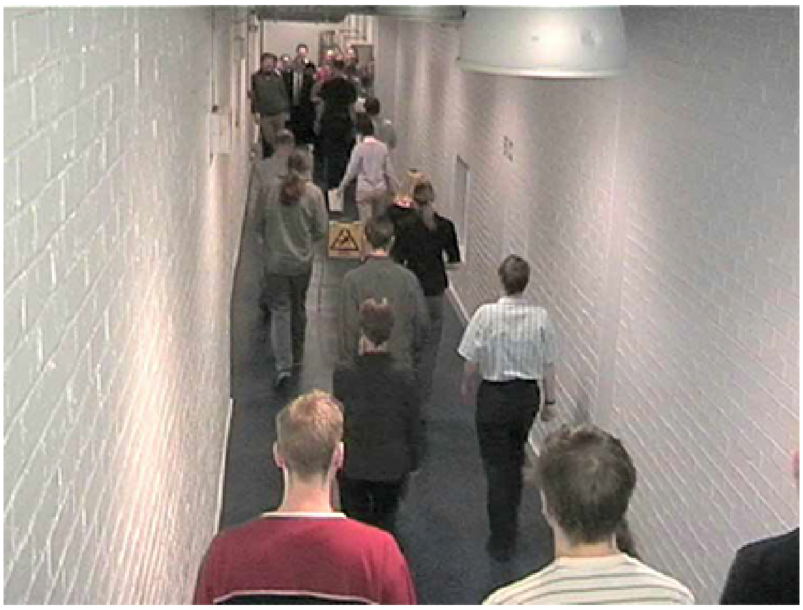


FIGURE 2.1: People avoiding an obstacle during the PERMEATE study. Taken from [12].

The chapter is organised as follows. In Section 2.2 indoor position systems are surveyed. This overview also addresses emerging areas of research in the field, in particular inertial positioning systems, as well as more traditional tracking technologies such as video. In Section 2.3 literature on pedestrian interactions is surveyed.

2.2 Indoor positioning systems

Traditionally, pedestrians have been tracked using early ubiquitous technologies such as video. As sensor components have become more and more miniaturised, and hence ubiquitous too, the number of means of tracking pedestrians has increased. Smartphones are a particular example. These often contain accelerometers, gyroscopes, magnetometers and GNSS receivers.

However, many of these have limited accuracy indoors. GNSS have limited coverage inside buildings. Magnetometers, that provide compass bearings, also suffer from severe errors inside buildings, due to buildings' structural components. Therefore, getting a pedestrian's absolute position is more challenging indoors than in an outdoor environment. This problem has led to a great deal of research interest. In part this is because of the utility of a reliable system to emergency services and the military, as well as the commercial possibilities inherent in Location Based Services.

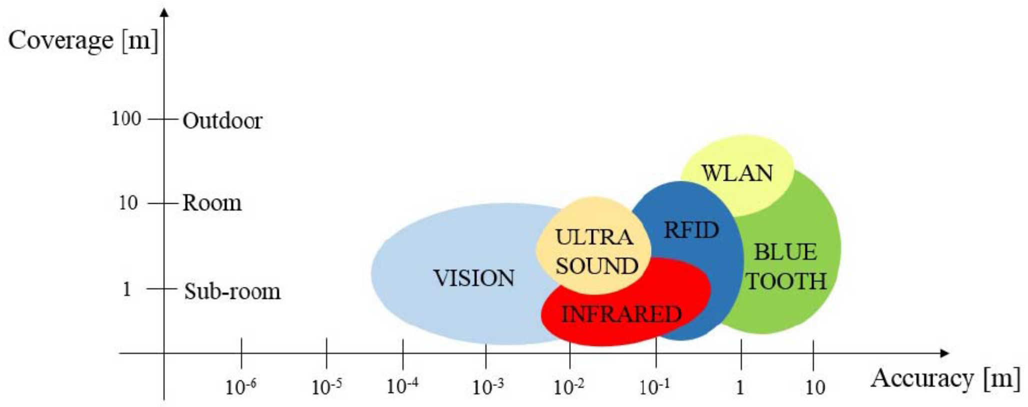


Figure 2.2: Graphical overview of technologies enabling indoor localization. Taken from [14].

Pedestrian tracking technologies can be broadly divided into those that require an underlying infrastructure (such as closed circuit TV, wireless local area network, Bluetooth and so forth) and those that do not. In practice, many hybrid systems exist that leverage the benefits of a fixed infrastructure used in conjunction with technology worn or carried by pedestrians, such as smartphones, and compensate for the shortcomings of each.

In [14], Mainetti et al. review available infrastructure dependent technologies. This is summarised in Figure 2.2. They list six technologies and compare them according to accuracy and coverage. This list is not exhaustive. In [15] focuses on pedestrian dead reckoning (PDR). In practice all approaches, other than the most expensive video systems, are hybrid and rely both on a fixed infrastructure as much as worn or carried devices.

In order to cover as many of these technologies as possible, this section is organised in three subsections. First, video technology is address in Section 2.2.1. Wireless network aided positioning systems are then examined in Section 2.2.2. Finally, inertial positioning systems are considered in Section 2.2.3.

2.2.1 Video

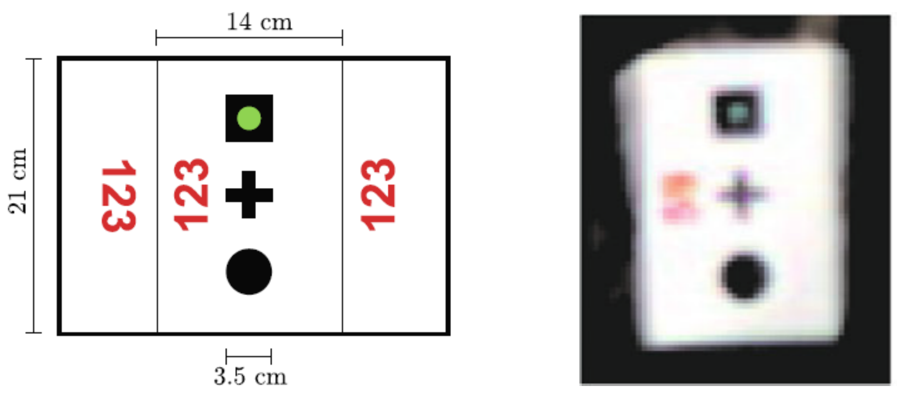
In pedestrian dynamics, the most used and cited system is `PeTrack`, described in [16]. In [17], this is described as a highly accurate video system that is able to deal with a large number of participants in laboratory experiments. In order to achieve its high degree of accuracy, a video camera is placed overhead, and participants are required to place a marker on their heads, that is shown in Figure 2.3(a).

This software has been used in many pedestrian dynamics laboratory experiments. Examples of this are given in [18, 19]. An example of the output of the system is shown in Figure 2.3. This demonstrates the level of accuracy and the number of participants in laboratory experiments the system is able to handle. In Figure 2.3(b), the participants can be seen wearing imprints similar to the one shown in Figure 2.3(a). In Figure 2.3(c), the collected trajectories, processed by the `PeTrack` software are shown.

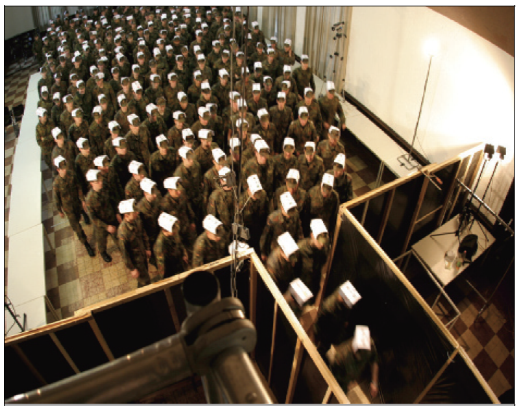
Rupprecht et al. [19] study the behaviour of human crowds in corridors and in bottleneck situations. Similar work was done previously by Winkens et al. in [18], with the participation of 250 soldiers from the Bergische Kaserne Düsseldorf.

While `PeTrack` is particularly adapted to studying pedestrian dynamics, video surveillance is ubiquitous and pervasive. A review of various optical indoor positioning systems is given in [20]. As Mautz et al. state:

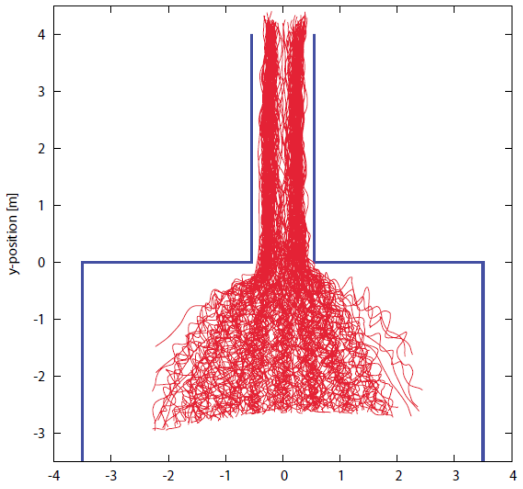
> "Optical positioning is currently becoming a dominating technique that covers a wide field of applications at all levels of accuracy, with its



(a) Marker used in `PeTrack`: the picture on the left shows the imprint, the picture on the right how this is seen by the video camera. Taken from [17]



(b) Participants in the corridor bottleneck experiment.



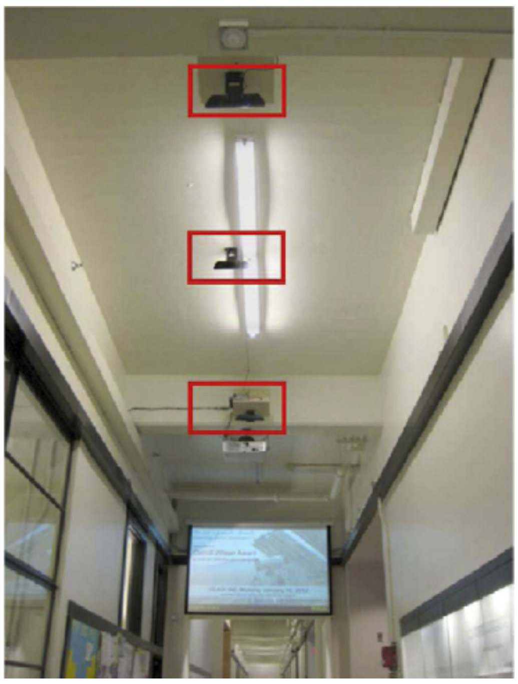
(c) Representation of the collected trajectories.

Figure 2.3: In (a) the marker used by `PeTrack` is shown [17]. Figures (b) and (c) are examples of the `PeTrack` software being used in [19]

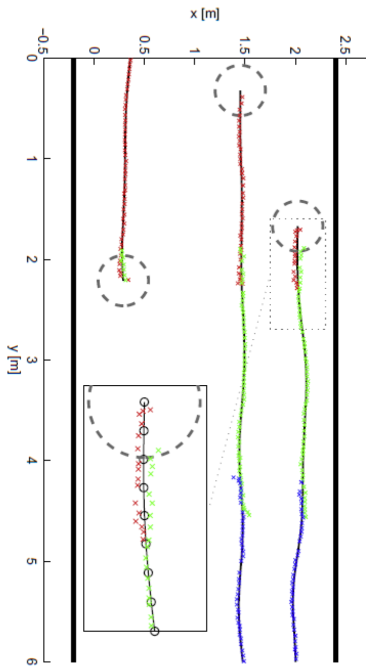
> main application area in the sub-mm domain. The success of optical methods originates from improvement and miniaturization of actuators (e.g. lasers) and particularly advancement in the technology of the detectors (e.g. CCD sensors). In parallel there has been an increase in the data transmission rates and computational capabilities as well as profound development of algorithms in image processing." [20]

A full discussion of these advances is outside the scope of this research. However, an important development in optical positioning systems is the Microsoft Kinect, available at consumer prices and with active infra-red as well as a visible light camera.

In [21], three Kinect devices were mounted on the ceiling of the Infinite Corridor at the Massachusetts Institute of Technology (MIT). The Kinect uses active infra-red capabilities for three dimensional reconstruction of scenes, a common problem with camera technology. This subject is covered in detail in [22], as well as software extensions to the pre-existing capabilities of the sensors.



(a) Ceiling mounted Microsoft Kinect sensors shown in red boxes.



(b) Trajectories captured by the Kinect sensors.

FIGURE 2.4: Microsoft Kinect sensors ceiling mounted in the Infinite Corridor at MIT, which an example of the trajectories captured. Taken from [21].

The work presented in [21] is very similar to the work presented in this research. The Kinect sensors are ceiling mounted, as shown in Figure 2.4(a). The output from the system is shown in Figure 2.4(b). The captured trajectories are represented by B-splines. The sensors have a relatively small field of view when used this way, 2×2 m. They do not investigate mounting the Kinect sensors obliquely to increase the field of view, given that the sensors are able to measure the distance objects are from them with some accuracy.

2.2.2 Wireless network aided positioning

In 1999, the US Wireless Communications and Public Safety Act mandated the creation of what was to be called the enhanced 911 or E911 service. The intention was to allow the emergency services instant access to the precise location of callers. All mobile phones were required to be compliant with the requirements of the E911 service by 2005 [23]. Much of the interest in wireless network aided positioning arose from this. While many of the problems of outdoor location services have been solved by the use of GNSS and cellular networks, indoor wireless positioning problems persist.

Wireless Local Area Networks (WLAN) or Wi-Fi is widely used for localisation. Due to the ubiquity of Wi-Fi networks, no new infrastructure need be installed. Another advantage is that Line of Sight (LoS) is not necessary. Most WLAN positioning is done using the Received Signal Strength Indicator (RSSI). According to [14], there are three basic ways of determining location using the RSS method:

- Cell of Origin (CoO) method: this works by knowing the exact location of the access point (AP) nodes connect to;
- Triangulation: the position of the node is determined by triangulating the signal strength from multiple APs;
- Fingerprint method: According to [14], this is "most viable method" and involves creating a radio map of the environment. That is sampling the signal strength of APs at various locations. The system is then able to localise nodes by comparing the recorded RSS values against those in the radio map database.

The accuracy of WLAN positioning is 20 m to 40 m. Bluetooth, on the other hand, provides an accuracy of 2 m to 3 m, although the need to run the device discovery procedure when used means that there is a latency of 10–20 s in localising a device.

Bluetooth technology is further extended by Apple's iBeacon standard, itself built on the Bluetooth Low Energy (BTLE or Bluetooth 4.0[24]) standard. These range from devices powered by coin cell batteries to those that relie on an external power supply. They behave only as beacons, transmitting a 25 byte data packet at a preset interval and signal strength. This transmission contains the unique identifier of the iBeacon and the measured signal power.

One of the functions of iBeacons is region monitoring. iBeacons send push notifications to devices entering or leaving the region they cover. This is of particular interest in retail environments, and they are deployed in all Apple Stores. Of particular interest in this context is the ranging function. This comes in two forms, state estimations, and more accurate distance estimates. State estimates are shown in Figure 2.5.

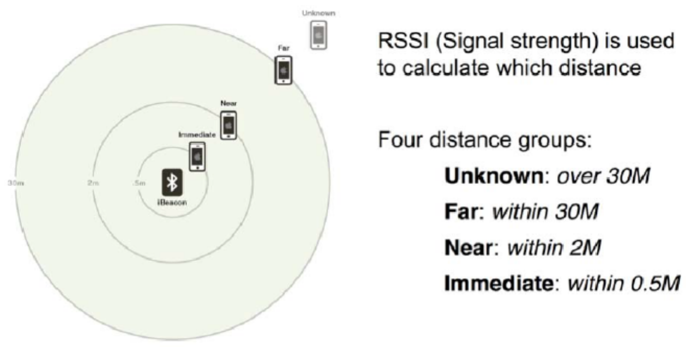


Figure 2.5: Distance calculations with iBeacon. Taken from [25].

A second ranging function is built into the iOS operating system that provides more accurate and specific distance estimates in metres. This is derived from the RSSI value.

At this time, support for iBeacon is only fully available in iOS 7.0 phones or above. There is limited support in Android 4.3 or above. Windows and Windows phones are not supported. In addition to these limitations, iBeacon suffers all the problems associated with all other wireless network positioning systems, interference from structural elements in buildings and so forth. Köhne et al. [25] recommend fusion with another source of localisation information, such as inertial navigation sensors, to increase accuracy. These are covered in Section 2.2.3.

2.2.3 Inertial positioning systems

A very extensive survey of inertial positioning systems is given by Harle in [15]. He attributes the recent interest in pedestrian dead reckoning, for localisation and navigation, to the increasing availability of the necessary sensors in smartphones. The interest arises from the possibility of tracking with accuracy similar to GNSS, but in indoor situations.

The diversity of micro electro-mechanical systems (MEMS) available for this purpose are shown in Figure 2.6. Walking movements can be measured in two ways, stance detection and step cycle detection. In stance detection, sensors are foot mounted and software detects when any given foot is planted on the floor. In step cycle detection software determines “cycles in sensor data caused by the repetitive motion of walking” [15]. As he states: “the strong periodicity in the movement coupled with the tendency of humans to sustain a consistent pace allows for a variety of constraints to be applied,” making calculations tractable. Accuracies of 99 % have been reported in laboratory conditions, although other studies have found error rates as high as 29.3 % in less controlled situations with a more diverse test population group.

Inertial navigation systems (INS) use accelerometers and gyroscopes, all available on modern smartphones, to track users. However, since these are triaxial, since there is no control over their orientation with respect to the “world frame of reference”, this leads to errors in estimates that accumulate leading to “drift.” Although the gyroscopes can estimate the orientation of the device, MEMS components are small and sources of errors themselves. Drift is usually limited by Extended Kalman Filters.

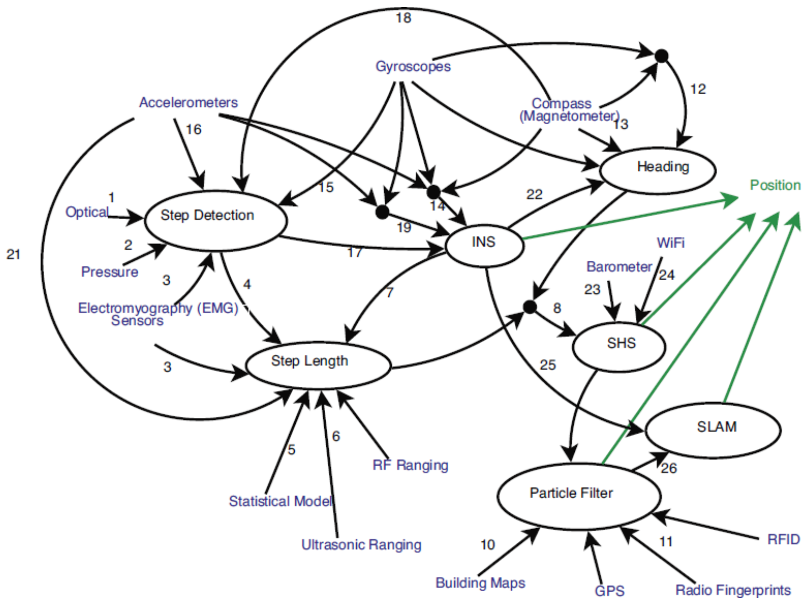


FIGURE 2.6: PDR configurations. System inputs are connected via annotated arrows to ellipses, which represent algorithms and system subunits. The arrow annotations give a numerical key to the list of literature references in [15].

The addition of magnetometer capabilities to accelerometers and gyroscopes allows the system to get an estimate of the user's absolute heading. This provides another means of correcting drift from the other sensors. They are not immune to error themselves, however, particularly in buildings. Harle [15] explains that gyroscopes and magnetometers are complementary. Gyroscopes accumulate errors in estimates of orientation over the long term, magnetometers are only subject to short-term errors. Research suggests that the fusion of outputs from the two provides robust estimates of heading changes. Given the three components mentioned above are tri-axial, they provide three dimensional estimates of motion. Step and heading systems (SHS) provide two dimensional estimates, which are sufficient in most applications. These output step and step heading polar vectors that are then summed in a two dimensional vector space to track position. The challenge is then to estimate the step length, and the heading. As stated above, the periodicity of walking is usually constant for individuals, so the step length can be dealt with as a constant. This does not handle any unusual steps well however, and dynamic step length estimation is an important area of research. Heading estimation is usually achieved using gyroscope signals.

SHS systems can be incorporated with INS systems, in order to minimise errors and drift. More recent developments have been the addition of techniques discussed above for WiFi and GSM location, either using a previously created radio map or creating one as the device is being used. Simultaneous location and mapping (SLAM), long established technique in robotics, has also been applied to the problem.

Harle [15] acknowledges the resource constraints of mobile devices and the demands of more sophisticate error reduction techniques. A particularly successful technique mentioned are particle filters, a bayesian technique, each particle representing a prior probability distribution. However, these particles often number from the tens of thousands to millions to accurately map an indoor environment. However, the same speed of innovation in mobile devices and smartphones that inspired research in this field also means that the necessary computing power will soon be available on handsets. Increasing numbers of smartphones are equipped with graphical processing units (GPU). These are designed to handle large amounts of data and run algorithms in parallel. As Harle [15] states "[w]ith particle filters in particular, there are parallelisation opportunities that might be able to exploit the imminent wave of GPU-enabled smartphones."

2.3 Pedestrian interaction research

The purpose of this research is the collection of data for study into pedestrian interactions. At this level of detail, there is very little literature available in transportation research. Most research is concerned with crowds or evacuation situations. Exceptions to this are the work of Daamen and Hoogendoorn in [26], Versluis in [7] and Daamen et al. in [8], and Ma in [27]. In [26], Daamen and Hoogendoorn investigate free speed distributions, where pedestrians are unconstrained. Table 2.2 reproduces their survey of the literature on the subject worldwide. Variations in free speeds were found on a region by region basis, with European pedestrians walking faster than those from Asia. This is shown in Table 2.1.

Region	$\bar{v}$
Europe	1.41
United States	1.35
Australia	1.44
Asia	1.24

TABLE 2.1: Average free speeds by region. Mean speed is denoted by $\bar{v}$, and is measured in $\mathrm{m\,s^{-1}}$. Taken from [26].

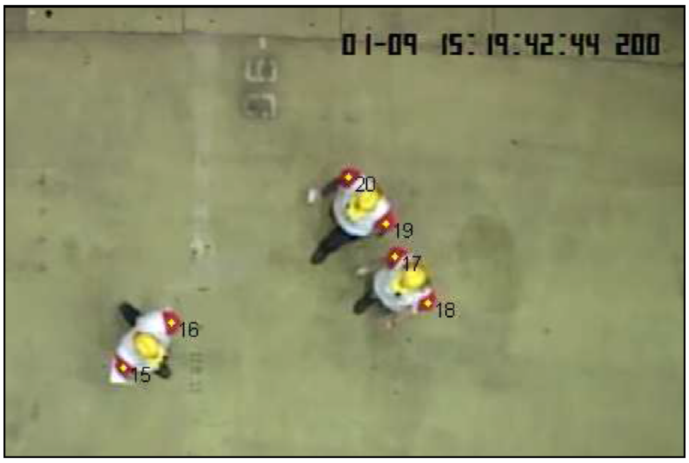
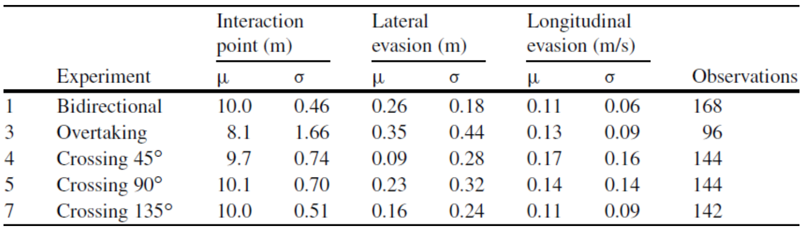


FIGURE 2.7: Experiment into pedestrian interactions. Taken from [7].

Versluis [7] and Daamen et al. [8] consider basic pedestrian interactions. In [28], Hoogendoorn and Daamen categorise pedestrian interactions as either unilateral or bilateral. In unilateral interactions, one pedestrian is not aware of the other. Examples of situations where this happens are when one pedestrian follows or overtakes another. In bilateral interactions, both pedestrians interact. Examples are when one pedestrian crosses the path of another, or in avoidance behaviour.



		Interaction point (m)		Lateral evasion (m)		Longitudinal evasion (m/s)		
	Experiment	μ	σ	μ	σ	μ	σ	Observations
1	Bidirectional	10.0	0.46	0.26	0.18	0.11	0.06	168
3	Overtaking	8.1	1.66	0.35	0.44	0.13	0.09	96
4	Crossing 45°	9.7	0.74	0.09	0.28	0.17	0.16	144
5	Crossing 90°	10.1	0.70	0.23	0.32	0.14	0.14	144
7	Crossing 135°	10.0	0.51	0.16	0.24	0.11	0.09	142

FIGURE 2.8: Results of pedestrian interaction experiments. Taken from [8].

Versluis performed a number of experiments where pedestrians walked towards each other at various angles of approach: 0°, 45°, 90°, 135°, 180°. An example of one of these experiments is shown in Figure 2.7. Various statistics were derived from these laboratory experiments, further reported by Daamen et al. in [8], are shown in Figure 2.8. They conclude that the heterogeneity of pedestrian behaviour in uncrowded situations is hard to add to simulations designed to model crowds. They also note that pedestrian interactions varied widely if one or more of the participants was hurried. Much of the resulting behaviour was context sensitive.

More and more interest from other fields in fine grained trajectory data, at the operational and microscopic level, is emerging. This is in part due to improvements in technology in pedestrian tracking and expands on the work presented above. These fields of research include, in particular, robotics [10, 29], human gait studies [30] and neuroscience [31, 32].

In [33], Bruneau et al. attempt to create a model for leader-follower behaviour in agent simulations. They define four types of distance, each representing a step in their model:

1. A physical distance which is defined by the minimum distance between the follower and the leader before contact;
2. A personal distance determined by proxemics research [34];
3. A reaction distance which gives the follower enough time to react to the leaders change in motion;
4. A safe distance to prevent any risk of collision from the leader jerkiness.

They find that the leader-follower trajectories their model generates are akin to real data. The subject is covered by Ducourant et al. in [31], from a neuroscientific perspective, in particular the ability of the follower to anticipate the speed and direction of the leader and the consequences on their speed, gait and step length. These were found to be significantly lower than the leader's.

In [35], a more subtle phenomenon is investigated: pedestrians' personal space. In Figure 2.9 various configurations of personal space are shown in Figure 2.9. In Figure 2.9(a) shows the circular space around the human body. In Figure 2.9(b) the circle describes the head orientation and visual mechanism. In Figure 2.9(c) personal space is characterised by additional margins in the front in order to anticipate and avoid oncoming pedestrians. They perform three types of laboratory experiment to measure the front zone of the pedestrian's personal space. These are: standing still while being approached by another pedestrian, walking towards a stationary object, and walking towards an approaching pedestrian. They find that the faster the walking speed, the greater the size of the front zone. When approaching a stationary object or pedestrian, this remained consistent. Walking speed affected the size of the front zone when approaching an oncoming pedestrian. The greater the approach speed, the longer this distance.

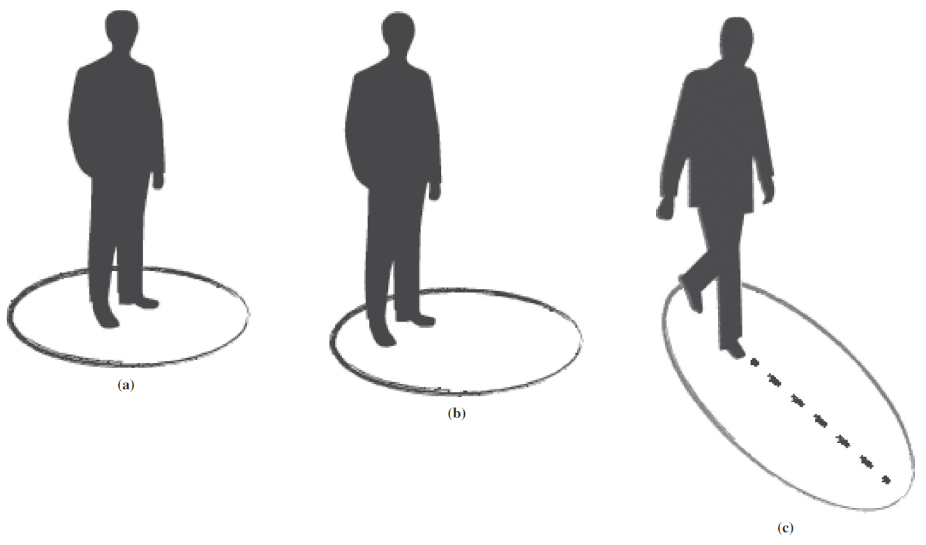


Figure 2.9: Spatial configurations of personal space. Taken from [35].

In [30], Olivier et al. investigate the crossing behaviours of two pedestrians. In laboratory experiments, two participants walk towards each other at a 90° angle. Using occluding walls, the point at which pedestrians see each other in controlled. The crossing point is then determined. They find that pedestrians are able to mutually adapt at soon as they see each other. They also find that strategies for collision avoidance are "asymmetric" in that one pedestrian gives way to another.

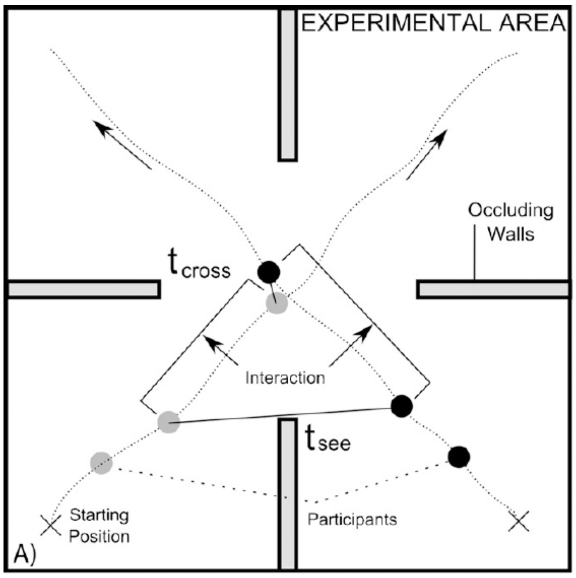


FIGURE 2.10: Experimental setup of Olivier et al.'s [30] laboratory experiments. Taken from [30].

Vieilledent et al. in [32] consider the subject of trajectory curvature and velocity. They attempt to show that pedestrian trajectories follow similar findings in handwriting and drawing studies: the "two-thirds power law." The purpose being to shed light on neurological processes. The "two-thirds power law" basically states that in the motion of the human hand, its angular velocity "is proportional to the two-thirds root of its curvature or equivalently that the instantaneous tangential velocity is proportional to the third root of the radius of the curve" [32].

In order to test this hypothesis with human walking trajectories, participants in laboratory experiments are asked to follow an ellipse on the floor. These ellipses vary in width. They find supporting evidence for the "two-thirds power law" hypothesis, suggesting intrinsic neurological processes at work in processing two dimensional space.

A final research area where pedestrian interactions are relevant is in robotics. Guy [10] states the effortless collision avoidance of pedestrians is applicable to robotics and multi-agent systems. Given that, as studies cited above have shown, anticipation of another pedestrian's actions and cooperation in avoiding collision are key to interactions, replicating this ability in robotics would be a useful goal. While Guy

[10] focuses on robot and simulated agent navigation, Dondrup et al. [29] focus on human-robot spatial interactions (HRSI). A key component of their system, along with proxemics and path planning, is the prediction of how humans are likely to behave.

This section is by no means an exhaustive survey on pedestrian interactions, their evaluation and the ultimate use of such data. It does, however, demonstrate the number of fields in which these interactions are studied, and the pertinence of pedestrian data within and outwith pedestrian dynamics research.

2.4 Conclusion

In this chapter, two very different fields have been surveyed. Research into indoor positioning systems has led to extremely diverse technologies becoming available. Traditional optical positioning systems are being either replaced or augmented by wireless network and inertial positioning systems. Many hybrid approaches are reported in the literature. This survey is by no means all encompassing. Many other technologies are also available for these purposes. Only the main ones gaining research interest are mentioned.

The second field that was looked into was research into pedestrian interactions. In contrast with the previous topic, very little research is available at this time, and all was done by laboratory experiment, underlying the difficulty in acquiring data on these behaviours. However, as technologies improve and become more ubiquitous, it is likely that more data will become available as a side-effect of this trend.

Source	$\bar{v}$	σ	Location
CROW [36]	1.40		the Netherlands
Daly et al. [37]	1.47		United Kingdom
FHWA [38]	1.20		United States
Fruin [3]	1.40	0.15	United States
Hankin and Wright [39]	1.60		United Kingdom
Henderson [40]	1.44	0.23	Australia
Hoel [41]	1.50	0.20	United States
Institute of Transportation Engineers [42]	1.20		United States
Knoflacher [43]	1.45		Austria
Koushki [44]	1.08		Saudi-Arabia
Lam et al.[45]	1.19	0.26	Hong Kong
Morrall et al. [46]	1.25		Sri Lanka
	1.40		Canada
Navin and Wheeler [47]	1.32		United States
O'Flaherty and Parkinson [48]	1.32	1.00	United Kingdom
Older [49]	1.30	0.30	United Kingdom
Pauls [50]	1.25		United States
Roddin [51]	1.25		United States
Sarkar and Janardhan [52]	1.46	0.63	India
Sleight [53]	1.37		United States
Tanariboon et al. [54]	1.23		Singapore
Tanariboon and Guyano [55]	1.22		Thailand
Tregenza [56]	1.31	0.30	United Kingdom
Virkler and Elayadath [57]	1.22		United States
Young [58]	1.38	0.27	United States
Estimated overall average	1.34	0.37	

TABLE 2.2: Free speeds observed in literature. Mean speed is denoted by $\bar{v}$ and the standard deviation where available as σ, both in $\mathrm{m\,s^{-1}}$. Taken from [26].

Chapter 3

Infra-red Sensors

3.1 Introduction

While infra-red radiation (IR) was first discovered by Herschel in 1800 [59], the development of technologies employing the properties of IR only began during WWII when lead salts were found to be effective at IR detection. As late as 1945, the German Wehrmacht deployed an active IR sniper rifle system called Zielgerät 1229 (ZG 1229)[60], codenamed Vampir, to soldiers designated as Nachtjäger (night hunters). Since then, materials, technologies and applications, both military and civilian, have become ever more diverse.

At the heart of the proposed pedestrian tracking system are six Irisys MkII pyroelectric infra-red sensors. Given the sensors' importance, the purpose of this chapter is to give an in-depth explanation of how thermal infra-red sensors operate and a detailed description of the sensors. This includes the purpose for which they are manufactured, their capabilities, and their limitations.

Irisys MkII sensors are commercial products. As such, very little is known about their actual construction and underlying mechanisms. What is known is available through public sources, their patents [61, 62], as well as limited proprietary information [63] that has been kindly shared by the manufacturer, and a paper published by Holden [64]. However, there are many sources of information about the field of IR sensing as a whole, from which this chapter draws extensively. These are Massoud [65] for the physics of thermal radiation, Kuenzer and Dech [66] for the theoretical background of IR sensing, and Rogalski and Chrzanowski [67] for information on infra-red sensor hardware and manufacturing methods.

The structure of this chapter is as follows. In Section 3.2, a basic explanation of the physical principals behind thermal radiation is given. Section 3.3 focuses on the Irisys sensors themselves. In Section 3.4, efforts to correct optical flaws in the sensors are detailed.

3.2 Infra-red

The electromagnetic spectrum is shown in Figure 3.1. All electromagnetic radiation, including thermal radiation, does not require a medium to pass through. The temperature of a surface determines the wavelengths at which thermal radiation is emitted. Wavelengths are expressed as micrometres (µm) and temperatures in degrees Kelvin (K). The region of the electromagnetic spectrum involved in heat transfer is in the range 0.1–100 µm, which includes part of the ultraviolet, all of the visible light (0.4–0.7 µm) and infra-red spectra (0.7–100 µm). The IR range is further divided into the bands shown in Table 3.1.

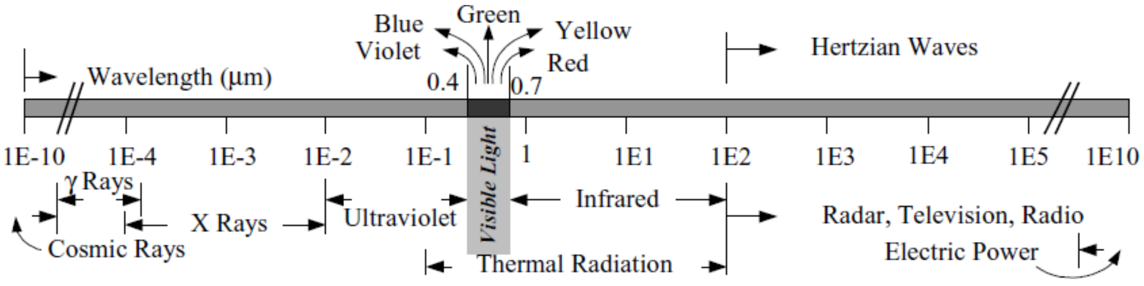


FIGURE 3.1: Depiction of the electromagnetic spectrum on a log scale. Taken from [65].

Region (abbreviation)	Wavelength range (µm)
Near infra-red (NIR)	0.7 – 1
Short wavelength IR (SWIR)	1 – 3
Medium wavelength IR (MWIR)	3 – 6
Long wavelength IR (LWIR)	6 – 15
Very long wavelength IR (VLWIR)	15 – 30
Far infra-red (FIR)	30 – 100

TABLE 3.1: Division of infrared radiation. Adapted from [67].

Thermal radiation is, however, affected by atmosphere. Molecules in the atmosphere, including water vapour and aerosols, all have an extinction effect on thermal radiation. The atmosphere has two main atmospheric windows where this effect is mitigated, in the 3–5 µm and 8–14 µm wavelengths. These are shown in Figure 3.2. The vertical axis shows the transmittance level. When this at zero, thermal radiation is blocked at that wavelength. The horizontal axis shows the wavelength in µm and the absorbing molecules at various wavelength ranges.

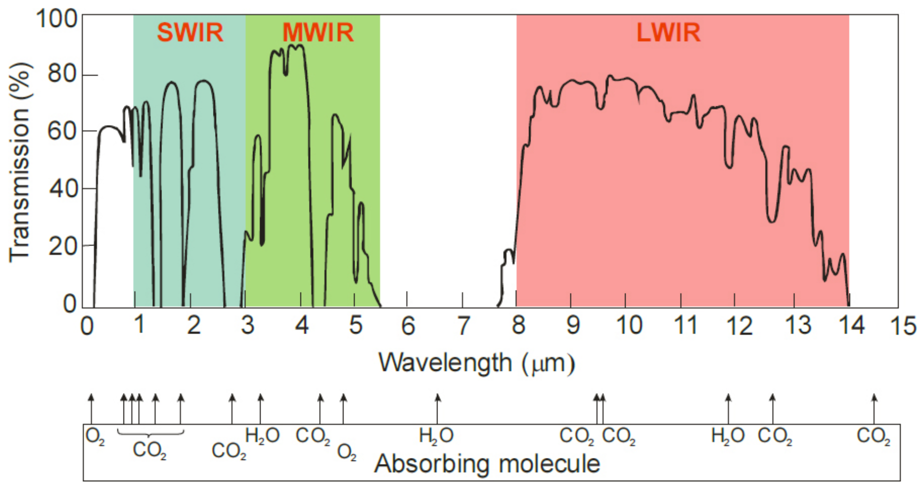


FIGURE 3.2: Atmospheric windows, with transmitance on the vertical axis and wavelength on the horizontal axis. Absorbing molecules are shown. Taken from [67].

Many active (or reflective) IR sensors, such as the Microsoft Kinect, use wavelengths in the SWIR range below the first atmospheric window, where smaller transmittance windows are available. The Irisys sensor is a passive IR (PIR) sensor, meaning that to make measurements it does not need its own source of IR radiation that reflects back from the target. However, this determines the wavelengths it must be able to capture, which is discussed next.

A fundamental principle used in infra-red technologies is that of black body radiation. A black body is an idealised concept. Any incident radiation that falls on it is absorbed. A black body that is in thermal equilibrium with its environment (i.e. at a constant temperature) emits black body radiation. The spectral properties of this radiation depend only on the absolute temperature of the black body, irrespective of its composition. Two other important properties are that a black body is an ideal emitter (it emits more energy that other materials at the same temperature but of lower emissivity) and it is a diffuse emitter (the emissions are isotropic, ie. not directionally dependent).

Radiation intensity, I_λ, is the energy density emissive from, or incident on, a surface. It is expressed in $\mathrm{W\,m^{-2}\,\mu m^{-1}\,sr^{-1}}$ where sr denotes steradians, or square radians, used to describe solid angles in three dimensional space. Spectral emissive power, E_λ, is the rate of radiation emitted in all directions at the wavelength λ, per unit surface area. It is expressed in $\mathrm{W\,m^{-2}\,\mu m^{-1}}$. The emissivity of objects ε is a constant scale factor of the radiation emitted by a surface. Emissivity is equal to 1 for a black body. All other materials fall within the range 0 to 1 [65]. For example, according to [66, Table 1.1, p.10] ice has an emissivity of 0.97, while aluminium foil has an emissivity of 0.036.

According to Ignatov et al. [68]:

> "Studies have shown that in the long wavelength infra-red region (8–14 µm) the human skin radiates as a black body, regardless of age, degree of pigmentation and other features. Therefore, the emissivity of the human skin can be considered equal to 1 absolute unit. In practice, it is proved that the difference between the emission characteristics of the human skin and black body still exist, however, it is small and depends essentially on the influence of the surrounding background." [68]

This places human infra-red emissions in the long wavelength IR (LWIR) range (see Table 3.1). Since humans can be treated as black body emitters of thermal radiation, the mathematics involved in measuring this radiation are greatly simplified. Where black body radiation is being considered, radiation intensity and emissive power is written with the additional subscript b. Radiation intensity becomes $I_{\lambda,b}$ and emissive power becomes $E_{\lambda,b}$.

According to the Stefan-Boltzmann law shown in § 2, the total emissive power of a black body is proportional to the fourth power of the absolute temperature.

$$E_b = \sigma T^4 \tag{3.1}$$

where E_b is the total emissive power, T is the absolute temperature in degrees Kelvin, and σ is the Stefan-Boltzmann constant, $\sigma = 5.67 \times 10^{-8} \mathrm{W/m^2K^4}$. The spectral emissive power of a black body is given by Max Planck's equation. He expressed radiation intensity $I_{\lambda,b}$ in terms of wavelength and temperature.

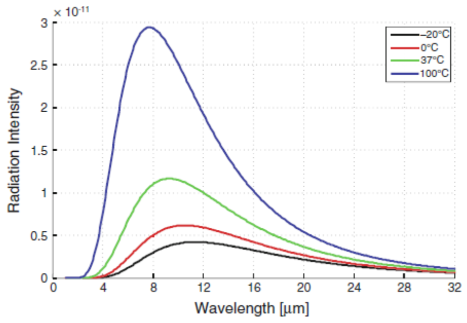
$$I_{\lambda,b}(\lambda, T) = \frac{(2\hbar c^2)\lambda^{-5}}{\exp(\hbar/k\lambda T) - 1} \tag{3.2}$$

where λ is the wavelength, $\hbar$ is the Planck constant, $\hbar = 6.6256 \times 10^{-34}\,\mathrm{J\,s}$, c is the speed of light in a vacuum, $c = 2.998 \times 10^{8}\,\mathrm{m\,s^{-1}}$, and k is the Boltzmann constant, $k = 1.439 \times 10^{4}\,\mathrm{\mu m\,K}$. The results of this for various temperature ranges is given in Figure 3.3(a).

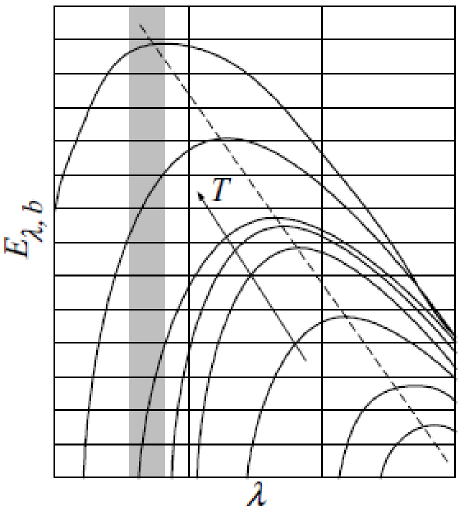
Since black bodies are isotropic diffuse emitters, their total emissive power, given a certain wavelength and temperature, is given by:

$$E_{\lambda,b}(\lambda, T) = \pi I_{\lambda,b}(\lambda, T) \tag{3.3}$$

From § 3, Planck's distribution can be derived, shown in Figure 3.3(b). This shows that as the surface temperature increases, the peak spectral emissive power shifts towards shorter wavelengths.



(a) Spectral radiation intensity of black body radiation. Taken from [69].



(b) Planck distribution showing the spectral emissive power of black body radiation. The arrowed black line labelled T shows the direction of temperature increases. Taken from [65].

FIGURE 3.3: Spectral radiation intensity ($I_{\lambda,b}$) and emissive power ($E_{\lambda,b}$) of black body radiation.

In order to determine the maximum or peak wavelength, it is necessary to use Wien's displacement law. In Figure 3.3(b), the dotted diagonal line moving from right to left passes through the peak wavelengths of each curve. The maximum or peak wavelength of a black body at a certain temperature is calculated using the equation below.

$$\lambda_{max} = \frac{b}{T} \tag{3.4}$$

where λ_{max} is the maximum wavelength, b is Wien's displacement constant (2897.6 µm K) and T is absolute temperature of the black body in degrees Kelvin. For example, given an absolute temperature of 310.14 K (37 °C), the maximum wavelength of the emitted radiation is given by:

$$\lambda_{max} = \frac{2897.6}{310.14} \approx 9.34\mu\text{m}$$

This falls within the range given above for human emissivity. In practice, peoples' temperatures, through clothes, are around 28 °C. Using § 2, the peak wavelength at this temperature is approximately 9.62 µm.

These concepts will be revisited in Section 3.3.2 where the choice of lens materials is determined by the desired wavelength coverage, and in Section 3.4, where a substitute IR source is used with the sensors.

3.3 Irisys sensors

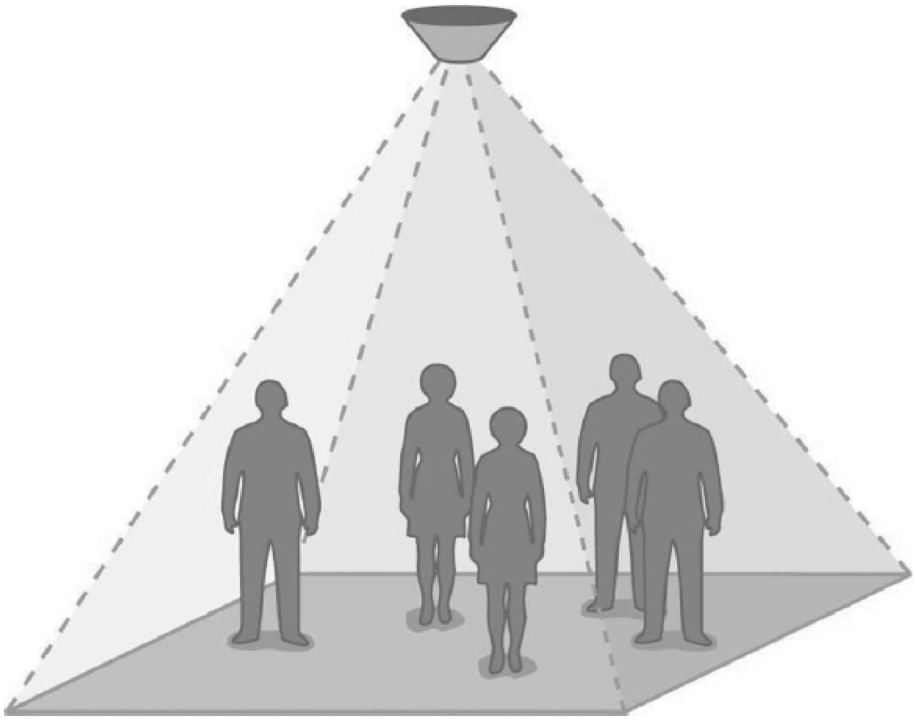


FIGURE 3.4: Irisys sensors' field of view. Taken from [64].

Irisys manufacture a range of relatively low-cost but low-resolution passive infra-red (PIR) sensors for the purpose of capturing information about people's movements. These sensors are ceiling mounted, as shown in Figure 3.4. Their field of view is 4×4 m. The sensors are mounted at 3.5–4 m in height, and are designed to pick up heat sources at 1.5 to 2 m from the ground, the average height of adults' heads and shoulders. In addition to their relative low-cost and unobtrusive installation, Holden [64] refers to the anonymisation of data gathered. It is impossible to determine people's identities or activities from the data collected by the sensors.

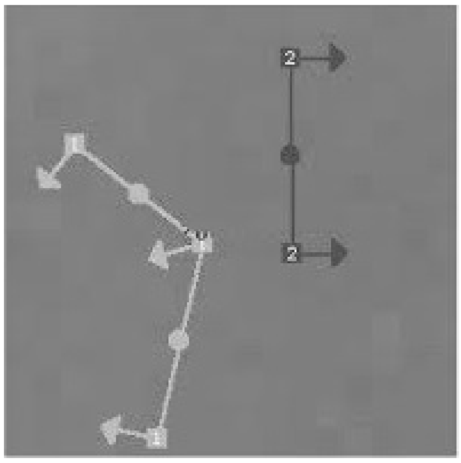


FIGURE 3.5: Arbitrary datum lines (light grey and black lines) defined on a sensor's field of view.

The sensors are able to collect data irrespective of ambient light levels. They are widely deployed in retail environments, to count people crossing an arbitrary datum line, as shown in Figure 3.5. In order to do so, pedestrians (also called sensor targets or targets) are tracked by the onboard tracking system. Queue management, another application that the sensors are intended for, is illustrated in Figure 3.6.

The construction of these particular sensors is explained in the following sections. Section 3.3.1 discusses the Focal Plain Array (FPA), Section 3.3.2 the optics of the sensors, Section 3.3.3 the sensors' onboard Digital Signal Processor (DSP), and, finally, the sensors' communications abilities are briefly described in Section 3.3.4.

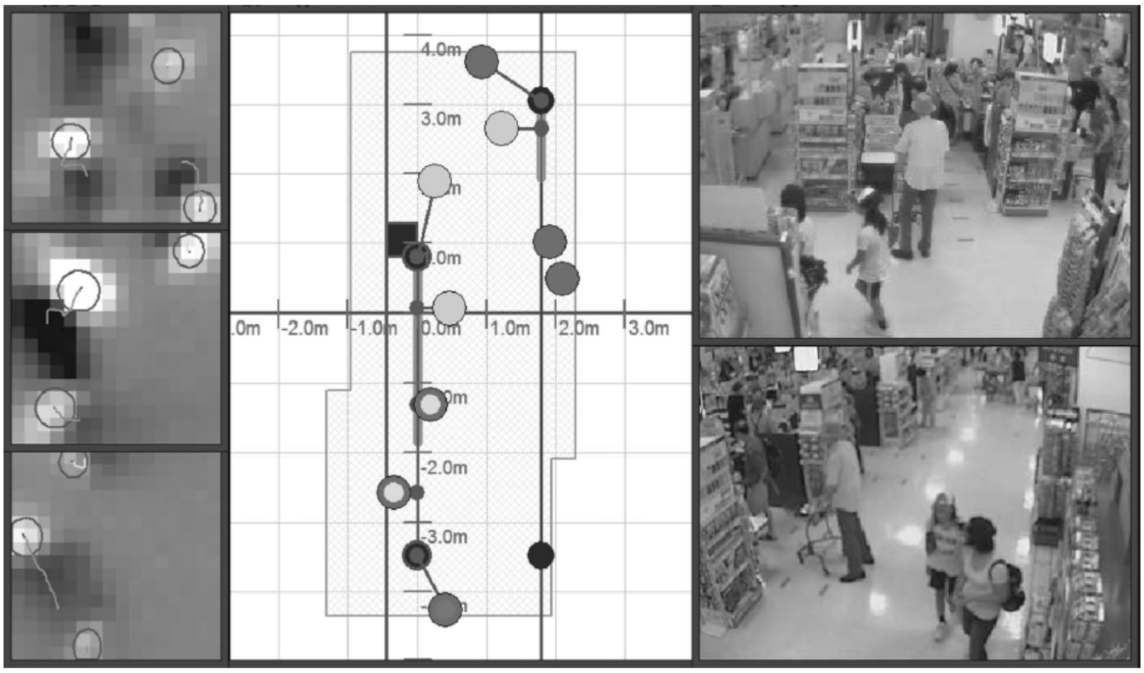


FIGURE 3.6: Layout for a supermarket queue monitor showing, from left to right, the raw thermal image, the analyzed queue layout, and oblique video stills of the actual queue forming. Taken from [64].

3.3.1 Focal plane array

A focal plane array (FPA) is an assemblage of detector pixels located within the focal plane of the system's lens. These can be either one or two dimensional. In the case of the Irisys sensor, the FPA is a two dimensional 16 $\times$ 16 array. The FPA in the Irisys sensor is made from a piezo- and pyroelectric mixed-oxide ceramic [64]. Pyroelectric materials react to incident radiation by local changes in polarity, creating a voltage change and producing an electrical output that is picked up by the underlying circuitry. These materials are also piezoelectric, meaning they react to vibrations, something that the manufacturer has sought to address in the design of the packaging of the FPA. Although a great deal of spectral information is contained within IR radiation, only photon detectors can capture this. Thermal detectors, as in this case, only capture changes in the radiant power of the incident radiation, over the entire wavelength band covered by the system. The IR wavelength range within which this incident radiation is captured by the sensor is determined by the material used to make the lens which is explained in the next subsection.

FPAs can be manufactured in one of two ways, using either monolithic or hybrid methods. It is not clear from patent the sensors' patent documents [61, 62] which of these methods has been used in their manufacture. The patents cover both possibilities. Holden [64] states that the sensors are manufactured using the hybrid method, although it is not known if that was the case for earlier sensors from Irisys, like the ones used in this work. According to Rogalski and Chrzanowski [67], monolithic FPAs are difficult to produce and the technology involved has not yet reached maturity. This suggests that the hybrid method has been used in the manufacture of all Irisys products.

The distinction between the two manufacturing methods concerns the way in which the FPA's output is multiplexed. In the monolithic manufacturing method, multiplexing is done within the FPA itself. In the hybrid method, the detector is connected to a separate readout circuit, which is bonded onto the detector array using "bumps" [67]. According to Holden [64] this is done using a stencil printed interconnect conducting adhesive (ICA) in Irisys sensors, although in [67] indium is suggested as a material.

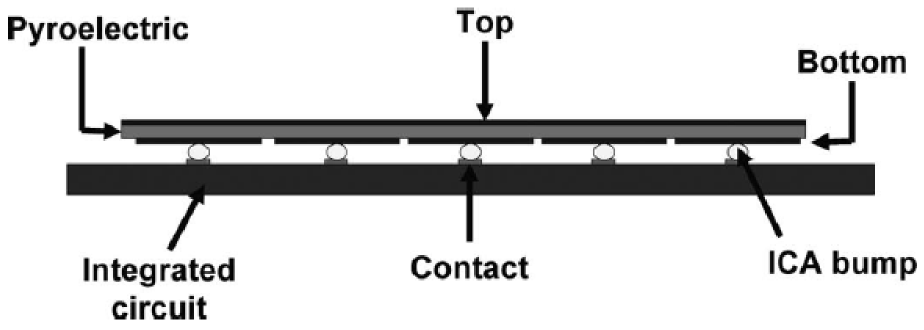


Figure 3.7: A schematic detailing the construction of a pyroelectric detector array hybrid as manufactured by Irisys. Taken from [64].

The schematic for the Irisys sensors is shown in Figure 3.7, with definitions given of the components in Table 3.2. The pyroelectric material in the FPA has contact material above and below it, that completes a circuit allowing the material to produce a current when incident radiation reaches it. An application-specific integrated circuit (ASIC) scans each pixel in turn. Contact between the ASIC and the FPA is achieved using ICA bumps as discussed above.

Item	Description
Pyroelectric	Pyroelectric ceramic wafer (200 µm thick)
Top	Continuous contact that is transparent to IR radiation
Bottom	Continuous contact that completes the circuit between the top and bottom of the pyroelectric ceramic wafer so that voltage differences can be generated
Integrated circuit	Application-specific integrated circuit (ASIC) that reads from each array element in a regular scan
Contact	Contact between the ICA bump and ASIC
ICA bump	Stencil printed interconnect conducting adhesive (ICA)

Table 3.2: Description of the elements from the schematic in Figure 3.7.

Infra-red cameras differ from infra-red sensors in that they are equipped with "choppers", a rotating blade that covers the lens so that static elements can be viewed. Holden [64] explains that chopping has an upper limit of about 10 Hz, which is too slow for the sensors' application. They must respond to slow moving targets, "translating to response at well below 0.1 Hz" [64]. The drawback is that sensors often lose targets if they become stationary, whereas IR cameras don't. On the other hand, since the background is static, it is not registered by the sensor which simplifies onboard processing.

3.3.2 Lens materials

As can be seen in Figure 3.8 and Table 3.3, there are many materials available that are suitable for IR optics. One is normal glass, which can capture not only visible light wavelengths but also those up to 2.3 µm, in the SWIR range. This explains how visible light video cameras with glass lenses sometimes have built-in night vision capabilities. The available waveband provided by lens materials is particularly relevant for passive IR sensors. The emissions of the target determine which waveband needs to be covered. In active (or reflective) IR sensors, the IR radiation is emitted by the sensor itself and its reflection from the targets is measured. The choice of waveband can, therefore, be determined by the sensors' designer. In passive sensors, the waveband is determined by the natural emissions of the target, which the sensor must be adapted to.

The choice of material therefore depends on the application. The waveband each material covers is shown in Table 3.3. In the Irisys sensor, a germanium (Ge) lens is used. While human emissions peak in the 8–14 µm range, Germanium lenses capture IR emissions in the 2–12 µm range, creating an overlap between 8–12 µm.

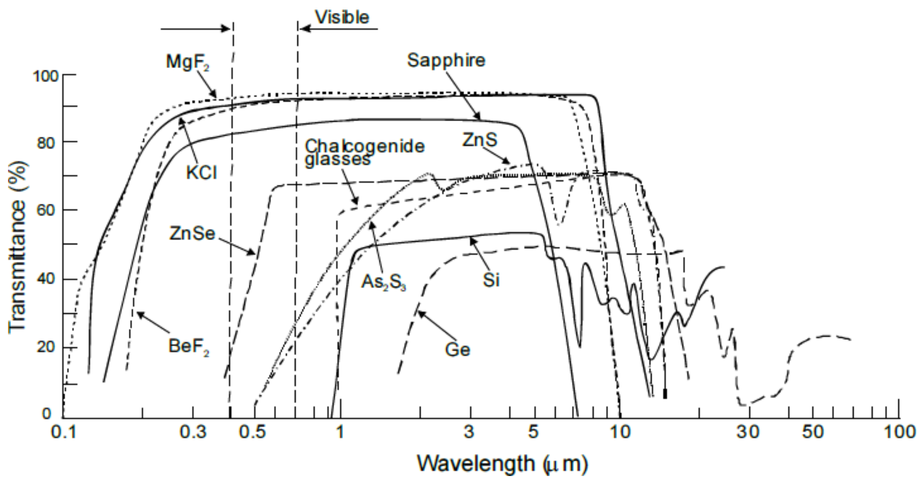


FIGURE 3.8: Lens materials used for IR optics. Taken from [67].

Material		Waveband (µm)	Characteristics
Germanium	Ge	2–12	Brittle, semiconductor, diamond-turning capability, visibly opaque, hard.
Chalcogenic glasses		3–12	Amorphous IR glass, can be slumped to near-net shape.
Silicon	Si	1.2–7.0	Brittle, semiconductor, diamond-turned with difficulty, visibly opaque, hard.
Gallium arsenide	GaAs	3–12	Brittle, semiconductor, visibly opaque, hard.
Zinc sulfide	ZnS	3–13	Yellowish, moderate hardness and strength, can be diamond turning, scatters short wavelengths.
Zinc selenide	ZnSe	0.55-20	Yellow-orange, relatively soft and weak, diamond-turning capability, very low internal absorption and scatter.
Calcium fluoride	CaF_2	3–5	Visibly clear, diamond-turning capability, mildly hygroscopic.
Sapphire		3–5	Very hard, difficult to polish due to crystal boundaries.
Glass		0.35–2.3	Typical optical glass.

TABLE 3.3: Description of some of the lens materials shown in Figure 3.8. Adapted from [67].

According to Rogalski and Chrzanowski [67]:

> "Germanium is a silvery metallic-appearing solid of very high refractive index (> 4) that enables design of high-resolution optical systems using a minimal number of germanium lenses. Its useful transmission range constitutes from 2 to about 12 µm. It is quite brittle and difficult to cut but accepts a very good polish. Germanium is non-hygroscopic and non-toxic, has good thermal conductivity, excellent surface hardness, and good strength. Additionally, due to its very high refractive index, antireflection coatings are essential for any germanium transmitting optical system. Germanium has low dispersion and is unlikely to need

colour correcting except in the highest-resolution systems. A significant disadvantage of germanium is the serious dependence of its refractive index on temperature, so germanium lenses may need to be athermalized. In spite of high material price and cost of antireflection coatings, germanium is a favourite choice of optical designers of high performance infrared objectives for thermal imagers." [67]

The difficulty in producing germanium lenses given the hardness of the material, particularly good quality wide angle lenses, means that in the case of the Irisys sensors, they suffer from considerable optical aberrations. This is discussed further in Section 3.4.

3.3.3 Digital signal processor

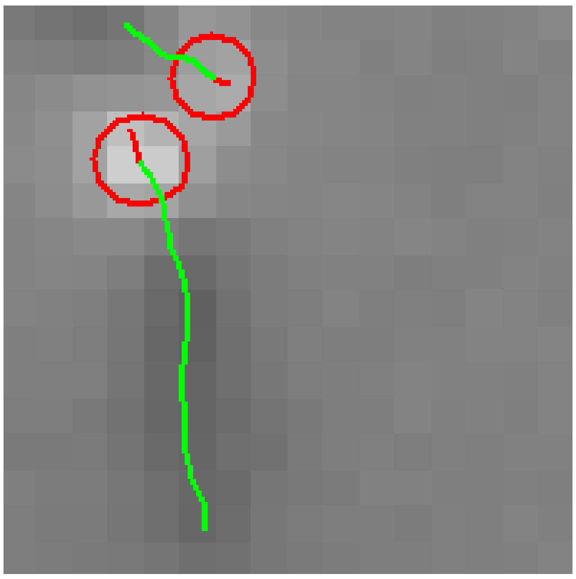


Figure 3.9: Array image returned by the Irisys sensor

It is important to note that the sensors themselves do not do any processing of information beyond digital signal processing, discussed in Section 3.3.3. This is done by the manufacturer's own propriety software running on a remote desktop that the sensors are connected to. Much of the information returned by the sensors is optimised and encoded for use by this software. The format of much of this is unknown. What is known is that each update returned contains an identifier for each target (or pedestrian) and their x and y coordinates within the sensor's field of view. An example of information inaccessible in a custom use sense is shown in Figures 3.5, 3.6 and 3.9, is the "array data." This is the grey scale representation of information returned from the 16×16 sensor element, and is in an unknown format, and therefore is not used in this research.

Array data shows the intensity of each pixel, ranging from bright white to black, according to detected temperature differences. An increase in temperature is shown by the brightening of the pixel, a decrease by its darkening. The coordinates of targets (pedestrians) are determined using ellipse fitting by the DSP. This allows for subpixel accuracy and compensates for the relatively coarse 16 x 16 array of PIR cells built into the sensor. The returns from the DSP are shown in Figure 3.9. In this figure, pedestrians are shown as bright white pixels. They are followed by a darker wake as the elements of the pyroelectric array cool back to the ambient background temperature. Each pedestrian has an ellipse fitted to identify them, shown in red. The DSP is able to follow pedestrians across the field of view, reporting their trajectories, shown here in green.

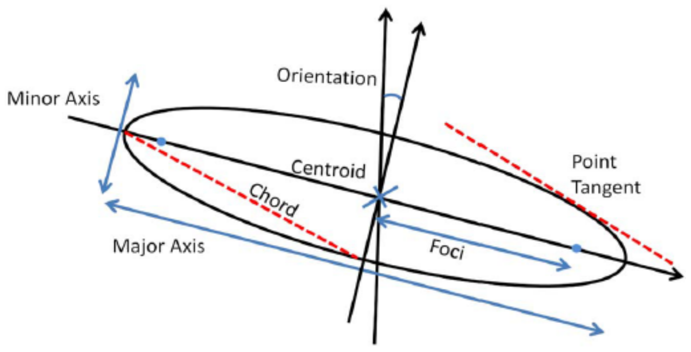


FIGURE 3.10: The parameters (blue) and geometric properties (red dotted line) of an ellipse. Taken from [70].

Figure 3.9 shows that, seen from above, people register as ellipses. As the sensors are dependent on temperature change to track pedestrians, should one stand still for a short length of time (approximately 1 second), they disappear from the sensor's view. There is no change in the background temperature for the sensors to track.

As a consequence, the sensors often lose targets, creating data processing problems discussed further in Chapter 6.

A second type of data returned by the DSP is "target data." This is a frame of fixed length packets of hexadecimal values, providing the active targets' identifiers, x and y coordinates and various other data items. Targets' coordinates are returned as Euclidian coordinates, relative to the origin of the sensor's coordinate system, the bottom left corner. Although the necessary information required for decoding coordinates from target data was kindly provided by Irisys, the format of array data and other data items remained confidential.

Ellipse fitting is a very active area of research in image processing, albeit with images at a much higher resolution. A survey of ellipse fitting methods are provided in [70, 71]. According to [70] ellipse detection methods have gained a great deal of maturity in the last 30 years. Ellipses have the benefit of being easy to detect and allow simple feature extraction. Ellipses have many useful mathematical properties, and provide a great deal of flexibility. In effect, circles are a special case of ellipse where the major and minor axes (see Figure 3.10) are equal.

Although there are many optimisation algorithms for fitting ellipses, the basic procedure in image processing remains the same. The first step is background subtraction from the source image. The second is edge detection using the remaining pixels. The third is using an optimisation algorithm to fit the ellipse using the edge pixel data. Finally, the results are returned.

Given the low resolution of the Irisys sensors, and the fact that the background in any image is easy to discern, this task appears to be much simplified, although the exact details of how this is done are unknown. Ellipses have the following parameters: their centroid (x_0, y_0), orientation (α), and length the major and minor axes (a and b). The standard formula for ellipses given in Equation (3.5).

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{3.5}$$

Any point (x, y) on an ellipse must satisfy Equation (3.5), where a and b are the minor and major axes. The centroid (x_0, y_0) is at the intersection of a and b. Using these properties, the DSP is able to return the centroid with sub-pixel accuracy.

3.3.4 Communication capabilities

The sensors have three available communication protocols, a simple relay system, a Controller Area Network (CAN) bus, and RS232 capabilities. In this instance, RS232 is used. The maximum serial port speed supported is 115 Kbps. The sensors return data on average every 30 ms. A naïve RS232 protocol is used by the sensors; no handshaking capabilities are provided.

3.4 Optical distortion correction

The five basic types of aberration, due to the geometry of lenses or mirrors. These are applicable to systems dealing with monochromatic light and are known as Seidel aberrations. They were first described in a paper published in 1856 by Ludwig von Seidel [72]. Lens made to capture infra-red radiation suffer the same problems as glass lenses meant for visible light.

The fifth of these aberrations is called simply distortion, although this is often called barrel or pincushion distortion. The light from points in the object might be brought together on the image plane at the wrong distance from the optical axis, instead of being linearly proportional to the distance from the optical axis in the object. If distance increases faster than in the object, pincushion distortion takes place, if more slowly, barrel distortion. In this instance, the type of distortion is pincushion distortion. Collectively, these are called radial distortion.

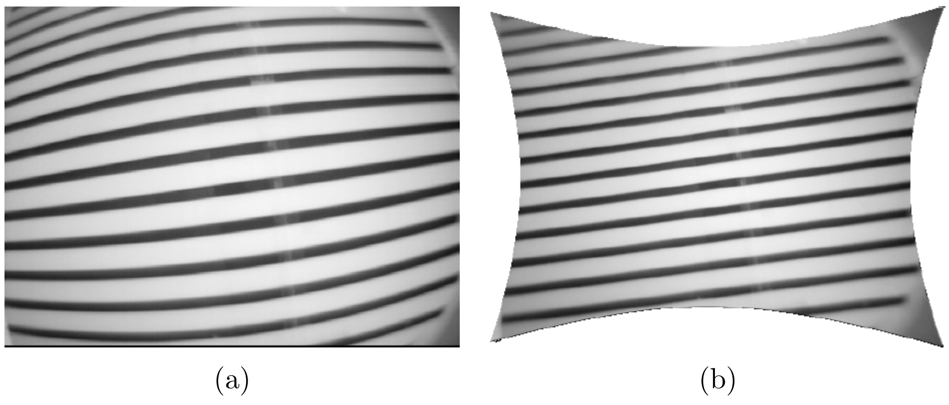


FIGURE 3.11: An example of (a) pincushion distortion and (b) the same image with the distortion corrected. Taken from [73].

As can be seen in Figure 3.11(a) the sensor "sees" a rectangular area, however, the area it actually covers (the "ground truth"), shown in Figure 3.11(b) is shaped like a pincushion. According to Tsai in [74]:

> "There are two kinds of distortion: radial and tangential. For each kind of distortion, an infinite series is required. However, my experience shows that for industrial machine vision application, only radial distortion needs to be considered, and only one term is needed. Any more elaborate modeling not only would not help but also would cause numerical instability." [74]

Mathematically, it is described by the following Taylor polynomial, with the first term κ_1 being dominant. Here r is the radial distance, and r' the corrected radial distance.

$$r' = r + \kappa_1 r^3 + \kappa_2 r^5 + \ldots + \kappa_n r^{2n+1} \tag{3.6}$$

Unfortunately, it is impossible to obtain the necessary value of κ_1 to correct distortion analytically. Numerical methods must be used instead. These yield an approximate value of κ_1 that produces the best results in correcting the distortion

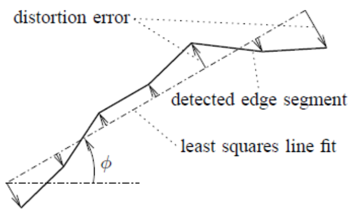


FIGURE 3.12: Line straightening in distortion correction. Taken from [73].

with the data available. It was only necessary to carry out this process once, before data collection was started in earnest.

3.4.1 Methods

A great deal of research effort has been dedicated to finding efficient and capable algorithms for correcting pincushion distortion. Most of these approaches have been targeted at photographic stills or video footage. A survey of the literature has failed to find a single instance of pyroelectric IR lenses being addressed.

However, some of the ideas used to correct traditional (visible light) data can be applied to this problem. In particular, Devernay and Fauregas describe in [73] the use of straight lines in photographs to correct distortion in software. By extracting features from photographs representing lines, they were able to calibrate lenses by finding optimal values to make these lines straight. Figure 3.12 illustrates this technique. This inspired the formulation of an experimental setup that guaranteed a target emitting infra-red radiation and moving in a straight line. As PIR sensors cannot detect stationary radiation sources, a moving target is necessary. Straight line data can then be processed to find the single most optimal value of κ_1.

A LEGO®[1] trainset (Figure 3.13) was acquired for this purpose. This allowed a small bottle containing 200 ml of hot water to be pulled along a straight track, underneath a Dexon rig that held a single sensor. Care was taken to ensure that the test rig was at the centre of the track, and that the curves in the track were outside

[1]LEGO and the LEGO logo are trademarks of the LEGO Group.
© 2016 The LEGO Group.

the field of view. By moving the sensor along the rig, relative to the track, straight line data was collected at roughly 10 cm increments within the field of view of the sensor. Six sensors were tested in this way. Examples of data gathered are shown in Figure 3.15.

There are many differences between such an IR source and a human target. These include size, speed and temperature. As discussed in Section 3.3.3 the tracking software in the sensors is able to pick up a target when it overlaps, at minimum, more than one pixel. The sensors are optimised to detect targets that are between 1.5 m and 2 m below them. In this case, the test rig was 1.5 m above the track. The IR source was of sufficient size to meet the minimum requirement, fill more than one array pixel, and therefore be tracked.

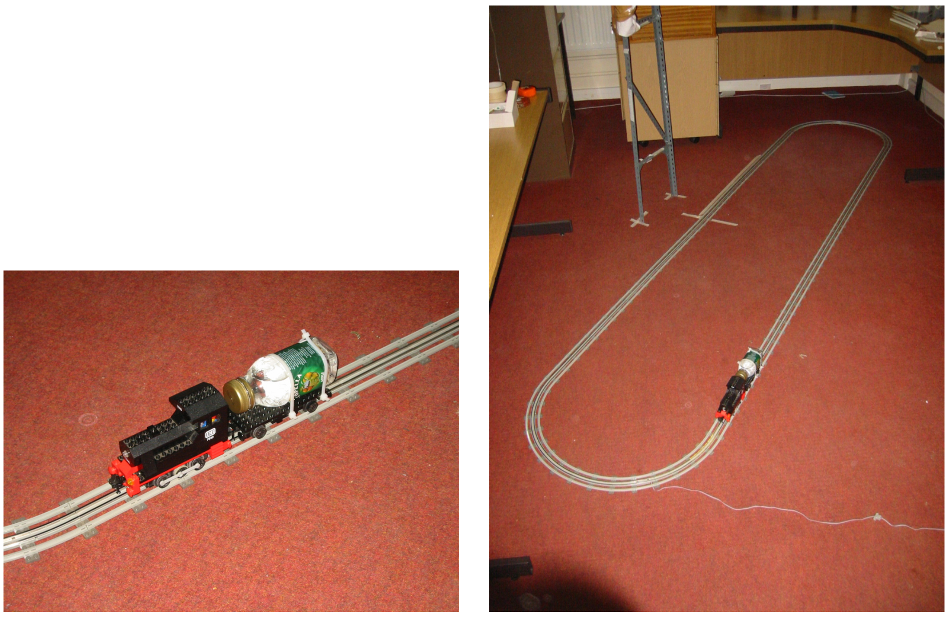


FIGURE 3.13: (a) LEGO train with IR source; (b) LEGO track.

In addition, updates from the sensors averaged at around 33 ms, which is was sufficient to accommodate a constantly moving target, albeit moving more slowly than a pedestrian would at 1 m s^{-1}. As explained in Section 3.3.1, the sensors are optimised to cope with slow moving targets.

In order to mimic a human target, the hot water bottle had to emit sufficient thermal radiation for the sensors to pick it up. Clear plastic has an emissivity of 0.94

and could, therefore, be considered close to a black body emitter, as discussed in Section 3.2. Hot water was found to be at or around 60 °C from the tap. As it cooled, by the time it was at 30 °C, it could no longer be detected by the sensors. In Figure 3.14, the spectral emissive properties of a black body at both these temperatures are shown. As can be seen, the peak wavelength for both temperatures is within the 8–12 µm range covered by the sensors.

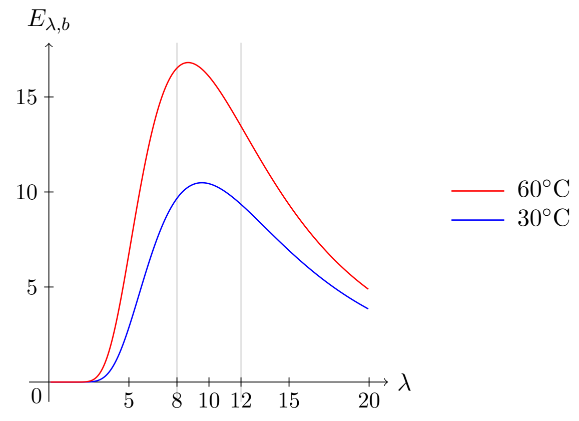


FIGURE 3.14: Spectral radiation intensity $I_{\lambda,b}$ of a black body emitter at 30 °C and 60 °C. The vertical axis represents spectral emissivity (W m^{-2} µm^{-1} sr^{-1}), the horizontal axis represents wavelength (µm)

In Figure 3.15, the point clouds for each graduation are shown as a jagged line. The axes are scaled according to the Euclidian coordinates of the sensors. The driving direction of the trains is shown in the legend. The distortion increases the further away the line is from the centre of the lens. Additionally there is a greater amount of noise in the lines, as demonstrated by the lines at the top and bottom of the example shown. Only some of this extra noise is caused by the pincushion distortion. The rest may be due to to a greater amount of error and uncertainty in the data processing algorithms of the sensor at these points. Precise causes for this are unknown. Therefore, the data returned by the outer two pixels of the sensors were ignored when calculating κ_1.

By ignoring all but the first terms of Equation (3.6), the challenge of finding an optimal value of κ_1 becomes a single non-linear optimization problem. The functions necessary to perform this optimization are available in the MATLAB [75] software

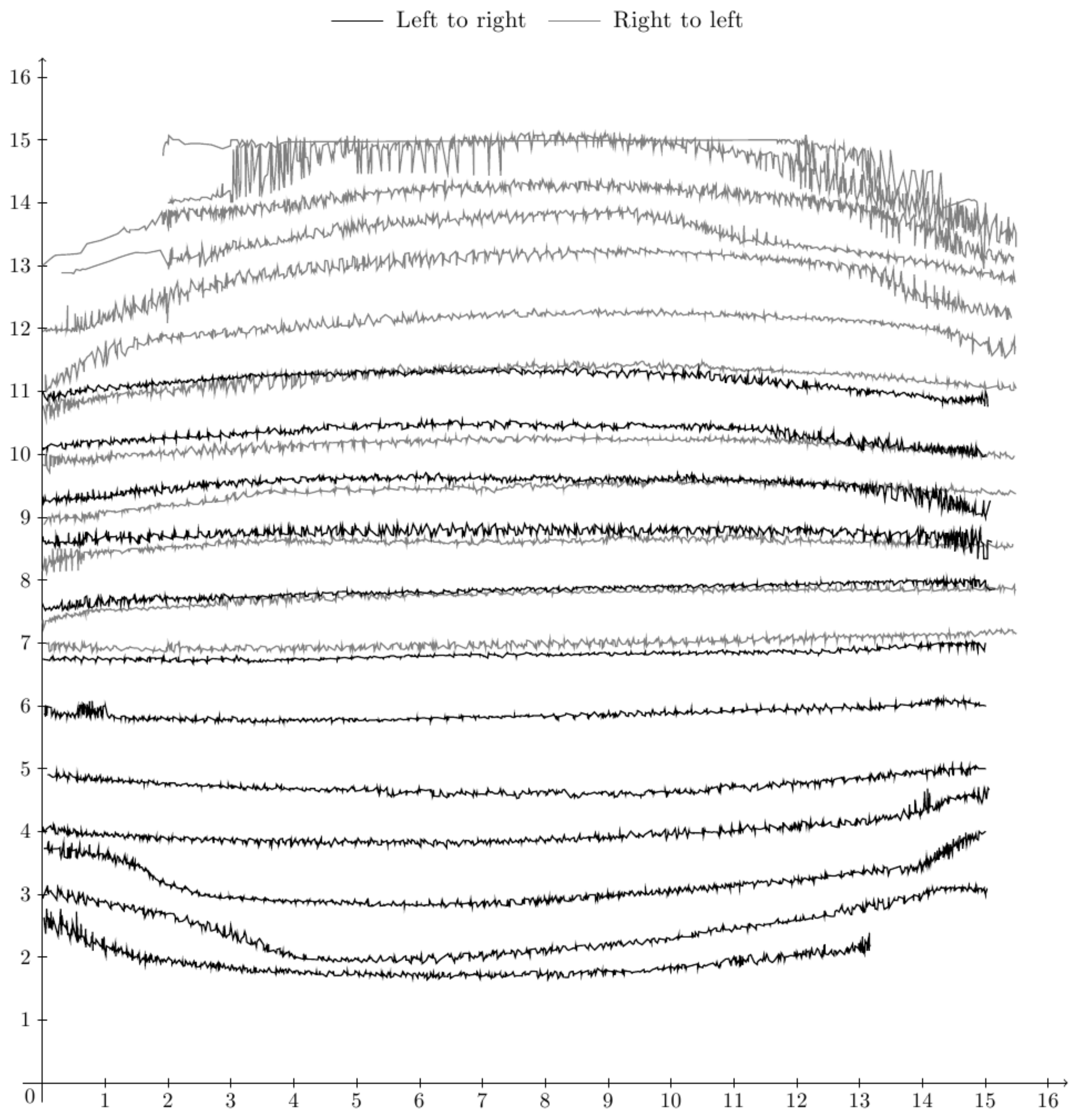


FIGURE 3.15: Recorded data from IR source moving in a straight line

package. An objective function was programmed in using MATLAB's built in programming language, given in Algorithm 3.1. Each array of points, representing a line shown in Figure 3.15, was loaded into an array. The objective function then calculated how well the data fit a straight line for any given value of κ_1. MATLAB functions then minimised this value. An example of corrected line data is shown in Figure 3.16, giving some indication of the extent of the distortion and how much it could be corrected using these methods.

```
function [error] = kappasearch(kappa)
load vars x1 y1;
for i = 1:length(x1)
    [uxs(i), uys(i)] = undistort(x1(i), y1(i), kappa);
end
linereg = polyfit(uxs, uys, 1);
lys = polyval(linereg, uxs);
err = 0;
for i = 1:length(uxs)
    err = err + (lys(i) - uys(i))^2;
end
error = err;
end
```

ALGORITHM 3.1: Objective function for finding an optimal value for κ_1.

3.4.2 Results

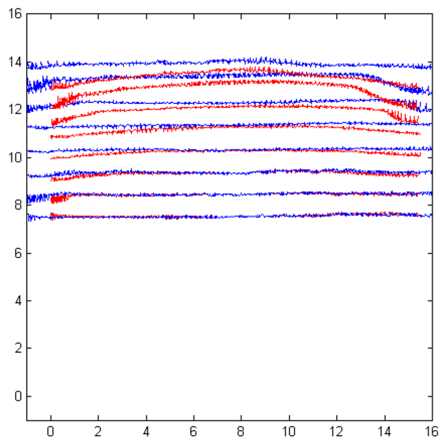


FIGURE 3.16: Corrected line data (blue) and distorted line data (red).

Straight line data were collected from six individual sensors and then processed to find a value of κ_1. Despite the sensors being from different manufacturing batches, the results were very similar, as shown in Table 3.4. The average of these values, $\kappa_1 = 0.0022$, was used thereafter in correcting data received from the sensors.

Sensor	κ_1
1	0.0023
2	0.0022
3	0.0022
4	0.0023
5	0.0021
6	0.0021
Mean value	0.0022

Table 3.4: Values of κ_1 for six sensors, rounded to 4 dp.

3.4.3 Discussion

A great deal more could be done to find optimal solutions to distortion problems in the sensors. In these experiments, the 10 cm graduations were imprecise, and sometimes the sensor was slightly rotated on the test rig. Although the method suggested by Devernay and Fauregas [73] does not require a particularly high level of precision, it is very hard to estimate the ground truth from these graduations with any great certainty, nor was that the purpose of the experiment. Better results can be achieved with better equipment in a less ad-hoc setting. This, however, was not available. An additional consideration is whether the investment in time is worthwhile. Since this research, Irisys have brought out two further models of people counters. Their lenses may be manufactured in the same way and suffer the same problems, perhaps making a re-evaluation worthwhile. However, at the same time, consumer rather than commercial IR products have been brought to the market that have considerable hardware advantages, such as the Microsoft Kinect, now at version 2.0. This is very well documented platform with far better support for custom applications and software development, available at a similar price. It is also highly flexible and is not constrained to a single application in the same way as the Irisys products are. It is highly likely that more sophisticated products will continue to come to market in the future.

3.5 Conclusion

In this chapter, infra-red radiation and the means for detecting it have been explained. Additionally, the IR sensors used in this research have been thoroughly examined. Problems related to their use have also been addressed. In the next chapter, their deployment in a public area is described.

Chapter 4

Study Location

4.1 Introduction

In order to assess the feasibility of using the Irisys MkII sensors, six were deployed in the ceiling tiles in a corridor of the Merchiston Campus of Edinburgh Napier University. An infrastructure was built to record data from the sensors and save this unto the hard drive of a standard PC desktop, as well as provide power and connectivity to a small private network.

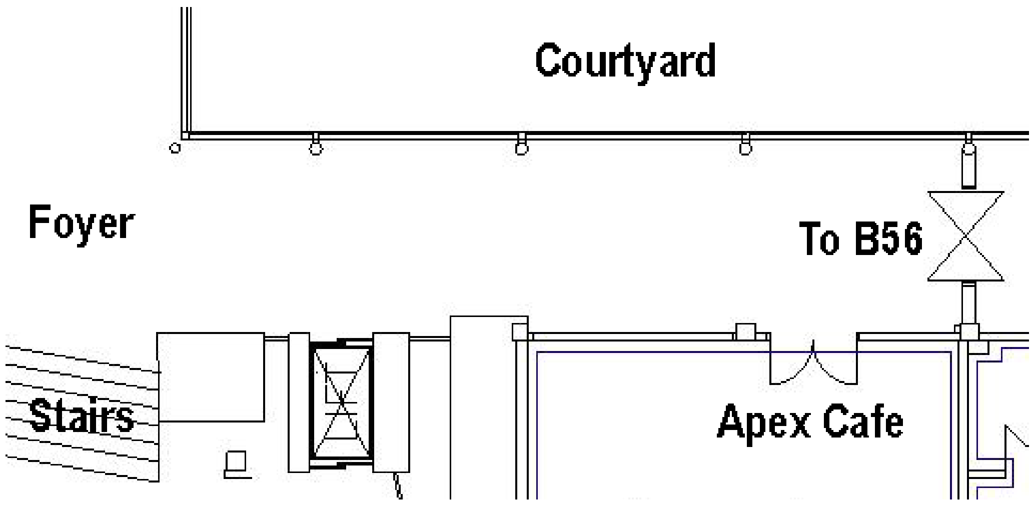


FIGURE 4.1: Annotated building blueprint of the corridor being studied.

The system was intended to run automated and unattended to record as much data as possible. The corridor used in the study is described in Section 4.2, followed by the infrastructure in Section 4.3. In Section 4.4, the software used to collect data is described.

4.2 Study Location

This particular corridor was chosen because it leads to an on campus catering outlet, the Apex café, guaranteeing a heavy traffic flow. It also leads to a computing lab and the campus foyer. Traffic is constant to all the three possible exits. Additionally, the ceiling panels made it possible to mount the sensors and run cabling above them.

Although not to scale, Figure 4.1 shows the basic layout of the area under investigation.

The corridor itself is 15 m long and 3.75 m at its widest point. The ceiling is at a height of 4 m. Photographs are provided in Figures 4.7(a) and 4.7(b). The length of the corridor, and its relative width, as well as number of origins and destinations, made it an ideal location to instrument.

Although this corridor is ideal for gathering pedestrian data, there are a number of features that may affect the application of the results of this study to other corridors. In particular, the wall opposite the entrance to the Apex café shown in Figure 4.7(c) is made of glass, and is perpendicular to a glass wall of the foyer. This creates the possibility that pedestrians alter their behaviour when walking to and from the foyer as they can see oncoming traffic. A second feature of this location is a staircase at the edge of the area in the foyer, shown in Figure 4.7(d).

Both these features are unusual in basic standard building corridors. Unfortunately, due to time constraints, it has been impossible to study a different location. Therefore it is not possible to say if either of these features affect pedestrian behaviour.

However, this corridor benefits from a constant flow of traffic, day and night. Night time traffic is due to people coming the computer centre in an adjacent building that is open 24 h a day.

4.3 Hardware Infrastructure

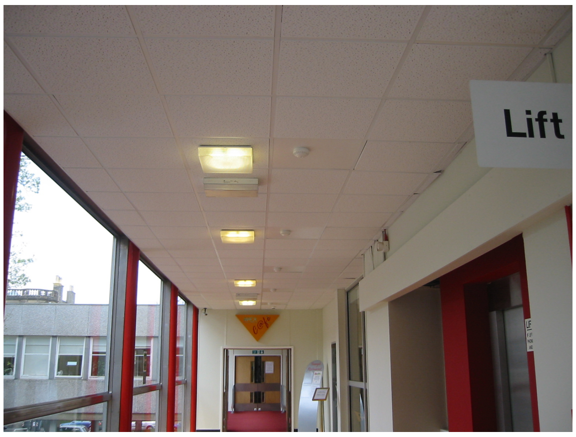


Figure 4.2: Photograph of emplacement of the sensors in the corridor.

In Section 3.3, the limitations and features of the Irisys Mk II sensor were described. The limited field of view (4×4 m) and the limitations of the communications protocol used (RS232 serial communications [76]) created a number of challenges. As their field of view is limited, six sensors were required to cover the length of the corridor. The leftmost sensor in Figure 4.1 also provided some coverage of the foyer. Unfortunately, one of the sensors failed. This was the sensor closest to the door heading towards the computer laboratory in B56. Since this did not affect coverage of the door towards the café, it was decided to continue. Its location is marked in Figure 4.5, although its field of view is not, to the right of the diagram.

The emplacement of the sensors in the corridor is shown in Figure 4.2. The sensors were placed on wooden boards that were the same width and length of the ceiling panels they replaced. Wiring was threaded through a hole cut into each board, and passed along on top of the ceiling panels. Once a power supply was secured, the RS232 cables were added. Unfortunately, long cables did not work reliably. An attempt to add modems to the cables to amplify their signal also failed. It is possible that the implementation of the RS232 protocol in the Irisys MkII sensors doesn't comply with the standard. In order to overcome this problem, two Ethernet Serial Servers (Figure 4.3) were purchased thanks to a Faculty grant. These were placed above the ceiling tiles and connected to the sensors' RS232 outputs. These use the IETF RFC2217 [77] protocol to convert RS232 signals to TCP/IP packets.

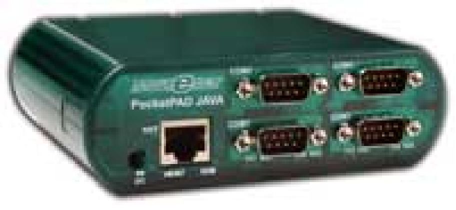


Figure 4.3: PocketPad Java Ethernet Serial server.

The Ethernet outputs from the servers ran above the ceiling tiles to a computing laboratory in B56. There they were connected to a private Ethernet network along with a general purpose PC that recorded the data returned by the sensors. This is shown in Figure 4.4. Given that only three elements were part of the network, latency was negligible.

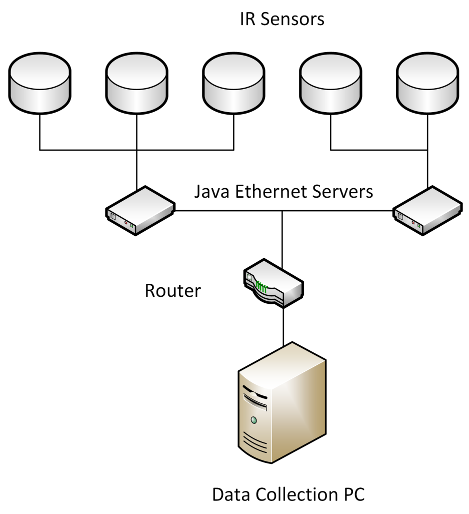


Figure 4.4: Data gathering infrastructure.

A webcam was also available, although this was not used. It had a fixed IP address on the University's network and it wasn't possible to connect it to the private network. It also had a narrow angle lens that meant that its coverage of the corridor wasn't complete enough to make it useful. Since the price of webcams, and IP enabled webcams specifically, has dropped considerably in the intervening time, the inclusion of video capabilities would be desirable. This is explored further in Section 8.2.1.1.

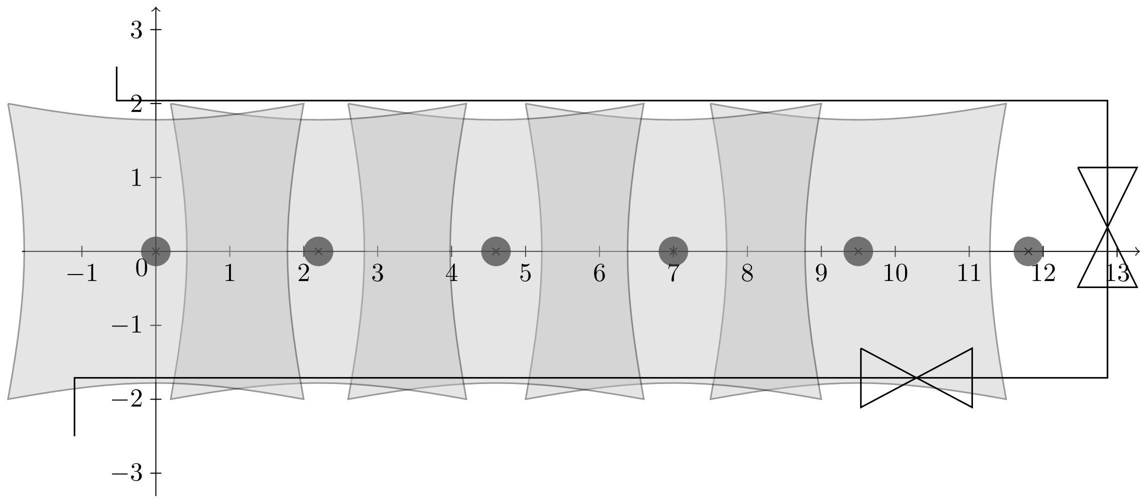


Figure 4.5: Plan of corridor showing the sensors (dark grey circles) and their field of view (light grey).

The sensors were installed in the configuration shown in Figure 4.5 in order to create overlaps in their field of view. The purpose of this was to make it possible to match co-occuring trajectories when one pedestrian was entering the field of view of one sensor and entering that of the next. As can be seen in Figure 4.5, the overlaps were generally just a bit more than 1 m at their narrowest points. The leftmost sensor was placed slightly closer to the other sensors with a 2.2 m separation from the next sensors. The other sensors were place 2.4 m apart.

The overlap between the sensors wasn't sufficient to recover data from an adjacent sensor when errors occurred. There were a number of causes for these, particularly intense sunlight shining through the glass wall of the corridor. The addition of more sensors, however, would have exacerbated existing communication bandwidth problems.

4.4 Data Gathering Software

The software written to record data was written entirely in Java. Although, as interpreted language, Java is not necessarily suitable for real-time programming. It is usually overlooked in favour of languages such as C++ that compile to machine code for real-time applications. These limitations were helped by the use of a parallel programming framework called JCSP [78, 79]. In addition, the intervals at which the sensors returned data was between 30 ms and 150 ms, with an average of 33 ms, a slow enough pace of communication for Java to handle. Initial efforts to use Java and the JCSP API for this purpose are described in [80] and were found to be successful. Lightweight processes were able to read data from four serial ports connected to sensors concurrently with little or no loss of data.

With the deployment of Ethernet serial servers, the burden of servicing serial port hardware interrupts were entirely removed from the data gathering PC. However, network communications are often handled with coalesced interrupts. Although these communications weren't hampered by 16 B buffers like serial ports, it became clear that the servicing of the coalesced interrupts could occasionally cause problems with synchronising inputs from the virtual serial ports. Timestamps were only added after the byte stream from the ports were processed by the data gathering software. No delays caused by the network and the time taken for servicing the virtual serial ports' coalesced interrupts were compensated for. The sensors' had no internal clocks and the Ethernet servers did not support the Network Time Protocol, so it was impossible to assess how significant these delays were, if at all.

The use of serial over Ethernet did however solve the problem of cable length, and the processing of five inputs in parallel, solving some of the problems mentioned in [80].

The data collected was recorded in text files, in comma separated values (CSV) format, easily readable by spreadsheet programs. The byte streams from the sensors were decoded, and then written to file as text. Information was recorded in the same form and order as it was received from the virtual serial ports. In this way, if more than one target was in the sensors' field of view, target (pedestrian) data was interleaved.

An example of this data is shown Figure 4.6. The identifiers of two separate targets are interleaved. The order of the fields is as followed: target identifier, parent target identifier, sensor status, x coordinate, y coordinate, fields that contain proprietary information, and a Java timestamp.

The data collection software did nothing other than record data. In part this was because the behaviour of the sensors used together in this way wasn't yet understood, and could only be determined from studying the data. Secondly, given the amount of communication the PC had to handle, it was important to reduce any overhead.

One of the most pernicious problems with the sensors, particularly related to limited communications ability was the fact that when too many people were in the field of view, data became increasingly corrupted. These errors were impossible to mitigate due the naïve RS232 protocol implemented in the sensors. Data was overwritten in the sensors' own buffers before it could be transmitted, breaking the structure of the data frames they used to transmit results. Unfortunately, there was no way to reliably record data when more than two people were in the field of view, severely limiting the applicability of the system. However, the data that was collected was consistent with findings in the literature, as explored in Chapter 7.

```
847,0,261,13.71875,  0.89108276,...,1255396287796
848,0,259,0.51336765,9.037109,.....,1255396287812
847,0,261,13.585449, 0.8413391,....,1255396287828
848,0,257,0.4571495, 9.029297,.....,1255396287843
```

FIGURE 4.6: Example of the data recorded in data files.

4.5 Conclusion

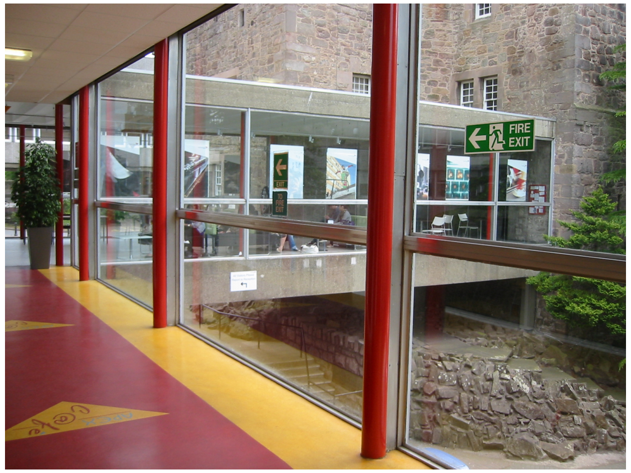
In this chapter, the location of the study and the infrastructure used to collect data were described. The system ran for 99 days, unattended. In this time, more that 5 GB of data was collected. A number of challenges had to be overcome to achieve this: parsing of the byte stream returned by the sensors, the shortcomings of the RS232 protocol and its implementation in the sensors and the recording in parallel of data from five sensors at the same time. These efforts were mostly successful.

However, since, Irisys have released two further models of the sensor, both equipped with Ethernet and WiFi capabilities. In addition, other IR sensors have come on the market. While much was learned in setting up the data collection system, many of the problems addressed here have been addressed by these new products.

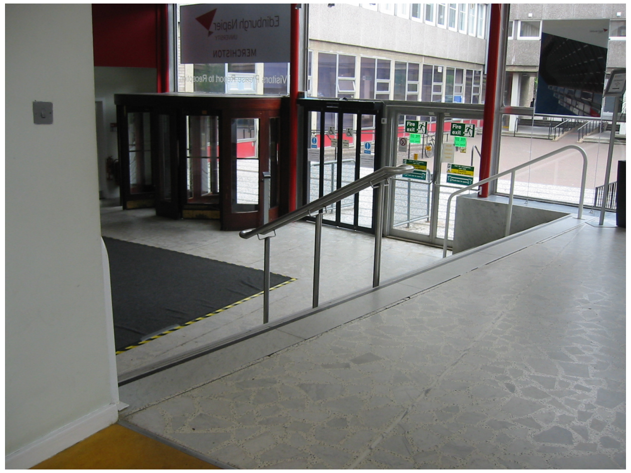
In Chapters 5 and 6, the processing of the collected data is described.

(a) Looking down the corridor towards the foyer.

(b) Looking up the corridor towards the computing laboratory in B56.



(c) Looking through the glass wall towards the perpendicular glass wall of the foyer.



(d) Stairs at the end of the corridor towards the left.

FIGURE 4.7: Location of the study.

Chapter 5

B-splines

5.1 Introduction

The infrastructure described in previous chapters ran successfully for 99 days. This resulted in the accumulation of 5 Gb of data. The challenge then was to process the data in such a way as to make it tractable. Initially, the data was loaded into a database cluster, which contained 27 million rows. However this was not helpful in filtering or classifying the collected trajectories. A different approach, inspired by Sillito and Fischer in [81, 82] was taken, an example of which is shown in Figure 5.1. Trajectories were converted into splines, specifically B-splines, allowing each of them to be described by a fixed length vector of control points. Data reduction techniques (also called "trajectory compression" in GIS) are described in more detail in the next chapter. Once the data was processed in this way, many of the previous problems of storage and retrieval were solved, allowing the greater part of the data to be stored in a single database instance on a standard desktop computer.

A second reason for transforming data into B-splines is noise reduction. A number of methods of noise reduction are used in signal processing, for example moving averages and the Savitzky-Golay filter. B-splines are also used for this and various other purposes in signal processing. These uses are described in detail by Unser et al. in [83, 84].

B-splines are parametric curves, that allow the description of Cartesian curve coordinates according to the parameter t. This makes them uniquely suited to describing spatio-temporal data, by using this parameter as time (in a more general sense, the parameter to the curve may represent any dimension). In [82], a comparison is made of various parametric representations of trajectories. They found that splines had "greater fidelity to the original trajectory" than the other methods used. These were the Discrete Fourier Transform, Chebyshev polynomial approximation, and the Haar wavelet transform. All of these provided the possibility of fixed dimensionality in storing and processing trajectories. However, given the fact that B-splines coefficients are calculated using least-squares approximation, they provided the best accuracy in experiments with various datasets in [82], measured by the sum-of-errors-squared.

B-splines are not the only parametric splines available. Other types of widely used splines are Bézier and Catmull-Rom splines. The former is used widely in graphics software in a non-parametric form (including the standard Java graphics API). Neither of these provide parametric continuity greater than C^1 and do not preserve information about higher order derivatives. However, B-splines do. This is explained further in Section 5.3.1.

B-splines are highly flexible piecewise polynomials used widely in the field of Computer Aided Geometric Design (CAGD). The history of their development, and the relatively recent development of the field of CAGD, is briefly given in Section 5.2. They are used in a diverse array of fields other than CAGD. A few examples are animation, kinematics, robotics and medical imaging.

The elements of B-splines are explained in Section 5.3, and their underlying mathematical principles are discussed in Section 5.4. Finally the Cox, de Boor, Mansfield

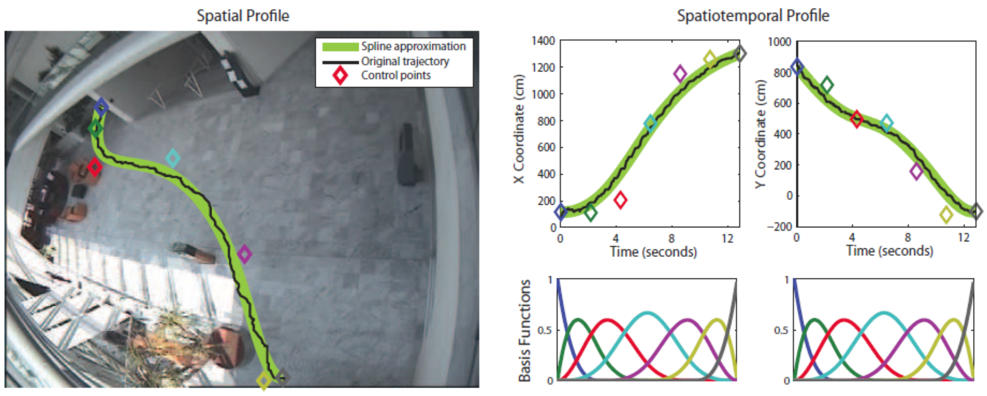
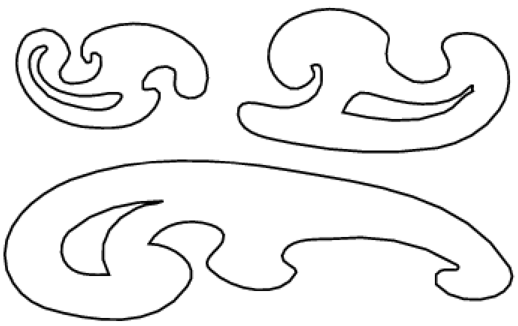


FIGURE 5.1: Example of a B-spline used to approximate a trajectory taken from the CAVIAR [85] dataset by Sillito and Fisher in [81]. The original trajectory (black) is superimposed on the spline in the left of the picture. The basis functions of the spline are shown in the bottom right.

algorithm, used in their calculation is given in Section 5.5. A worked example in Section 5.6 explains the steps involved in calculating splines.

5.2 History


(a) Mathematically derived French Curves [86].

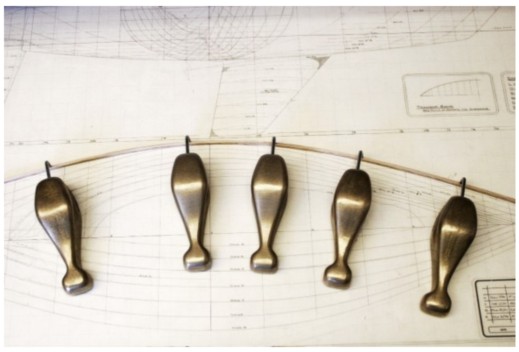

(b) Ducks used to create splines in ship design [87].

FIGURE 5.2: Classic design tools based on curves and splines.

The term Computer Aided Geometric Design was coined in 1974 by researchers Barnhill and Riesenfeld. In practice, mathematical principles have been used in ship building since Roman times. According to Farin:

> "These techniques were perfected by the Venetians from the 13th century to the 16th century. The form of the ribs was defined in terms

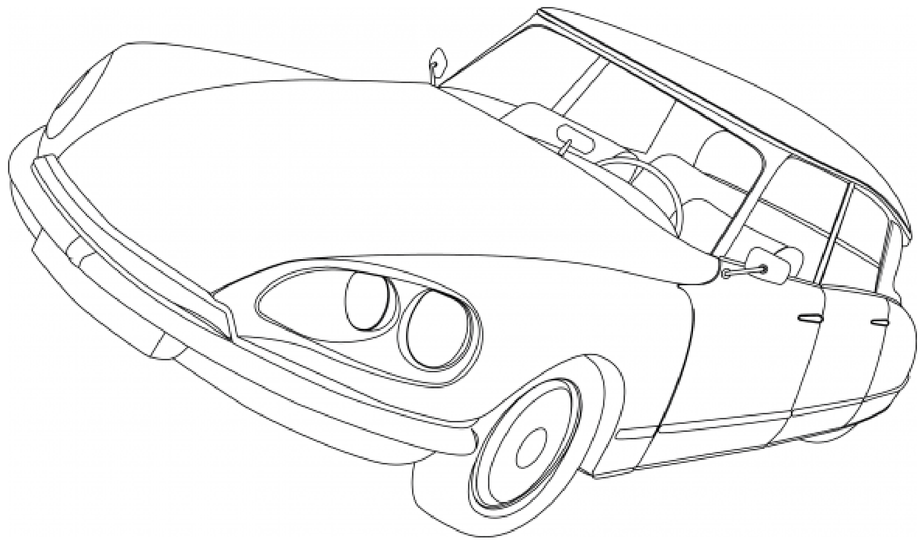


FIGURE 5.3: Citroën DS, also known as déese. Taken from [87].

> of tangent continuous circular arcs - NURBS in modern parlance. The ship's hull was obtained by varying the ribs' shapes along the keel, an early manifestation of today's tensor product surface definitions." [88]

A remarkable consequence of this was that the Venetians did not use design drawings. These were adopted by English shipbuilders from the 17th century. Splines were developed at that time. Originally, these were flexible pieces of wood that had "ducks", metal weights, attached to them along their length to define their curvature, see Figure 5.2(b).

As this historical background suggests, the importance of what would now be called CAGD grew first from the ship building industry and, since the 20th century, from aeronautical and automotive design. In the second half of the 20th century, advances were driven by the use of computers in these fields. This was to address the "blueprints to manufacturing" problem.

To illustrate this central problem in manufacturing and design, Townsend [87] gives the example of the Citroën DS (see Figure 5.3). The extremely popular and iconic car was originally designed by an aeronautical engineer, André Lefèbvre (who pioneered the use of wind tunnels in automotive design), in collaboration with the Italian sculptor, Flaminio Bertoni. However, the clay models used to design the shell had to be translated into blueprints by hand, creating a great deal of imprecision in the process. At the same time, numerical control (NC) milling machines, that created the stamps and dies used on the production line, were becoming available. The first car produced by Citroën entirely described mathematically was the Citroën GS. Although not as aesthetically pleasing as the DS, it reflected the design and manufacturing realities of the time and, in more sophisticated form, to this day.

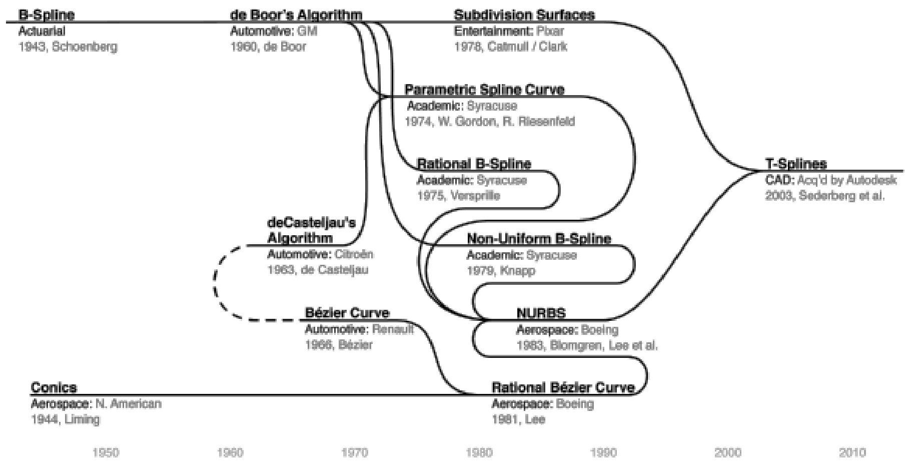


FIGURE 5.4: History of the development of CAGD. Taken from [87].

Various parallel developments occurred in the later half of the 20th century, originating in industry. A simplified overview of these are shown in Figure 5.4, taken from [87], where the influence various researchers have had on each other are shown. Liming, employed by North American Aviation (NAA), that produced the Mustang fighter for the American Army and the Royal Air Force during WWII, found that blueprints could be entirely eliminated by the use of conics. The parts of an aeroplane could be entirely described algorithmically. The need for such solution was driven in part by the need for secrecy, designs could be stored as tables of numbers, as well as the massive wartime armaments production effort. Conics have been used for a long time in design. French curves, shown in Figure 5.2(a), are an example of the use of conics. Their development is attributed to the mathematician and geometer Ludwig Burmester [89].

Although the mathematical principles behind conics are different from splines, they both addressed the same need, the replacement of blueprints by more accurate geometric representations. The first method of computing splines was developed by de Castlejau at Citroën in 1959. De Castlejau was prevented from publishing his method by his employers, until the 1980s. Bézier, the head of design at Citroën's competitor Rénault, set his team the task of independently recreating de Castlejau's work during the 1960s. While de Castlejau was kept silent, Bézier was allowed to publish widely. Originally developed by de Castlejau, these mathematically described curves became widely known as Bézier curves.

B-splines were originally developed by Schoenberg in 1946 for use with actuarial data, parametrised by time. Later it was discovered that Bézier curves are actually a special case of B-splines and the same methods could be used in their calculation. The main proponent of B-splines was Carl de Boor, who worked at General Motors, and co-discovered the algorithm used in their evaluation. He championed the use of B-splines in approximation theory. The method he used was also independently discovered by Mansfield and Cox. The algorithm, later described in Section 5.5, is named after all three.

In 1974, Riesenfeld introduced parametric spline curves. In the process he discovered that the B-spline algorithm was "the natural generalization of the de Castlejau algorithm" [88], and that Bézier curves were a subset of B-splines. Hence B-splines superseded Bézier curves in early CAD/CAM software.

The main focus of advances in CAGD since that time have been in the algorithmic description of surfaces, made possible by Riesenfeld's work. These include non-uniform rational B-splines (NURBS). This is outside the scope of this work, although details can be found in [90].

5.3 Elements of B-splines

A B-spline is described by three elements, its degree and order, the nature of the knot vector and the number of control points used in its calculation. These are described in turn below. This is then followed by the equation used to recursively calculate the basis functions of the spline, and the use of these results in approximating the least-squares fit of the curve with the data provided.

In order to avoid confusion, the purpose of the variables used is defined in Table 5.1. These are intended to be consistent with [91]. The only difference is that De Boor names the parameter in calculations x. In order to avoid confusion with Cartesian coordinates, here the parameter is named t.

N	the total number of data points
k	the order of the spline
n	the number of control points
$\boldsymbol{t}$	the knot vector
t_i	the ith element of the knot vector
t	a random variable of the same dimension as the knot vector, i.e. the parameter of the spline
$S(t)$	denotes the spline curve, parametrised by t
$B_{i,k}(t)$	the i^{th} basis function, of order k, at time t
$\boldsymbol{\alpha}$	the coefficients of the spline.
$\mathbf{D}$	the matrix of x, y and t coordinates of the data to be approximated, of length N
$\mathbf{L}$	the basis matrix whose elements are calculated by the basis functions and is $N \times n$ in size

Table 5.1: Variables used in B-spline calculations.

5.3.1 Order of a B-spline and continuity

The degree of a curve is a well defined concept (degree zero is a constant, degree one is linear, degree two is quadratic, degree three is cubic...etc). However, the order of a curve is a less well defined concept in mathematics. Often this is treated as synonymous with the degree of a curve. However in this chapter, the definition of the order of a curve is taken from [91, p. 1]: the order of the curve is equal to its degree plus one. Therefore a cubic curve will have an order of four and a degree of three.

B-splines are piecewise polynomials. According to the definitions given in [91], every piecewise polynomial that makes up the resulting curve is guaranteed to have an order that is one less than that of the overall curve. The properties of the resulting curve at the join points (i.e. the last point of a given polynomial and the first point of the next) also need to be considered. Geometric continuity is denoted by G^d and parametric continuity is denoted by C^d. The properties of each degree of continuity are explained in Table 5.2 below.

Geometric continuity implies that the geometry is continuous. Likewise, parametric continuity implies the underlying parametrisation, in this case t, is continuous. Parametric continuity of order k implies geometric continuity of order k but not vice-versa. Therefore parametric continuity is considered.

	Geometric continuity
G^0	the curves are joined
G^1	the first derivatives are proportional at the join point
	the curve tangents have the same direction if not the same magnitude
G^2	the first and second derivatives are proportional at the the join point
	Parametric continuity
C^0	the curves are joined
C^1	the first derivatives are equal
C^2	the first and second derivatives are equal, acceleration is continuous
C^n	the nth derivatives are equal

Table 5.2: Geometric and parametric continuity at the joint points for piecewise polynomials.

B-splines exhibit C^{k-2} continuity. The order k can be chosen according to the desired degree of continuity of the spline, as well as how closely it fits the data. For instance in order to obtain the continuity of velocities and accelerations at the time t, where the transition between two consecutive segments occurs, it is possible to assume a B-spline of order at least 4 or above. This has C^2 continuity, allowing the consistent and accurate calculation of the first and second derivatives, velocity and acceleration, throughout the resulting curve.

5.3.2 The knot vector

A uniform knot vector means that its values increase in uniform steps. This creates a knot vector $\boldsymbol{t}$ with these elements: $\boldsymbol{t} = \{0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1\}$. This is equivalent to the vector $\boldsymbol{t} = \{0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4\}$, given the increments stay proportionally the same, however there are computational benefits to normalising the knot vector. More floating points numbers are available between 0 to 1 than if the floating point number has an exponent.

A clamped uniform knot vector interpolates its first and last control points. Normally there is no guarantee that a B-spline will interpolate its control points, unlike Bézier splines. Where knots are repeated $k - 1$ times, the B-spline is guaranteed to pass through that point (the continuity is C^0). When the knots are repeated k times, the B-spline becomes discontinuous at this point (the continuity is C^{-1}) [91, p. 91-92]. In clamped uniform knot vectors, the first and last points have a multiplicity of k, meaning the B-spline only has local support in the interval $[0, 1]$ of t. In this application, this has the advantage that nothing is added to the information recorded outside its time range.

5.3.3 Number of control points

The number of control points is arbitrary, but influences the length of the knot vector. Clamped uniform knot vectors must be at least of length $2k$ to accommodate the multiple knots at the beginning and end of the knot vector. However this means that there are no internal knots. As stated in Section 5.2, B-splines were found to be generalisations of Bézier curves: a B-spline with no internal knots in its knot vector is a Bézier curve.

The knot vector is of length $k + n$. The number of internal knots is given by $n - k$. In other words increasing the number of control points will increase the length of the knot vector. In this research, given a similar application as [81], the same number of control points (7) has been chosen. Using the formulas above, as the order of the spline is 4 with 7 control points, this leads to a knot vector of length 11 with 3 internal knots.

5.4 B-spline evaluation

In the previous section, the three main elements of B-splines were described, their order, knot sequence and number of control points. In fact, a spline space $\$_{k,\boldsymbol{t}}$ contains all possible splines with the same elements. Although Sillito and Fischer repeat the diagram of the basis functions in the bottom right of Figure 5.1, the shape of the basis functions created from the same above three elements will not change for any possible value of the coefficients of the spline. The definition is given by de Boor [91, p. 93] is:

$$\$_{k,\mathbf{t}} = \left\{ \sum_i \alpha_i B_{i,k,\boldsymbol{t}} : \alpha_i \text{ real, all } i \right\}. \tag{5.1}$$

Where α_i are the coefficients (control points) used in the construction of the resulting curve, $B_{i,k,\mathbf{t}}$ is the ith basis function. The term $B_{i,k,\mathbf{t}}$ is simplified to $B_{i,k}$ throughout the rest of this chapter. The knot vector $\mathbf{t}$ is implied.

5.4.1 Basis functions

The term B-spline is an abbreviation of basis spline. The spline is made up of basis functions shown in Figure 5.5. Each control point depends on its basis function. Given that $n = 7$ here, there are seven basis functions, each valid over a certain interval in the knot vector. Where there is knot multiplicity, here at the beginning and end of the knot vector, the relevant basis function has a value of one.

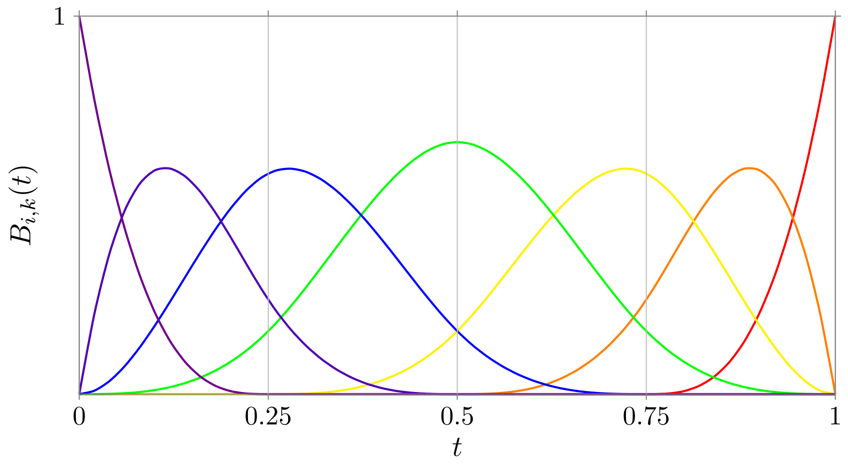


FIGURE 5.5: Basis functions of splines in the spline space $\$_{k,\mathbf{t}}$, where the order $k = 4$ and the clamped uniform knot vector $\mathbf{t} = \{0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1\}$.

The recurrent relation formula for the basis functions used to create Figure 5.5 is given in Equation (5.2). Each i represents a knot interval, and the function is non zero in the interval $[t_i, t_{i+1}]$. The order is represented by k.

$$
\begin{aligned}
B_{i,1}(t) &= \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1} \\ 0 & \quad \text{otherwise.} \end{cases} \\
B_{i,k}(t) &= \frac{t - t_i}{t_{i+k-1} - t_i} B_{i,k-1} + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} B_{i+1,k-1}
\end{aligned}
\tag{5.2}
$$

Equation (5.2) is used to construct a triangular array shown in Figure 5.6. Moving from left to right, the order of the basis functions evaluated increase by one until this is equal to the order of the spline. For basis functions of order 1, the value returned is either zero or one depending on whether t is in the interval $[t_i, t_{i+1}]$.

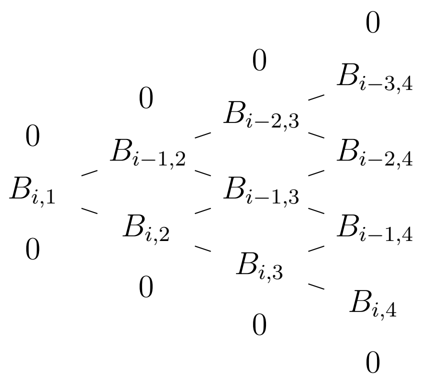


FIGURE 5.6: The triangular array of B-splines of order $\leq k$ ($k = 4$) that are nonzero on $[t_i, t_{i+k-1}]$.

The recurrent relation in Equation (5.2) is used to populate an $N \times n$ matrix $\mathbf{L}$, where:

$$L_{r,i} = B_{i,k}(t_r) \text{ where } 0 \leq r < N \tag{5.3}$$

In this equation, each t_r represents an element of the array $\mathbf{D}$. The basis matrix $\mathbf{L}$ is then used in the calculation of the control points, explained in the next section.

5.4.2 Approximation of data

Systems of simultaneous equations can be expressed in matrix form. Given $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} = \mathbf{bA}^{-1}$. However this requires that $\mathbf{A}$ be a square matrix, with full rank. In all other cases, the Moore-Penrose pseudoinverse matrix operator can be used. If the properties of the pseudoinverse are met, this guarantees a unique solution to the system of simultaneous equations, that is also the least-squares solution.

The Moore-Penrose pseudoinverse operator is defined as: $\mathbf{L}^{+} = (\mathbf{L}^T\mathbf{L})^{-1}\mathbf{L}^T$. This on its own has a time complexity of $O(mn^2) + O(n^3)$ [92] for an $m \times n$ matrix. However the actual implementation of the pseudoinverse operator depends on the library used, in this case JAMA [93], and any included optimisations. Reducing the complexity of the operator is a very active area of research, given its many applications, that are not limited to least-squares approximation.

The coordinates of the coefficients α can be approximated by multiplying the original data by the inverted matrix $\mathbf{L}^{+}$, populated with the values of the basis functions in Equation (5.3), such that:

$$\boldsymbol{\alpha} \approx \mathbf{L}^{+}\mathbf{D} \tag{5.4}$$

5.4.3 Resulting spline

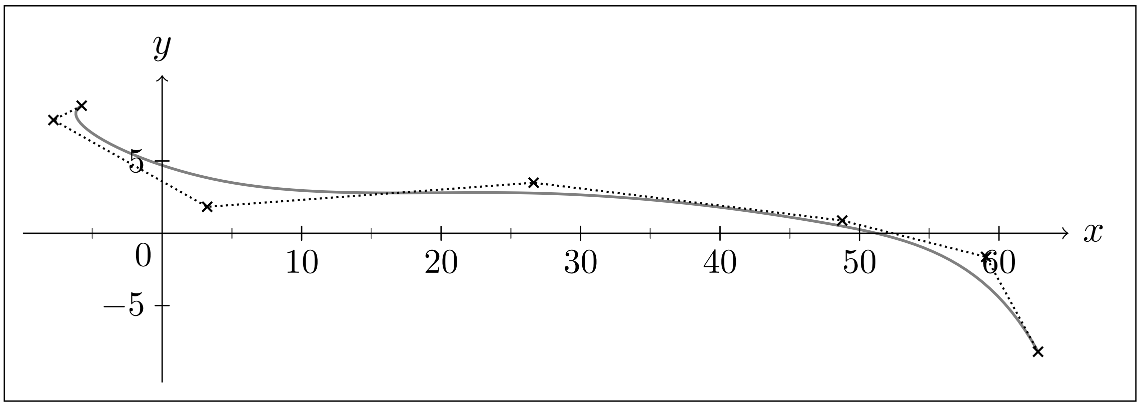
The resulting spline is calculated using Equation (5.5). Each point on the curve for an arbitrary value of the parameter t can be found using this equation. The resulting x and y are found by first calculating the basis functions for t, multiplying this by the control point associated with that basis function, and repeating this for as many control points as there are.

$$S(t) = \left\{ \sum_{i=1}^{n} \alpha_i B_{i,k}(t) \right\} \tag{5.5}$$

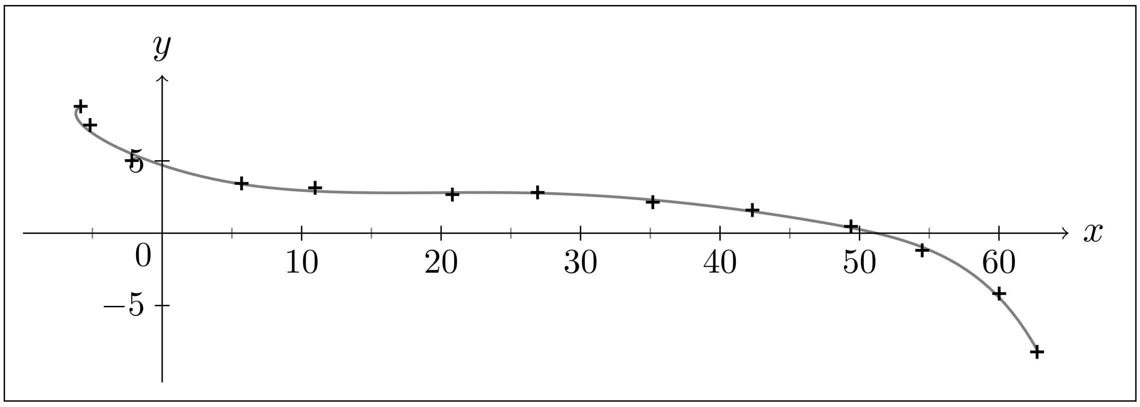
If the parameter t falls within a knot interval where a basis function is defined, as is shown in Figure 5.5, basis functions often overlap over intervals, then a non zero value is returned to the summation.

Figure 5.7(a) shows a B-spline within its control polygon with its control points marked. The spline is constructed from arbitrary values of t that fall with the interval $[0, 1]$.

In Figure 5.7(b), the original data points are shown, along with the spline that approximates the original data. These figures also demonstrated the local support property of B-splines with clamped knot vectors. The spline is not defined outside the time range of the data points, and interpolates the first and last points.



(a) A B-spline shown with its control points ($\times$) and its control polygon (dotted line).



(b) A B-spline shown with the data points that it approximates ($+$).

FIGURE 5.7: The same B-spline (grey) showing (a) its control points and control polygon, and (b) the data points it approximates.

5.5 Algorithm

Although B-splines were originally discovered in 1946 by Schoenberg, the algorithm to evaluate them in a numerically stable way wasn't discovered until the early 1970s. This discovery was made independently by Cox, de Boor and Mansfield. Hence the algorithm is named after all three.

The purpose of the algorithm is to evaluate the basis functions of the spline. All other calculations depend on this. Algorithm 5.1 represents a recreation of the algorithm given in [91, p. 110-111], and the FORTRAN77 procedure `BSPLVB` in [91, p. 111-112]. The purpose of this procedure is to evaluate a k length vector basis functions $B_{j,k}(t)$ where $j = i, \dots, i-k+1$, and i represents the knot interval in which the value of t being evaluated falls. In effect, this procedure creates the last, right-most column of the triangular array shown in Figure 5.6, by calculating the values in the preceding columns.

Key to the function is Equation 5.2 rewritten as:

$$b'_r = (t - t_{i-j+r-1})\frac{b_{r-1}}{t_{i+r-1} - t_{i-j+r-1}} + (t_{i+r} - t)\frac{b_r}{t_{i+r} - t_{i-j+r}} \tag{5.6}$$

where $j = 1, \ldots, k-1$ and $r = 1, \ldots, j+1$.

The following terms are then introduced:

$$\begin{aligned} \delta^L_{sj} &= t - t_{i-j+s}, \text{ and} \\ \delta^R_s &= t_{i+s} - t \end{aligned}$$

where $s = 1, \ldots, k-1$. The term δ^L_{rj} can be rewritten as $\delta^L_s = t - t_{i+1-s}$, independent of j. This is discussed in [91, p. 111], and is used in Line 6 of the algorithm. Equation (5.6) can then be rewritten as:

$$b'_r = \delta^L \frac{b_{r-1}}{\delta^L_{r-1,j} + \delta^R_{r-1}} + \delta^R_r \frac{b_r}{\delta^L_{r,j} + \delta^R_r} \tag{5.7}$$

where $r = 1, \ldots, j+1$.

In Algorithm 5.1, the inner `for` loop creates the entries in b_r, $B_{i-k+1,k}(t), \ldots, B_{i,k}(t)$, $r = 1, \ldots, j+1$, overwriting the previous values in b_r on line Line 13. The outer `for` loop repeats this process j times where $j = 1, \ldots, k-1$. In this way, at the jth iteration, the $j+1$ or kth basis function values are populated in the array b_r and then returned by the function.

The full listing of the Java code used is included in Appendix A. In the next section, a complete worked example is provided to further explain calculating B-splines.

```
function BSPLVB($i, k, t$)
        ▷ Where $i$ is the knot interval, $k$ is the order and $t$ the point at which the B-splines are to be evaluated.
    $b_1 = 1$
    for $j = 1, \dots, k-1$ do
        $\delta^R_j = t_{i+j} - t$
        $\delta^L_j = t - t_{i+1-j}$
        saved $= 0$
        for $r = 1, \dots, j$ do
            term $= \frac{b_r}{\delta^L_r + \delta^R_{j+1-r}}$
            $b_r =$ saved $+ \delta^R_r \cdot$ term
            saved $= \delta^L_{j+1-r} \cdot$ term
        end for
        $b_{j+1} =$ saved
    end for
return $b_r$
end function
```

Algorithm 5.1: Pseudocode for the function that evaluating B-spline basis functions.

5.6 Worked Example

The purpose of this section is to provide a worked example of calculating a B-spline. In Section 5.6.1, the calculation of the control points, or coefficients is shown. In Section 5.6.2, the reconstruction of the spline from the control points, without reference to the original data, is explained. To avoid confusion, variable names in Section 5.6.2 are appended with prime notation, where variable have the same purpose in Section 5.6.1.

5.6.1 Approximation

The data used to create B-splines in Figure 5.7 is given in the 13×3 matrix **D**. The columns of the matrix denote x, y and t. The values in t are normalised. Originally recorded as Java timestamp values (64-bit long integers), these are are normalised by first subtracting the start time from all values of t and then dividing them by the end time. This guarantees that all values of t are in the interval $[0, 1]$.

$$\mathbf{D} = \begin{pmatrix} -5.8431 & 8.7705 & 0.0000 \\ -5.1671 & 7.4771 & 0.0843 \\ -2.1750 & 5.0201 & 0.1686 \\ 5.6943 & 3.4427 & 0.2557 \\ 10.9686 & 3.1375 & 0.3413 \\ 20.8084 & 2.6694 & 0.4270 \\ 26.9189 & 2.8203 & 0.5113 \\ 35.1799 & 2.1430 & 0.5970 \\ 42.3187 & 1.5988 & 0.6840 \\ 49.3911 & 0.4705 & 0.7683 \\ 54.4946 & -1.1856 & 0.8540 \\ 60.0164 & -4.1736 & 0.9384 \\ 62.7329 & -8.2067 & 1.0000 \end{pmatrix}$$

The first step in approximating the data in **D** is by calculating the basis matrix **L**.

```
1: function BUILDBASISMATRIX( )
2:     for r=1,...,N do
3:         $t = D_{r,t}$
4:         $i =$ INTERVAL$(t)$
5:         $b_r =$ BSPLVB$(i, k, t)$
6:         for $j = 1, \dots, k$ do                ▷ Add the values $br$ to the basis matrix L.
7:             $L_{r,i} = b_{i-j+k}$
8:         end for
9:     end for
10: return L
11: end function
```

$$\mathbf{L} = \begin{pmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.2912 & 0.5670 & 0.1354 & 0.0064 & 0.0000 & 0.0000 & 0.0000 \\ 0.0345 & 0.5133 & 0.4011 & 0.0511 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.2333 & 0.5884 & 0.1783 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0640 & 0.5363 & 0.3916 & 0.0081 & 0.0000 & 0.0000 \\ 0.0000 & 0.0062 & 0.3408 & 0.5939 & 0.0591 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.1451 & 0.6647 & 0.1902 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0382 & 0.5453 & 0.4019 & 0.0146 & 0.0000 \\ 0.0000 & 0.0000 & 0.0031 & 0.3243 & 0.5729 & 0.0997 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.1327 & 0.5587 & 0.3082 & 0.0004 \\ 0.0000 & 0.0000 & 0.0000 & 0.0332 & 0.3290 & 0.5658 & 0.0720 \\ 0.0000 & 0.0000 & 0.0000 & 0.0025 & 0.0774 & 0.4922 & 0.4280 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{pmatrix}$$

The results of this are shown above (rounded to 4 d.p.). This shows that **L** is an $N \times p$ matrix. Given the B-spline has a clamped uniform knot vector, the first and last elements of the array are 1. In order to find the coefficients $\boldsymbol{\alpha}$ of the B-spline, the Penrose-Moore pseudoinverse of the basis matrix, $\mathbf{L}^+$, is multiplied by the data values to be approximated held in **D**.

$$\boldsymbol{\alpha} = \mathbf{D} \cdot \mathbf{L}^+$$

$$\boldsymbol{\alpha} = \begin{pmatrix} -5.7698 & 8.8211 & -0.0000 \\ -7.7966 & 7.8284 & 0.0833 \\ 3.2283 & 1.8305 & 0.2500 \\ 26.6352 & 3.4950 & 0.5000 \\ 48.7602 & 0.8798 & 0.7500 \\ 59.0750 & -1.6322 & 0.9167 \\ 62.8021 & -8.1879 & 1.0000 \end{pmatrix}$$

The coefficient matrix $\boldsymbol{\alpha}$, a $n \times 3$ matrix, now contains the n coordinates of the control points. In the next subsection, the reconstruction of the curve using the n control points rather than the N data points is explained.

5.6.2 Reconstructing the trajectory from control points

Reconstructing the trajectory from the control points starts with the creation of the basis matrix according to the desired granularity. This affects how smooth the resulting spline is. In Figure 5.7, the curves were created using $N' + 1$ (including the zeroth and the last) x, y and t coordinates, $N' = 100$. In this example, for the sake of brevity, $N' = 10$ (or 11 coordinates) will be calculated.

The values of t used in recreating a curve can be entirely arbitrary, so long as they are increasing. Here, each successive value of t is incremented by $\frac{1}{N'}$. The values of t are passed to the BSPLVB function, to create the basis matrix $\mathbf{L}'$.

$$\mathbf{L}' = \begin{pmatrix}
1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.2160 & 0.5920 & 0.1813 & 0.0107 & 0.0000 & 0.0000 & 0.0000 \\
0.0080 & 0.4160 & 0.4907 & 0.0853 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.1280 & 0.5880 & 0.2827 & 0.0013 & 0.0000 & 0.0000 \\
0.0000 & 0.0160 & 0.4093 & 0.5387 & 0.0360 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.1667 & 0.6667 & 0.1667 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0360 & 0.5387 & 0.4093 & 0.0160 & 0.0000 \\
0.0000 & 0.0000 & 0.0013 & 0.2827 & 0.5880 & 0.1280 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0853 & 0.4907 & 0.4160 & 0.0080 \\
0.0000 & 0.0000 & 0.0000 & 0.0107 & 0.1813 & 0.5920 & 0.2160 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000
\end{pmatrix}$$

In order to create a new matrix of coordinates, $\mathbf{D}'$, Equation 5.5 is used.

```
1: $x = 0$
2: $y = 0$
3: $t = 0$
4: for $i = 0, \dots, N' + 1$ do
5:     for $j = 1, \dots, n$ do
6:         $x = x + (\alpha_{i,x} \times L'_{i,j})$
7:         $y = y + (\alpha_{i,y} \times L'_{i,j})$
8:         $t = t + (\alpha_{i,t} \times L'_{i,j})$
9:     end for
10:    $D_{i,x} = x$
11:    $D_{i,y} = y$
12:    $D_{i,t} = t$
13: end for
```

$$\mathbf{D}' = \begin{pmatrix} -5.7698 & 8.8211 & 0.0 \\ -4.9923 & 6.9090 & 0.1 \\ 0.5674 & 4.5236 & 0.2 \\ 8.4942 & 3.0675 & 0.3 \\ 17.2996 & 2.7889 & 0.4 \\ 26.4216 & 2.7817 & 0.5 \\ 35.3681 & 2.2826 & 0.6 \\ 43.7658 & 1.2988 & 0.7 \\ 51.2755 & -0.0145 & 0.8 \\ 57.6636 & -2.5380 & 0.9 \\ 62.8021 & -8.1879 & 1.0 \end{pmatrix}$$

This populates the matrix $\mathbf{D}'$, from which the B-spline shown in Figure 5.8 is plotted as a dashed black line. This overlays a B-spline in grey, created using the same control points, but with 101 coordinates. Figure 5.8 shows the level of smoothing and detail possible, by increasing the number of coordinates calculated.

The rows of the matrix $\mathbf{D}'$ represent the x, y and t coordinates of a point on the curve.

Since one of the purposes of using splines is data compression, the spline can be stored as vector of control points with a fixed length. In this example, the original

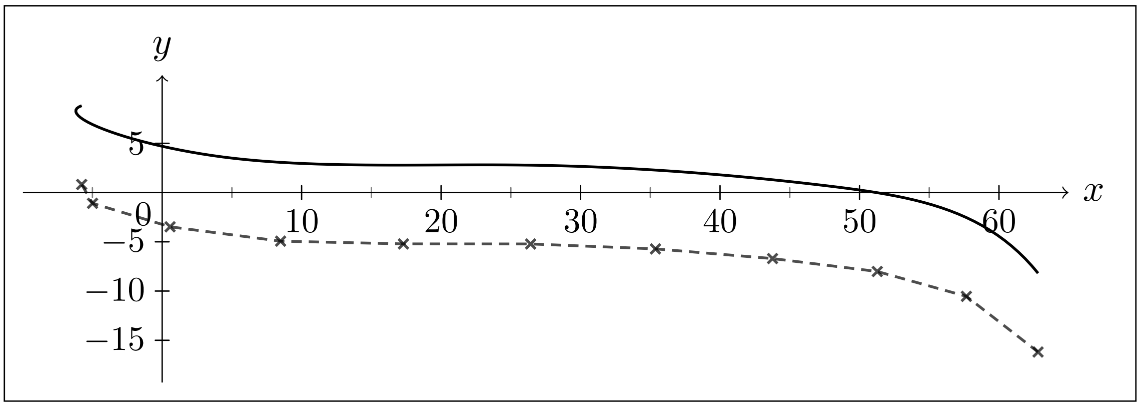


FIGURE 5.8: Example of spline smoothness: two splines created using different numbers of points. The top spline was created using $N = 100$, the bottom spline shown with the black dashed line was created using $N = 10$, its points marked by $(\times)$.

data is 13 points in length, while the coefficient vector is 7 points in length. The data reduction, in this case, is 53.85%.

5.7 Further properties of B-splines

The derivatives of B-splines can be calculated using the original B-spline. This produces another B-spline of one order lower. Derivatives can be calculated up to $k - 1$. For example, for a B-spline of order $k = 4$, up to the third derivative can be calculated.

These can be useful in this context, not just for calculating speed and acceleration across the length of the B-spline, but also for determining points of inflection. These are all equally important in pedestrian dynamics research. Inflection points are often called decision points in this context.

While speed and acceleration are given by the first and second derivatives, points of inflection require the third derivative to also be found when determining them. The conditions for determining if a point on a function is an inflection point are as follows [94, p. 231]:

- the tangent to $f(x)$ crosses the curve at x;

- the second derivative $f''(x) = 0$ or $f''(x)$ changes sign while traversing a point from left to right;
- the highest order odd derivative $f^{(k)} \neq 0$. Given B-splines of order $k = 4$ are being used, in this case this would mean the third derivative $f'''(x)$.

Derivatives can be calculated either numerically, using standard numerical methods, or analytically. The analytical means of calculation involves differencing the coefficients of a B-spline. Equation (5.8), taken from [91, p. 117 (15)], shows the m^{th} derivative of a B-spline is calculated from the coefficients of the B-spline one derivative higher multiplied by the basis function, $B_{j,k-m}$.

$$D^m \left(\Sigma_j \alpha_j B_{j,k}\right) = \Sigma_j \alpha_j^{(m+1)} B_{j,k-m} \tag{5.8}$$

Equation (5.9), taken from [91, p. 117 (16)], shows how the coefficients using in Equation (5.8) for the derivative function are calculated, where r is a dummy value denoting the interval and m is the derivative.

$$\alpha_r^{(m+1)} = \begin{cases} \alpha_r, & \text{for } m = 0 \\ \dfrac{\alpha_r^{(m)} - \alpha_{r-1}^{(m)}}{(t_{r+k-m} - t_r)/(k-m)}, & \text{for } m > 0 \end{cases} \tag{5.9}$$

The highest order derivative that can be calculated from a cubic B-spline ($k = 4$) is the third order ($k - 1 = 3$) derivative. This makes it possible to satisfy all the conditions given above to calculate the location of inflection points. In order to preserve all information relevant to pedestrian dynamics research, therefore, B-splines of order 4 or above must be used.

As mentioned in Section 5.3, the order of the spline and the number of control points influence the length of the knot vector. Once the order has been decided, the number of control points and therefore the length of the knot vector can be altered algorithmically. One such method of doing so is Schoenberg's variation diminishing spline approximation, explained in detail in [91, p. 141]. Using methods such as this allows for a better fit of the data.

5.8 Conclusion

In this chapter, basis splines (B-splines) have been introduced, including the fundamental concepts and evaluation procedures relevant to them. There are many different types of splines (Bézier, Catmull-Rom, ...etc). The particular benefit of B-splines is their mathematical sophistication, particularly the preservation of information to do with the first and second derivatives of the curve. An effort has also been made to provide a basic working understanding of the mathematical processes involved in constructing B-splines. While the mathematics involved in the formulation of B-splines are sophisticated, their computation is relatively straightforward, as shown in Section 5.5 and Section 5.6.

This is only a superficial examination the subject. Fuller explanations and mathematical proofs are given by de Boor in [91], as well as more advanced concepts such as knot insertion and non-uniform knot vectors.

Chapter 6

Data Processing

6.1 Introduction

As mentioned previously, over 99 days, 5 GB of data was collected. Initial attempts to process this data involved the use of a distributed MySQL database running on a Beowulf cluster. The entirety of the dataset included more than 30 million different rows representing 28 million data points. However, the limitations and unwieldy nature of the SQL programming language meant that a file based batch processing approach, using Java, was more practical.

The batch processing was carried out in a six step process: filtering (Section 6.2), conversion of coordinates from Euclidean to metric units (Section 6.3), track fusion (Section 6.4), conversion to B-splines (Section 6.5), classification (Section 6.6) and finally upload to a database (Section 6.7).

6.2 Filtering

The infrastructure described in Chapter 4 was designed to allowed unattended and automated collection of data. The software that processed the data returned by the

Error type	Description
Not enough files	Any data directories with less than five files in them were not processed any further.
Not enough data	Spurious trajectories often created data directories where only a single file contained any data. Additionally, strong sunlight could blind one or more of the sensors.
Too many targets	Trajectories were counted after spurious trajectories were removed. Any time intervals where there were more than two pedestrians in the field of view were not processed any further.
Invalid trajectories	Sometimes trajectories were lost by the sensor while they passed through, either because the pedestrian stood still for too long or because of sensor error. If a trajectory did not fully cross a sensor it was deemed invalid.
Matching error	When trajectories could not be matched.
Matrix rank deficient	When an error in creating the spline for a trajectory occured.

TABLE 6.1: Types of errors and their descriptions.

sensors merely recorded the data and a timestamp. As a result, a great deal of data was spurious in nature or contained sensor errors.

The format in which the data was recorded was determined by a “time out.” If the sensors were idle for more than 1 min, recording in the current of files ceased. As soon as the sensors became active again, a new set of files were opened, in a folder structure organised by date, hour, and then the start time of the interval.

In GIS, data returned from a query specified by a start and end time is often called a timeslice [95]. This is derived from the term slice in database programming, denoting the selection of data according to a condition. The data is selected according to time, hence the term timeslice, which will be used throughout this chapter.

Regrettably, the errors produced by the sensors meant that much of the data concerning situations where more than two pedestrians were in the corridor was corrupted. In order to prevent problems in further processing steps, timeslices with more than two pedestrians were excluded from this study. Possible efforts to mitigate these problems, and increase the applicability of the system, are discussed in Chapter 8.

Additionally there were a number of spurious trajectories the main cause of which were people drinking hot cups of coffee in the glass window of the Apex Café, just within sensor range. People walking in front of the corridor, across the foyer, also created trajectories. These spurious trajectories were simply removed before processing the data further.

6.3 Conversion to metric

The data returned by each sensor are all set to its own coordinate system. Although the manufacturer give guidelines as to the field of view covered by the sensors, these are approximate and vary according to the height at which they are mounted. In our case, the sensors are mounted at 4 m. According the manufacturer this means the field of is $4 \times 4\,\mathrm{m}$. The accuracy of this is untested.

In order to arrive at a figure, the maximum length of the x and y axes was calculated using the distortion correction algorithm, producing a value of 10.2528. The length of the "ground truth" estimated by the manufacturer was then divided by this value. Since the origin of the sensors' coordinate system was altered to record data relative to the centre of the lens, the point $(10.2528, 10.2528)$ in Euclidean coordinates represents $(2, 2)$ in metric units. Simply dividing $\frac{2}{10.2528}$ gave a result for the scaling factor of 0.195. Better estimates are not possible without a complete metrological assessment of the sensors.

6.4 Track Fusion

Conversion of trajectories a metric representation, per sensor, produces results such the trajectory shown in Figure 6.1. The sensors have been placed in the corridor in a such a way that their fields of view overlap. The purpose of this overlap is to allow track fusion between trajectories ending in one sensor's field of view and starting in the next at the same time. The end result is a complete trajectory throughout the length of the corridor.

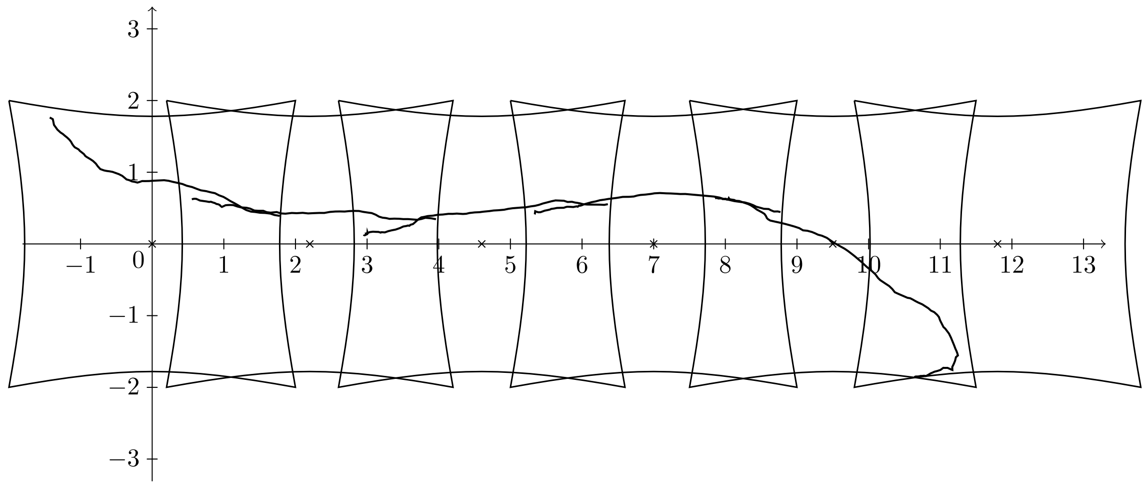


FIGURE 6.1: Example of a raw trajectory.

In Figure 6.1, the trajectory is moving from left to right. When the trajectory is leaving one sensor's field of view, its path remains relatively clear. However, when it is entering a sensor's field of view, its path is extremely imprecise and then converges towards the expected trajectory.

This creates a number of problems in matching trajectories from one sensor's field of view to another. However many of these challenges have been addressed in sensor fusion, and there are two main approaches in dealing with similar problems: RANdom SAmple Consensus (RANSAC) [96] and Kalman filters [97].

As this research was constrained by the sensors' limitations to dealing with only two trajectories at a time, a brute force approach was adopted. This is not a particularly efficient approach. In particular, it would not scale up well to a higher trajectory count. Inevitably this limitation reduces the applicability of the overall system. The brute force approach repeatedly measures the distances between cooccuring trajectories until the closest match is found. Other more flexible and effective approaches are detailed below.

Kalman filters assume that the errors in data are normally distributed. It is not known if this in the case with the sensors. Kalman filters do however have the advantage of working in real-time. RANSAC makes no assumptions about the data. It is a stochastic method that is passed a model against which it fits the data it is given in an effort to reduce the number of outliers. It continues iterating until a threshold value has been reached, returning the best parameters found, or it has run out of iterations.

There are many variants to both, and they aren't the only processes available that perform a similar function. However they are widely used in both sensor and track fusion. Track fusion was done using brute force techniques to combine trajectories from the sensors. This was not particularly flexible as it was only shown to work on a maximum of two cooccuring trajectories at a time. These are described in Sections 6.4.1 and 6.4.2.

6.4.1 Kalman filter

According to Faragher:

> "[The Kalman filter] is one of the most celebrated and popular data fusion algorithms in the field of information processing. The most famous early use of the Kalman filter was in the Apollo navigation computer that took Neil Armstrong to the moon, and (most importantly) brought him back. Today, Kalman filters are at work in every satellite navigation device, every smart phone, and many computer games." [97]

The equation for the Kalman filter is given in Equation (6.1). Its terms are explained in Table 6.2.

$$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t + \mathbf{w}_t \tag{6.1}$$

$\mathbf{x}_t$	the vector containing the values of the system being sought (eg. speed, position)
$\mathbf{F}_t$	the state transition matrix which is multiplied by $\mathbf{x}_{t-1}$ carrying forward the system state at $t-1$
$\mathbf{B}_t$	the control input matrix which applies the effect of the parameters in $\mathbf{u}_t$ on the state vector
$\mathbf{u}_t$	the vector containing any control inputs
$\mathbf{w}_t$	the vector containing the process noise terms for each parameter in the state vector

TABLE 6.2: Explanation of the terms of the Kalman filter in Equation (6.1).

It relies on the fact that the product of two Gaussian probability distributions is another Gaussian probability distribution. This is illustrated in Figure 6.2.

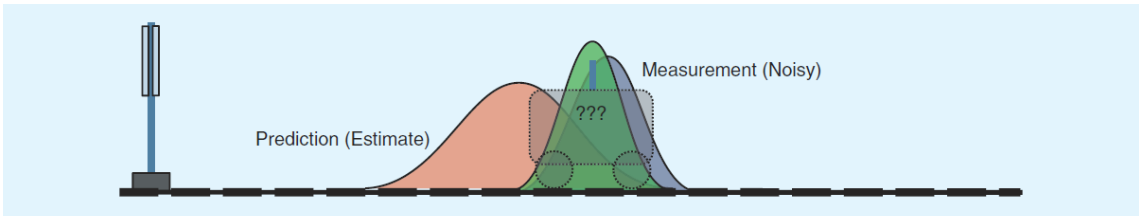


FIGURE 6.2: Estimate of the location of a train using a Kalman filter. Taken from [97].

In Figure 6.2, the red Gaussian pdf represents the predicted location, the blue pdf represents the noisy measurement of the location. The green pdf is the product of the prediction and measurement and represents the best estimate of the train's location.

Kalman filters are able to work in real-time and have many advantages. However, this method relies on the assumption that errors and noise in the system are normally distributed. It is unknown if this is the case for the Irisys sensors.

6.4.2 RANSAC

RANSAC is a stochastic method for fitting data to a predetermined model. Its purpose is to reduce the number of outliers in the data.

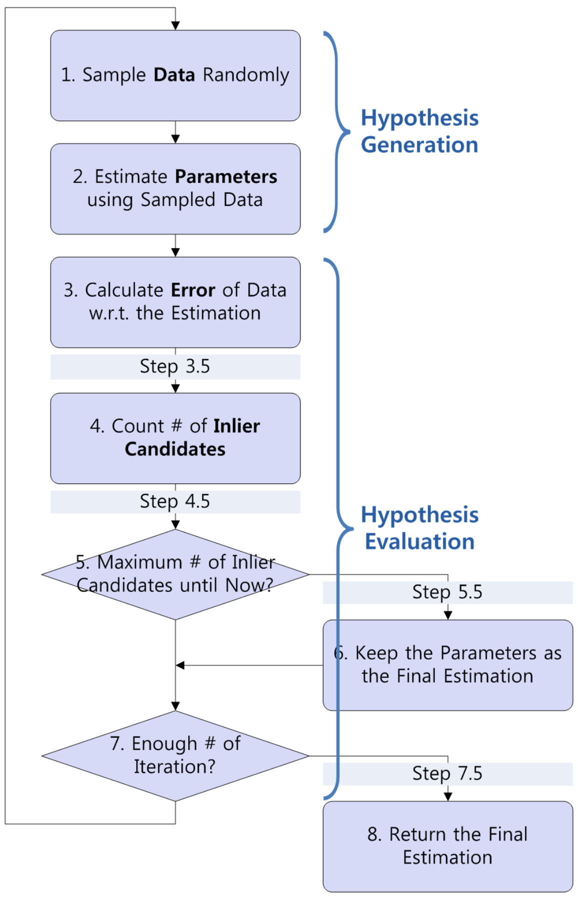

1. Sample **Data** Randomly
2. Estimate **Parameters** using Sampled Data
Hypothesis Generation
3. Calculate **Error** of Data w.r.t. the Estimation
Step 3.5
4. Count # of **Inlier Candidates**
Step 4.5
Hypothesis Evaluation
5. Maximum # of Inlier Candidates until Now?
Step 5.5
6. Keep the Parameters as the Final Estimation
7. Enough # of Iteration?
Step 7.5
8. Return the Final Estimation

FIGURE 6.3: Flowchart of the RANSAC algorithm. Taken from [96].

The basic algorithm is in two steps: hypothesis generation from random samples followed by verification with the data. This is illustrated in Figure 6.3. RANSAC makes no assumptions about the normal distribution of errors, like the Kalman filter does, and is far more flexible in model specification. However, it does not work in real-time like the Kalman filter.

As a stochastic method, there is no guarantee that the best value for the model parameters are arrived at, or even any. The threshold values for the number of inliers that fit to the model given a set of parameters, as well as the number of iterations, are chosen arbitrarily. However, as Choi et al. [96] point out, similar approaches that map data to a parameter space, such as Hough transforms, are extremely costly in terms of computing resources.

Many variants of these techniques exist, as well as hybrid techniques such as KALMANSAC [98].

6.5 Conversion to B-splines

Given their importance, B-splines were explained in detail in Chapter 5. Trajectory data, once matched, still contains outliers in areas of high noise. The B-spline algorithm is able to cope with these outliers without affecting the overall shape of the trajectory.

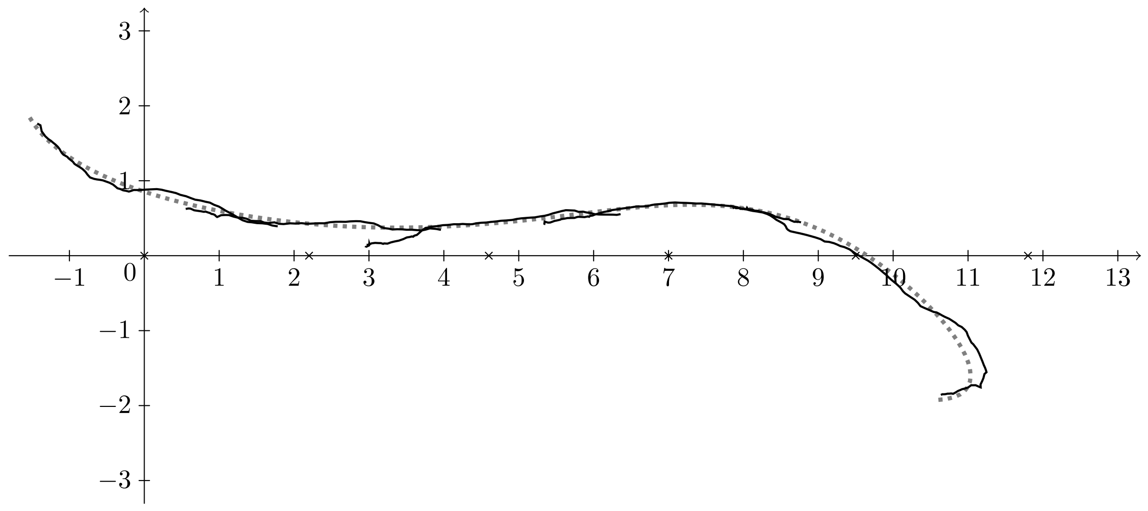


FIGURE 6.4: A raw trajectory shown in black with its associated B-spline shown as a grey dotted line.

Inevitably, as Figure 6.4 shows, there is some degree of error. The relatively low error rate meant that very little extra noise was added to the already noisy data from the sensors.

There are many forms of line representation already used in GIS and in trajectory compression. One example is the Douglas-Peucker algorithm, another is synchronous Euclidean distance (SED). These are explained in more detail in [95, p. 63]. However, the purpose of these algorithms are very different than in this research. There is no guarantee that information about speed and acceleration are preserved, and detail is lost. They are designed for much higher volumes of data, used for purposes other than studying pedestrian movements, where there is a greater need for accuracy. The compression ratio for the B-splines is shown in Table 6.3.

	Min	Max	Std.	Mean
Compression ratio	94.96 %	98.65 %	0.57 %	97.25 %
Error (m per data point)	0.0060	0.0207	0.0022	0.0107
Points per raw trajectory	139	518	57.78	266.01

Table 6.3: Statistics for conversion of raw data to B-splines.

6.6 Classification

All trajectories were classified according to their origin and destination, as well as the type of situation. There were twelve possible OD pairs, summarised in Table 6.4. The OD pairs between B56 and the Apex Café were ignored, given the short distance and lack sensor coverage between them.

Origin	Destination
Apex Café	Foyer Upper
Apex Café	Foyer Middle
Apex Café	Foyer Lower
B56	Foyer Upper
B56	Foyer Middle
B56	Foyer Lower
Foyer Upper	B56
Foyer Middle	B56
Foyer Lower	B56
Foyer Upper	Apex Café
Foyer Middle	Apex Café
Foyer Lower	Apex Café

Table 6.4: Origin and destination pairs.

The classification for the scenarios encountered are shown in Table 6.5.

Scenario	Description
Single	One person walking down the corridor alone.
Crossing	One person crossing another person's path.
Pair	Two people walking together.
Overtaking	One person overtaking another.
Following	One person following another.
Not contemporaneous	Two people whose trajectories don't interact.
Unprocessed	Trajectories that don't meet any of the other criteria.

TABLE 6.5: Definition of scenarios.

6.7 Database

As shown in Figure 6.5, the structure of the database used to store the B-splines representing trajectories is extremely simple. Each time interval recorded in the `timeinterval` table has one or many trajectories associated with it through a compound primary key in the `trajectory` table.

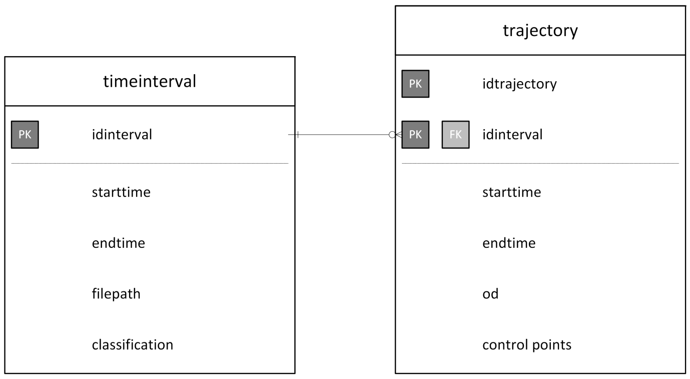


FIGURE 6.5: Entity relationship diagram for the database.

While the `timeinterval` table holds information about start time and end time, as well as other information, records in the `trajectory` table contain the control points for the B-spline.

Building on spatial databases already used in GIS, a new field of study, spatio-temporal or moving object databases, has very recently started to emerge. These are extremely sophisticated are discussed more in Section 8.4.2.

6.8 Conclusion

The results of the batch process are shown in Tables 6.6 to 6.8. In Table 6.6, the number of time slices, here called time intervals, accepted or rejected is given. In Table 6.7, the number of accepted time intervals per scenario is given. In Table 6.8, the kinds of errors and the numbers of time intervals rejected for each is given.

Number of days	99		
Number of intervals	117,445		
Accepted	27,702	23·69	%
Rejected	89,743	76·41	%

Table 6.6: Overall metrics

Single	22,575
Crossing	2,468
Pair, overtaking and following	2,031
Not contemporaneous	628

Table 6.7: Classification of accepted data

Not enough files	342
Not enough data	47,267
Too many targets	41,020
Invalid trajectories	1,114
Matching error	0
Matrix rank deficient	0

Table 6.8: Error types and incidence

Chapter 7

Data Analysis

7.1 Introduction

In this chapter, some of the properties of the acquired data are investigated. In this instance, summary statistics for speed and trajectory length for single and crossing trajectories are detailed. This is a limited exploration of the data. The purpose is to compare with studies available from literature. There is limited availability of information on pedestrian speed, summarised in [99].

In Section 7.2, unconstrained walking speed is examined. In Section 7.3, the results in Section 7.2 are compared with those of crossing trajectories to discern differences in trajectory speed and length.

7.2 Single trajectories

7.2.1 Speed distribution

In total, 22,575 single trajectories were recorded. These provide an opportunity to compare various measures against other studies of pedestrian behaviour. Pedestrian walking speed is of particular interest, a subject that was analyzed in detail by Daamen and Hoogendoorn in [99]. Their term for single trajectories is unconstrained trajectories. It is possible to compare results with not only the literature cited in this paper, but also with their own results.

Table 2.2 taken from this paper shows a remarkable consistency between speeds recorded by various researchers worldwide. In Table 7.2, summary statistics for speeds in the Apex corridor are given. The overall mean was $1.5\,\mathrm{m\,s^{-1}}$, with a standard deviation of $0.28\,\mathrm{m\,s^{-1}}$. This is consistent with Daamen and Hoogendoorn's finding of average speeds in European studies averaging $1.41\,\mathrm{m\,s^{-1}}$. In their laboratory experiments, they find unconstrained mean speeds of $1.49\,\mathrm{m\,s^{-1}}$ and a standard deviation of $0.15\,\mathrm{m\,s^{-1}}$.

Additionally, the skew of the trajectories is positive. This is also consistent with results cited in [99].

7.2.2 Trajectory length and distribution

A further table, Table 7.3 shows that there is a significant lack of divergence in trajectory length between the twelve studied origins and destinations. Even more remarkable, when the cumulative data for each origin and destination is plotted using a pseudocolour chart, it quickly becomes apparent that the trajectories tend to cluster, as shown in Figure 7.1. On further investigation, by calculating the mean trajectory, and its standard deviation, a remarkable level of consistency emerges. This is shown in Figure 7.2. The histograms for trajectory lengths are given in Figure 7.3.

Trajectory type	Avg. length	Avg. std
Single	13.52	0.48
Crossing	13.57	0.62

TABLE 7.1: Comparison of Single and Crossing Trajectories

7.3 Crossing trajectories

7.3.1 Speed

Speeds between single trajectories and crossing trajectories, given the same origins and destinations, differ substantially. The overall mean speed dropped from $1.5\,\mathrm{m\,s^{-1}}$ to $1.35\,\mathrm{m\,s^{-1}}$. Additional, where the skew of the summary statistics of single trajectories were consistently positive, six out of twelve of the origin-destination pairs considered for crossing trajectories showed negative skew.

7.3.2 Length of trajectory

The length of crossing trajectories were surprisingly not much longer than those of the single trajectories, although the standard deviation was somewhat larger. This suggests that in the confines of the corridor, people optimise the amount of manoeuvering they do to avoid oncoming pedestrians, while reducing speed.

7.4 Other scenarios

Other possible scenarios that can be derived from the data in the database are: pairs of people walking together, people following others ahead of them and overtaking. Distinguishing between people walking in the same direction is difficult in the form in which the data is at present.

7.5 Conclusion

As this chapter shows, the results for the data are consistent with European and experimental averages in terms of speed. This validates much of the post-processing of the raw data explained in the previous chapter. However, given the filtering of raw data before processing, the trajectories used in deriving these statistics were of the best quality in the dataset. The mean width of trajectories, which is relatively narrow, shown in Figure 7.2 may therefore be an artefact of this. They could be said to pass through the area where the accuracy of the sensors are at their best. However, it is clear that at the various origins and destinations, at the edges of the sensors' field of view, the means and standard deviations of trajectories appear be equally narrow. Given the amount of rejected data, there is no guarantee that these results are representative. However, it is encouraging that even in a limited way, the data for unconstrained walking are consistent in terms of mean and distribution speed, including the skew of distributions.

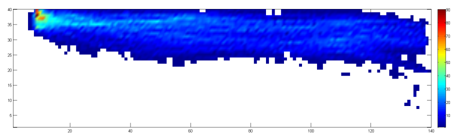
Comparison with crossing trajectories showed an expected speed reduction. However the trajectories are not much greater in length compared to unconstrained pedestrians. This is unexpected and warrants further investigation.

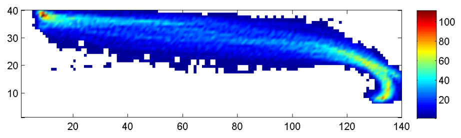
		sample	mean	std dev					
Origin	Destination	size	$m.s^{-1}$	$m.s^{-1}$	skew	kurtosis	median	min	max
Apex	Foyer Upper	3090	1·42	0·25	0·38	2·45	1·42	0·54	3·15
Apex	Foyer Middle	608	1·35	0·25	0·14	0·22	1·35	0·65	2·16
Apex	Foyer Lower	2940	1·38	0·25	0·52	2·01	1·38	0·54	2·67
B56	Foyer Upper	1850	1·59	0·3	0·15	2·74	1·6	0·55	3·7
B56	Foyer Middle	723	1·52	0·32	0·7	5·86	1·51	0·52	3·74
B56	Foyer Lower	2688	1·55	0·26	0·62	3·72	1·53	0·52	3·15
Foyer Upper	Apex	3369	1·48	0·26	0·37	1·9	1·48	0·62	3·02
Foyer Upper	B56	1632	1·69	0·32	0·87	5·87	1·67	0·58	3·74
Foyer Middle	Apex	446	1·5	0·3	0·56	2·96	1·5	0·56	3·06
Foyer Middle	B56	511	1·65	0·35	0·8	5·45	1·66	0·72	3·9
Foyer Lower	Apex	2618	1·34	0·24	0·17	0·28	1·34	0·58	2·29
Foyer Lower	B56	2100	1·48	0·28	0·26	0·94	1·47	0·66	3·0
	Totals	22575	1·5	0·28					

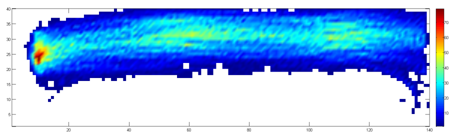
TABLE 7.2: Summary statistics for unconstrained trajectory speed.

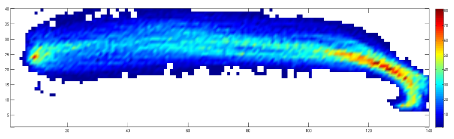
Origin	Destination	sample size	mean m	std dev m	skew	kurtosis	median	min	max
Apex	Foyer Upper	3090	13·57	0·56	0·31	0·1	13·55	12·08	16·59
Apex	Foyer Middle	608	13·81	0·42	1·09	2·44	13·75	12·94	16·3
Apex	Foyer Lower	2940	14·07	0·56	0·63	1·25	14·03	12·19	18·13
B56	Foyer Upper	1850	13·12	0·57	2·59	29·47	13·1	11·72	21·32
B56	Foyer Middle	723	13·33	0·28	2·28	9·71	13·26	12·61	15·31
B56	Foyer Lower	2688	13·69	0·46	1·22	5·38	13·65	12·48	17·69
Foyer Upper	Apex	3369	13·66	0·55	1·3	10·12	13·59	11·92	20·53
Foyer Upper	B56	1632	12·93	0·46	1·84	10·15	12·88	11·0	17·29
Foyer Middle	Apex	446	13·78	0·47	0·93	1·37	13·7	12·63	15·91
Foyer Middle	B56	511	13·22	0·39	2·54	11·3	13·13	12·31	16·35
Foyer Lower	Apex	2618	13·85	0·62	0·87	1·23	13·75	11·98	17·06
Foyer Lower	B56	2100	13·27	0·47	1·25	3·49	13·19	11·67	16·24
	Totals	22575	13·52	0·48					

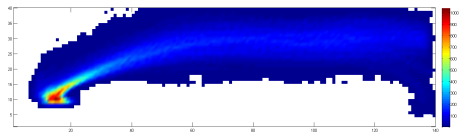
TABLE 7.3: Summary statistics for unconstrained trajectory length.

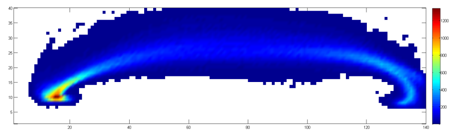

(a) Foyer Upper to B56

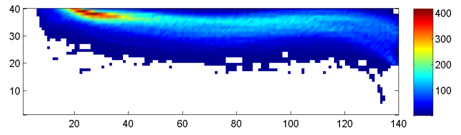

(b) Foyer Upper to Apex

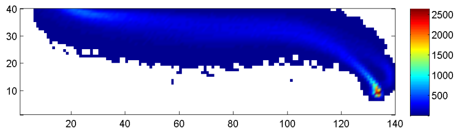

(c) Foyer Middle to B56

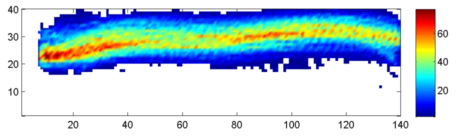

(d) Foyer Middle to Apex

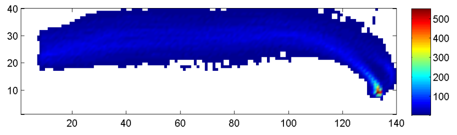

(e) Foyer Lower to B56

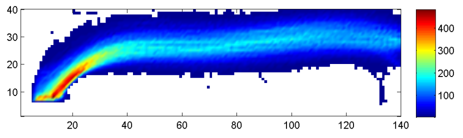

(f) Foyer Lower to Apex


(g) B56 to Foyer Upper


(h) Apex to Foyer Upper


(i) B56 to Foyer Middle


(j) Apex to Foyer Middle


(k) B56 to Foyer Lower

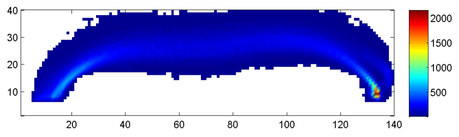

(l) Apex to Foyer Lower

FIGURE 7.1: Cummulative results for unconstrained single pedestrian trajectories by origin and destination

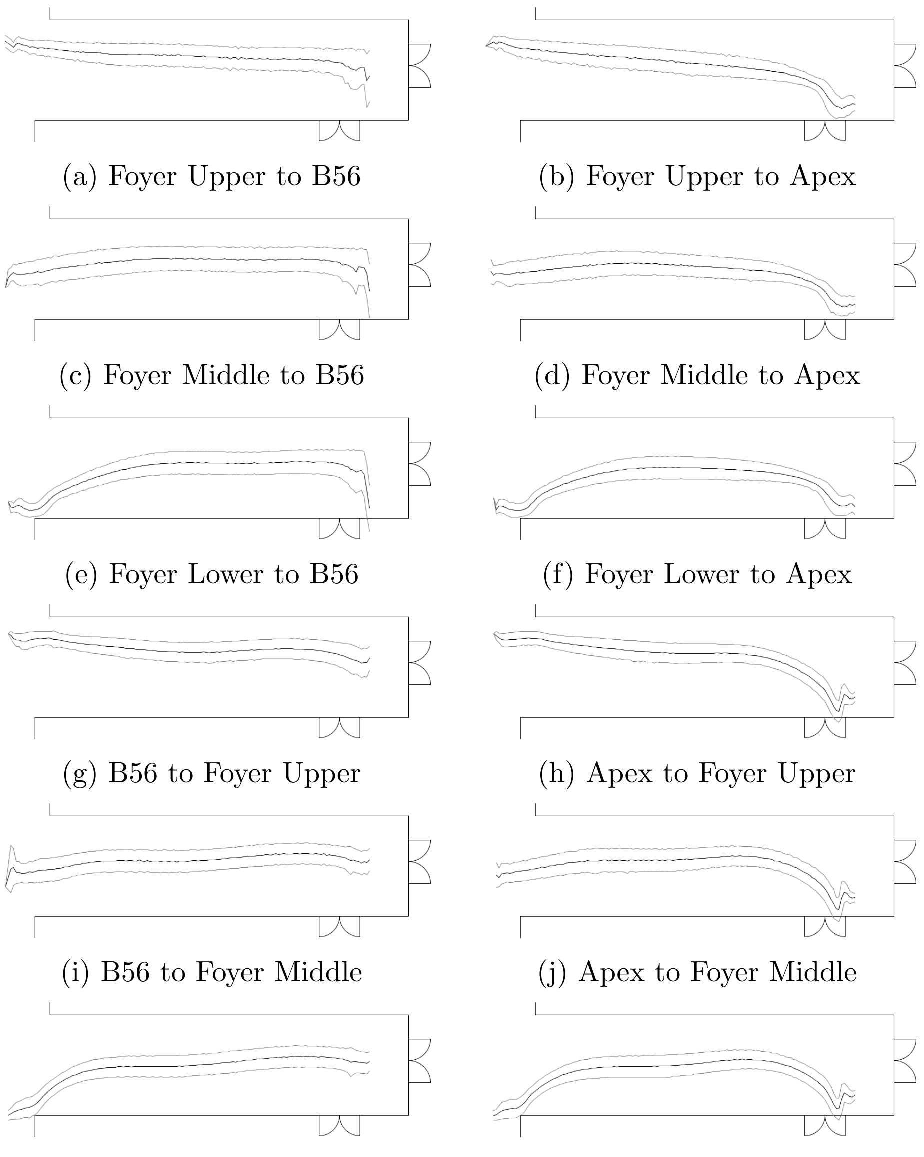
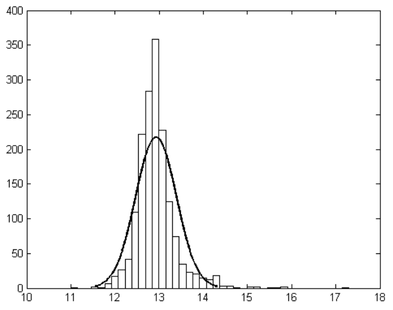
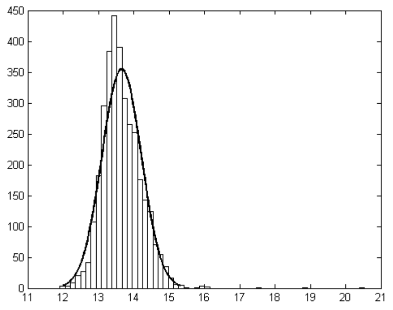
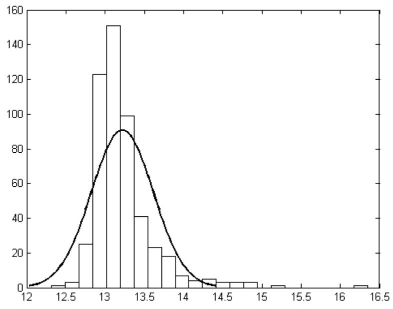
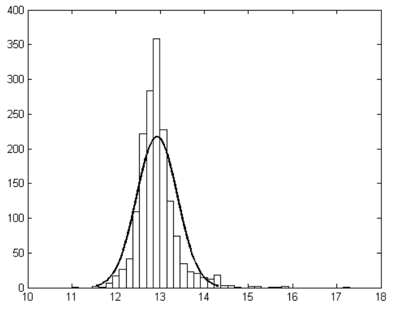

(a) Foyer Upper to B56
(b) Foyer Upper to Apex
(c) Foyer Middle to B56
(d) Foyer Middle to Apex
(e) Foyer Lower to B56
(f) Foyer Lower to Apex
(g) B56 to Foyer Upper
(h) Apex to Foyer Upper
(i) B56 to Foyer Middle
(j) Apex to Foyer Middle
(k) B56 to Foyer Lower
(l) Apex to Foyer Lower

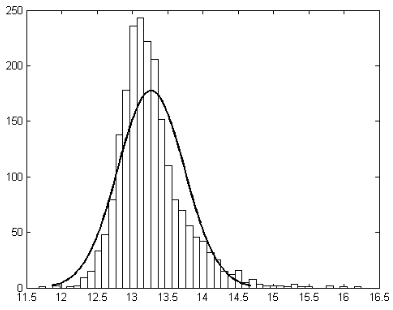
FIGURE 7.2: Mean (black) and standard deviation (grey) of unconstrained single pedestrian trajectories by origin and destination

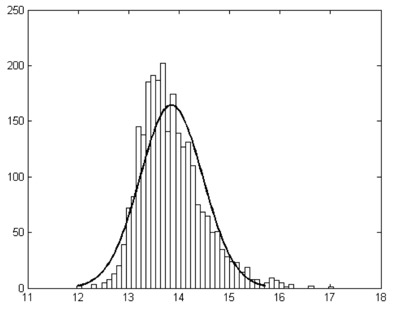

(a) Foyer Upper to B56

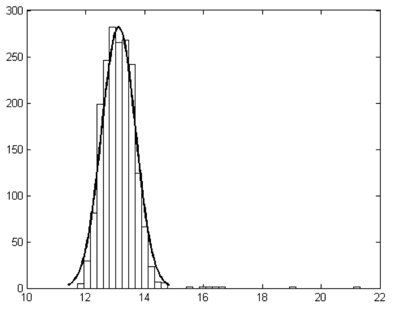

(b) Foyer Upper to Apex

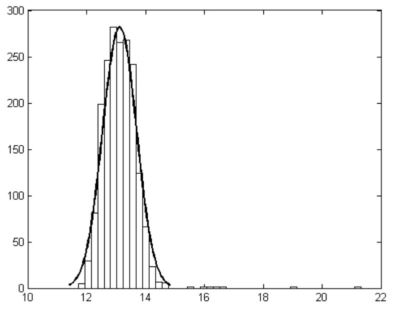

(c) Foyer Middle to B56

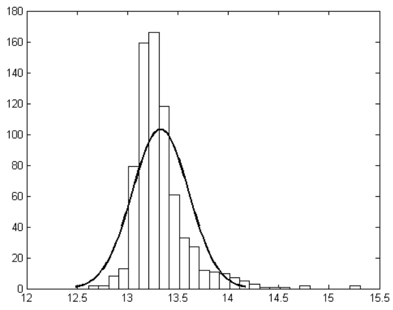

(d) Foyer Middle to Apex

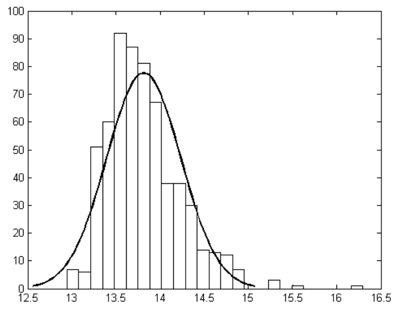

(e) Foyer Lower to B56


(f) Foyer Lower to Apex


(g) B56 to Foyer Upper


(h) Apex to Foyer Upper


(i) B56 to Foyer Middle


(j) Apex to Foyer Middle

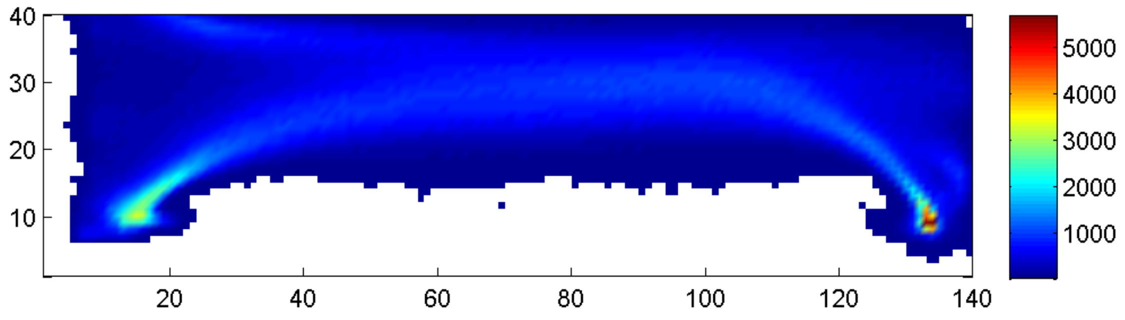


Figure 7.4: Cummulative total of all single pedestrian trajectories

Origin	Destination	sample size	mean $m.s^{-1}$	std dev $m.s^{-1}$	skew	kurtosis	median	min	max
Apex	Foyer Upper	283	1·29	0·22	−0·3	0·28	1·29	0·53	1·83
Apex	Foyer Middle	68	1·2	0·22	−0·11	−0·48	1·19	0·69	1·65
Apex	Foyer Lower	312	1·28	0·21	0·21	0·58	1·26	0·61	2·06
B56	Foyer Upper	196	1·42	0·23	−0·43	0·67	1·44	0·65	2·13
B56	Foyer Middle	93	1·35	0·26	−0·2	0·78	1·35	0·51	1·95
B56	Foyer Lower	285	1·39	0·21	−0·02	1·16	1·39	0·69	2·28
Foyer Upper	Apex	270	1·37	0·2	0·13	0·37	1·36	0·7	1·98
Foyer Upper	B56	174	1·5	0·25	0·33	2·5	1·47	0·7	2·69
Foyer Middle	Apex	58	1·41	0·25	0·09	−0·09	1·38	0·87	1·98
Foyer Middle	B56	77	1·52	0·25	−0·36	1·43	1·51	0·7	2·19
Foyer Lower	Apex	358	1·24	0·21	0·25	0·58	1·23	0·58	1·9
Foyer Lower	B56	294	1·35	0·2	0·05	−0·19	1·33	0·79	1·89
	Totals	2468	1·36	0·23					

TABLE 7.4: Summary statistics for crossing trajectory speed.

		sample	mean	std dev					
Origin	Destination	size	*m*	*m*	skew	kurtosis	median	min	max
Apex	Foyer Upper	283	13·63	0·54	0·49	0·68	13·59	12·31	15·52
Apex	Foyer Middle	68	13·93	0·43	0·84	1·67	13·9	13·17	15·53
Apex	Foyer Lower	312	14·07	0·62	0·54	−0·32	13·98	12·94	15·72
B56	Foyer Upper	196	13·16	0·51	0·5	0·48	13·1	12·07	15·22
B56	Foyer Middle	93	13·39	0·3	2·94	15·52	13·31	12·94	15·26
B56	Foyer Lower	285	13·89	1·83	15·28	249·18	13·76	12·58	43·71
Foyer Upper	Apex	270	13·71	0·55	0·37	0·11	13·65	12·35	15·46
Foyer Upper	B56	174	12·98	0·55	1·15	4·95	12·94	11·37	15·8
Foyer Middle	Apex	58	13·74	0·59	1·79	7·15	13·73	12·67	16·5
Foyer Middle	B56	77	13·26	0·41	1·76	3·03	13·16	12·65	14·68
Foyer Lower	Apex	358	13·83	0·63	1·02	1·46	13·73	12·67	16·47
Foyer Lower	B56	294	13·29	0·49	1·08	1·77	13·22	12·36	15·15
	Totals	2468	13·57	0·62					

Table 7.5: Summary statistics for crossing trajectory length.

Chapter 8

Conclusion

8.1 Introduction

To recapitulate from Chapter 1, the research questions are as follows:

1. Can an automated and unattended pedestrian data collection system be created using the Irisys MkII sensors? If so, which of the scenarios listed in Table 8.1 can they provide data for?
2. Can the data be processed in such a way as it can be stored and retrieved easily without loss of information?
3. Can the data returned be processed in an accurate and efficient way?
4. Can the statistics derived from the data be validated against other studies?
5. Is it possible to generalise from a specific system to generic principles?

In this chapter each of these questions will be addressed in turn in Section 8.2. In Section 8.3, recommendations for alleviating the problems in the system are made. The chapter concludes with Section 8.4 where suggestions are made for further work.

Scenario	
Single pedestrians	
Two pedestrians	pair crossing following overtaking
Groups of three or more	
Crowds	

Table 8.1: Trajectory scenarios

8.2 Conclusions

8.2.1 Results

8.2.1.1 Sensors

The Irisys MkII sensors did not perform as well as expected. However, given the list of scenarios specified in Chapter 1 four out of six could be covered by the system. Effectively this meant that situations involving three or more pedestrians could not be tracked reliably. While many of the reasons for this were down to the outdated communications capabilities of the sensors, this is a more general problem in tracking applications.

As Koch [100] states:

> “In many distributed tracking applications, the quality of a state estimation suffers from communication bandwidth limitations. In particular, scenarios using wireless channels such as HF radio, WLAN, or 3G networks experience link breakdowns and limited capacity constraints. Furthermore, if sensors are involved which have a high update rate, e.g. sonar or lidar, or return much clutter, e.g. radar, network technologies

> are hardly sufficient to cover all needs with respect to a constantly full transfer of measurements." [100, p. 119]

Therefore, merely upgrading the sensor technology used, or using a different means of tracking pedestrians, does not on its own guarantee the absence of communications bandwidth problems. In fact, augmented sensor capabilities may in fact worsen this problem.

In this instance, the limitations of the system severely limited its applicability. To an extent, the built-in tracking system in the sensors removed the burden of implementing similar capabilities. However this system was undocumented and proprietary, removing any flexibility in dealing with problems that arose.

Another drawback of using a proprietary system was lack of information about errors likely to be generated by the system. In [101], a distinction is made between system errors and random errors. An example of a system error would be the radial distortion in the germanium lens, an intrinsic part of the system. It was possible to compensate for this to an extent, although the experimental findings in correcting radial distortion is at best only an estimate.

Given the low-resolution nature of the sensor's focal plain array, of itself, this was a source of both system and random errors. In the experiments described in Chapter 3, there was increasing variance in tracks that were identical, the further they were from the centre of the lens. Even at the centre of the lens, there was some variance in the tracks recorded. Although the resulting errors are likely to represent centimetres at most, the level of accuracy demanded is very much a decision for the designer of any new system.

A possible source of errors, not dependent on the sensors, was the the effect of using serial to Ethernet servers to overcome communications problems. Network communications are dealt with by coalesced interrupts on the receiving computer. While these providing buffering far superior to the 16 B of RS232 ports, they are addressed in a non-deterministic way. An obvious to this would have been to acquire better serial to Ethernet servers that supported network time protocol and were able to synchronise their internal clocks with the receiving computer, as well as process and timestamp data from the sensors. While this would eliminate one likely cause of error, the benefits may be marginal.

8.2.1.2 Data processing

Since no estimation could be easily made of error in the data produced, an overriding requirement in processing the data was that no additional error should be introduced. In this respect, B-splines served this purpose better than expected with a very high compression ratio and very low rate of error, as measured by the sum of errors squared.

The type of B-spline used was however one of the simplest, and the trajectory lengths and general form were easily accommodated by this type of spline. Longer trajectories, with more complex or sinusoidal forms, may require more sophisticated B-splines. Currently, there is little if any well-tested software available for creating and manipulating B-splines that is open source. Propriety software such as MATLAB [75] do provide this.

Track fusion was done using brute force techniques to combine trajectories from the sensors. This was not particularly flexible as it was only shown to work on a maximum of two cooccuring trajectories at a time. The main algorithms used for track fusion are Kalman filters and RANdom SAmple Consensus (RANSAC). These are described in Sections 6.4.1 and 6.4.2.

The classification of trajectories into various scenarios was less successful than expected. Many trajectories could not be classified. Simple algorithmic procedures do not accommodate data with so many degrees of freedom particularly well.

Database support was relatively trivial to implement once the data had been compressed using B-splines. However, the database implemented was only suitable for storage and retrieval. Spatial extensions, available for most database systems were not explored, neither was the emerging field of spatio-temporal databases described in Section 8.4.2.

8.2.1.3 Data analysis

Statistics were relatively simple to derive from the data set, and were found to be consistent with those in [7, 8]. This was a very limited use of the data available. A major area of research is behaviour classification [81, 82, 102–104], which has received a great deal more attention than pedestrian interactions. This field also overlaps with machine learning. Given time constraints, it was not possible to explore this avenue of research.

8.2.2 Discussion

Addressing the research questions given above in Section 8.1, the results are as follows:

1. It was possible to create an automatic and unattended pedestrian data collection system using the Irisys MkII sensors. These were reliable in the sense that they rarely suffered from hardware errors that required them to be reset.

2. However, the quality of the data returned suffered greatly from bandwidth limitations of the sensors. A great deal of data were corrupted and lost. In that sense the sensors were a not reliable source of pedestrian data. The data that was uncorrupted was relatively limited in that a maximum of two pedestrians in the field of view could be guaranteed to be tracked.

3. Corrupted data had to be filtered out at the start of processing. Track fusion was done in a simplistic manner on a very limited set of the data available, with a maximum of two pedestrians in the field of view of the sensors to guarantee success. Valid data was then converted into B-splines. B-splines were shown to be extremely capable in smoothing and compressing data while preserving speed and acceleration information from the recorded trajectory. Storage of a fixed length vector of B-spline coefficients was relatively straightforward. However, this storage strategy was rudimentary, involving a MySQL server instance running on a standard desktop PC. No spatial extensions were used.

4. Statistics were derived from a very limited number of scenarios. It was possible to compare these with literature and find agreement.

5. The specific system did not achieve its objectives. However, with the quick pace of progress in the field of video and IR surveillance, and the increasing affordability of equipment, many of the problems discussed above would and could be resolved. In principle, instrumentation of public spaces is entirely feasible.

8.3 Recommendations

The validity of any dataset is contingent on its accuracy, therefore the use of Irisys MkII sensors beyond their design specifications is not recommended. At the moment, the Microsoft Kinect is available at a similar price point. This is far more capable. As an active rather than passive IR sensor, it is able to provide depth maps, as well as IR tracking information, potentially providing three dimensional spatio-temporal data. In addition, it is a visible light video camera; this operates from the same lens as the IR sensor. This would allow proper measurements of an area under study in order to accurately determine the ground truth, markedly increasing the quality of data. A second advantage of this dual capability would be the ability to capture demographic information about pedestrians.

The use of B-splines, given the usefulness of their properties, should be investigated further. Even in their simplest form, they provided a great deal of flexibility in storing and analysing the data captured. Possible applications of this form of representation of spatio-temporal information are discussed in Section 8.4.

Track fusion is an inevitable problem when sensors are deployed. The adaptation of standard algorithms to this area of research would be recommended. More capable sensors would not solve this problem of themselves.

A fuller statistical analysis of the data acquired, creating a template for future applications, is also recommended.

8.4 Further Work

8.4.1 B-splines

The capabilities of B-splines in this context have not been fully explored. The possibility of using the derivatives of be splines to gather information about speed variations within a curve, rather than just the speed across the whole of the curve, haven't been explored. These derivatives would support the acquisition of further information, such as determining inflection points and speed variations.

Although clamped uniform B-splines were used here, other types of B-splines would be more suitable for finer grained and sinusoidal data. These would include B-splines with biinfinite knot vectors.

As mentioned above, the ability to calculate derivatives, with guaranteed continuity, analytically has not been explored. This would provide a means of extracting more sophisticated statistics from datasets, such as finding inflection points accurately.

8.4.2 Spatio-temporal databases

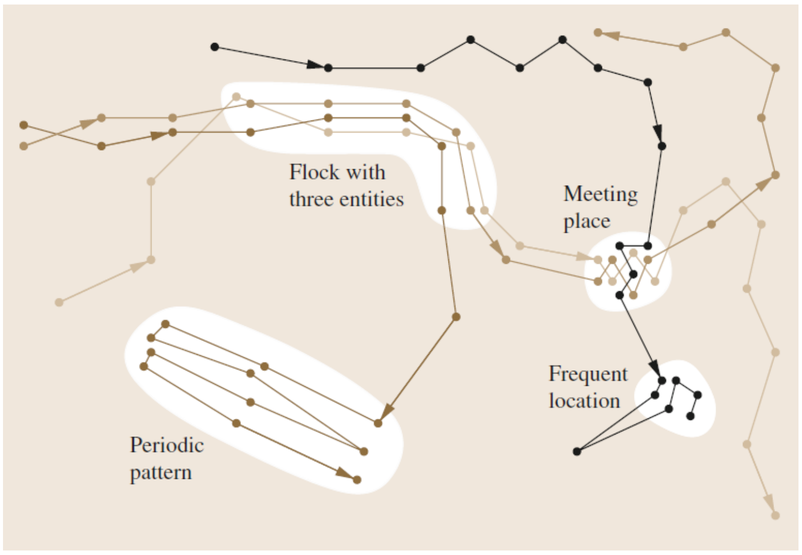


Figure 8.1: Flock, meeting, frequent location and periodic patterns in movement data. Taken from [105].

A recent development in GIS are spatio-temporal or moving object databases. Although the design of efficient indexing systems and the specification of required capabilities has been studied for many years, only very recently have the implementations become available. One such is Hermes [106]. This adds spatio-temporal capabilities to the existing database management systems (DBMS) Oracle and PostgresSQL, as described in [95]. (This is not be confused with the Hermes project at the Jülich Supercomputing Centre described by Seyfried et al. in [107] or the FP6 funded HERMES program IST-027110).

Another development in GIS is data mining to discover trajectory patterns [108], or computational movement patterns [105]. Various behaviour patterns were specified by Laube et al. in [109], illustrated in Figure 8.1. Both these developments merit further investigation in the context of pedestrian dynamics.

Appendix A

B-splines

A.1 Listing

```java
import Jama.Matrix;

public class JBsplvb {

    private static double testcoords[][] = {
            {-5.843105, 8.770474, 0.0},
            {-5.16712901, 7.477139516, 0.0843},
            {-2.17495799, 5.0200987, 0.1686},
            {5.694259025, 3.442695525, 0.2557},
            {10.96855033, 3.137490172, 0.3413},
            {20.80839729, 2.669446467, 0.427},
            {26.91894558, 2.820262283, 0.5113},
            {35.17994832, 2.142998648, 0.597},
            {42.31869639, 1.598836571, 0.684},
            {49.39105711, 0.470515525, 0.7683},
            {54.49460793, -1.185606044, 0.854},
            {60.01635061, -4.17355203, 0.9384},
            {62.73287282, -8.206721518, 1.0}
        };

    private double[] tau = null;

    private double[][] coords = null;
    private int order=4;
    private int ncontrolpoints=7;
    private int newlength=100;

    private Matrix oldbasis = null;
    private Matrix newbasis = null;
    private Matrix alpha = null;
    private Matrix newcoords = null;

    private double tN=0;

    public static void main(String[] args) {
        JBsplvb spline = new JBsplvb(testcoords);
        spline.report();
    }
```

```
public JBsplvb(double[][] coords) {
    this.coords = coords;
    init();
}

public JBsplvb(double[][] coords,
            int newlength, int order, int ncontrolpoints) {
    this.coords = coords;
    this.newlength = newlength;
    this.order=order;
    this.ncontrolpoints=ncontrolpoints;
    init();
}

public void report() {
System.out.println("Point reduction: ");
System.out.print("Original points: " + coords.length + " Coefficients: "
        + ncontrolpoints + " Reduction: "
        + (coords.length - ncontrolpoints));
System.out.println(" Percentage: "
        + (100 - (ncontrolpoints / (double) coords.length) * 100) + "\%");
double sumoferror = Math.sqrt(sumoferrorsquared);
double avgerror = sumoferror / coords.length;
System.out.println("Sum of error^2=" + sumoferrorsquared
        + ", Sum of error=" + sumoferror
        + ", Average error=" + (avgerror * 100));
}

public void init() {
    tau=makeknots();
    newbasis = buildbasismatrix(newlength);
    approximate();
    calculate();
}

public void approximate() {
    oldbasis = new Matrix(coords.length, ncontrolpoints);
    Matrix Coords = new Matrix(coords);
    normalisetime(Coords);
    oldbasis = buildbasismatrix(Coords);
    Matrix Linverse = oldbasis.inverse();
    alpha = null;
    alpha = Linverse.times(Coords);
}

private Matrix buildbasismatrix(int size) {

    double step = 1/(double)size;
    double[] t = new double[size+1];
    for(int j = 0; j <=size; j++) {
        t[j] = step * j;
    }

    return buildbasismatrix(t);
}

private Matrix buildbasismatrix(double[] t) {
    Matrix L = new Matrix(t.length, ncontrolpoints);
    for(int j = 0; j < t.length; j++) {

    int i = interval(t[j]);

    double[] br = bsplvb(t[j], order, i);
    for (int zz = 1; zz <= order; zz++)
        L.set(j, i - order + zz - 1, br[zz]);
    }
    return L;
}

private Matrix buildbasismatrix(Matrix m) {
    double[] t = new double[m.getRowDimension()];
```

```
        for (int i = 0; i < m.getRowDimension(); i++) {
            t[i] = m.get(i, 2);
        }
        return buildbasismatrix(t);

    }

    private void calculate() {
        if (newbasis==null) newbasis=buildbasismatrix(newlength);
        newcoords = new Matrix(newbasis.getRowDimension(), 3);
        for(int j = 0; j < newbasis.getRowDimension(); j++) {
            double bx = 0;
            double by = 0;
            double bt = 0;
            for (int i = 0; i < newbasis.getColumnDimension(); i++) {
                bx += alpha.get(i,  0) * newbasis.get(j, i);
                by += alpha.get(i, 1) * newbasis.get(j, i);
                bt += alpha.get(i,  2) * newbasis.get(j, i);
            }
        newcoords.set(j, 0, bx);
        newcoords.set(j, 1, by);
        newcoords.set(j, 2, bt);
        }
    }

    /** Array indexing in Java is zero based. FORTRAN aray indices start at 1.
     *  In order to preserve the indexing used in Carl de Boors original
     *  text the first element in the array br, br[0], is ignored.
    **/

    private double[] bsplvb(double t, int order, int i) {

        double[] br = new double[order + 1];
        double[] deltar = new double[order + 1];
        double[] deltal = new double[order + 1];
        br[1] = 1;

        for (int j = 1; j <= order - 1; j++) {
            deltar[j] = tau[i + j] - t;
            deltal[j] = t - tau[i + 1 - j];
            double saved = 0;
            for (int r = 1; r <= j; r++) {
                double term = br[r] / (deltar[r] + deltal[j + 1 - r]);
                br[r] = saved + deltar[r] * term;
                saved = deltal[j + 1 - r] * term;
            }
            br[j + 1] = saved;
        }
        return br;
    }

    private  int interval(double t) {
        int i = 0;
        for (int z = 1; z < tau.length - 1; z++) {
            i = z;
            if (t == 0.0) {
                if (tau[z + 1] > 0.0)
                    break;
            } else if (t == 1.0) {
                if (tau[z + 1] == 1.0)
                    break;
            } else if (tau[z + 1] > t && tau[z] <= t)
                break;
        }
        return i;
    }

    /** As the knot vector is used by the function bsplvb
     *  the first array location knots[0] is ignored, so that
     *  the knot vector can use 1 as its first index.
    **/
    private double[] makeknots() {
        int intknots = ncontrolpoints - order;
        double[] knots = new double[(2*order)+intknots+1];
        knots[0] = -1;
        for(int i = 1; i<= order; i++) {
            knots[i] = 0;
            knots[i  + order + intknots] = 1;
        }
        for(int i = 1; i <= intknots; i++) {
            double inc = (1/((double)intknots+1));
            knots[order + i] = inc * i;
        }
        return knots;
    }
    private void normalisetime(Matrix Coords) {
        tN=coords[coords.length-1][2];
        for (int i = 0; i < Coords.getRowDimension(); i++)
            Coords.set(i, 2, coords[i][2]/tN);
        }

    private void denormalisetime() {
        for (int i = 0; i < coords.length; i++)
            coords[i][2] *= tN;
    }
    public double[][] getCoords() {
        return coords;
    }

    public Matrix getOldbasis() {
        return oldbasis;
    }

    public Matrix getNewbasis() {
        return newbasis;
    }

    public Matrix getAlpha() {
        return alpha;
    }

    public double[][] getNewcoords() {
        return newcoords.getArrayCopy();
    }

    public void setCoords(double[][] coords) {
        this.coords = coords;
    }

    public void setAlpha(Matrix alpha) {
        this.alpha = alpha;
    }
}
```

Appendix B

Published Papers

B.1 CSP for .NET Based on JCSP

B.2 Tracking and Analysis of the Movement of Pedestrians

B.3 Application of CoSMoS Parallel Design Patterns to a Pedestrian Simulation

B.4 An Application of CoSMoS Design Methods to Pedestrian Simulation

B.5 Simulating Cellular Automata Using Go

Bibliography

[1] Jake Desyllas, Elspeth Duxbury, John Ward, and Andrew Smith. Pedestrian demand modelling of large cities: an applied example from London. Technical report, Centre for Advanced Spatial Analysis (UCL), London, UK., 2003. 1, 2

[2] *The Mayor's Transport Strategy.* 2001. URL `http://legacy.london.gov.uk/gla/publications/transport.jsp`. 1

[3] J. Fruin. *Pedestrian Planning and Design.* Metropolitan Association of Urban Designers and Environmental Planners, 1971. 1, 24

[4] S. P. Hoogendoorn and P. H L Bovy. Pedestrian route-choice and activity scheduling theory and models. *Transportation Research Part B: Methodological*, 38(2):169–190, 2004. ISSN 01912615. doi:10.1016/S0191-2615(03)00007-9. 3

[5] C J E Castle, N P Waterson, E Pellissier, and S Le Bail. A Comparison of Grid-based and Continuous Space Pedestrian Modelling Software: Analysis of Two UK Train Stations. In Richard D Peacock, Erica D Kuligowski, and Jason D Averill, editors, *Pedestrian and Evacuation Dynamics SE - 39*, pages 433–446. Springer US, 2011. ISBN 978-1-4419-9724-1. doi:10.1007/978-1-4419-9725-8_39. URL `http://dx.doi.org/10.1007/978-1-4419-9725-8_39`. 3

[6] W Daamen. A quantitative assessment on the design of a railway station. In J Allen, R J Hill, C A Brebbia, G Sciutto, and S Sone, editors, *Computers in railways VIII (Congress Proceedings of CompRail 2002)*, volume 2002, pages 191–200. WIT Press, 2002. 3

[7] Dirk Versluis. *Microscopic interaction behavior between individual pedestrians.* Master, Delft University of Technology, 2010. 4, 18, 19, 110

[8] Winnie Daamen, Serge Hoogendoorn, Mario Campanella, and Dirk Versluis. Interaction Behavior Between Individual Pedestrians. In Ulrich Weidmann, Uwe Kirsch, and Michael Schreckenberg, editors, *Pedestrian and Evacuation Dynamics 2012 SE - 107*, pages 1305–1313. Springer International Publishing, 2014. ISBN 978-3-319-02446-2. doi:10.1007/978-3-319-02447-9_107. URL `http://dx.doi.org/10.1007/978-3-319-02447-9_107`. 4, 18, 19, 20, 110

[9] Sarvesh Vishwakarma and Anupam Agrawal. A survey on activity recognition and behavior understanding in video surveillance. *The Visual Computer*, 29 (10):983–1009, 2013. doi:10.1007/s00371-012-0752-6. 4

[10] Stephen J Guy. Adopting Pedestrian Navigation Techniques for Multi-Robot Coordination. In *Collaboration Technologies and Systems (CTS), 2014 International Conference on*, pages 297–301, Minneapolis, MN, 2014. IEEE. ISBN 9781479951581. doi:10.1109/CTS.2014.6867580. 4, 20, 22, 23

[11] Alexandra Willis, Robert Kukla, Jon M Kerridge, and Julian Hine. Laying the foundations: The use of video footage to explore pedestrian dynamics in PEDFLOW. *Pedestrian and Evacuation Dynamics*, pages 181–186, 2002. 5

[12] Jon Kerridge, Alistair Armitage, David Binnie, and Lucy Lei. Monitoring the Movement of Pedestrians Using Low-cost Infrared Detectors : Initial Findings. *Transportation Research Record: Journal of the Transportation Research Board*, 1878:11–18, 2004. 8, 9

[13] J Kerridge, R Kukla, A Willis, A Armitage, D Binnie, and L Lei. A Comparison of Video and Infrared Based Tracking of Pedestrian Movements. In S. P. Hoogendoorn, S. Luding, P H L Bovy, M Schreckenberg, and D E Wolf, editors, *Traffic and Granular Flow 2003*, pages 383–391, 2005. 8

[14] Luca Mainetti, Luigi Patrono, and Ilaria Sergi. A survey on indoor positioning systems. In *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 111–120, 2014. ISBN 978-9-5329-0052-1. doi:10.1109/SOFTCOM.2014.7039067. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7039067`. 10, 14

[15] Robert Harle. IEEE Xplore - A Survey of Indoor Inertial Positioning Systems for Pedestrians. 15(3):1281–1293, 2013. doi:10.1109/SURV.2012.121912.00075. URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6407455`. 10, 16, 17, 18

[16] M Boltes. PeTrack. URL `http://www.fz-juelich.de/ias/jsc/EN/Research/ModellingSimulation/CivilSecurityTraffic/PedestrianDynamics/Activities/petrack/petrackNode.html`. 11

[17] Maik Boltes, Armin Seyfried, Bernhard Steffen, and Andreas Schadschneider. Automatic Extraction of Pedestrian Trajectories from Video Recordings. In Wolfram W F Klingsch, Christian Rogsch, Andreas Schadschneider, and Michael Schreckenberg, editors, *Pedestrian and Evacuation Dynamics 2008 SE - 3*, video;PeTrack, pages 43–54. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-04503-5. doi:10.1007/978-3-642-04504-2_3. URL `http://dx.doi.org/10.1007/978-3-642-04504-2_3`. 11, 12

[18] A Winkens, W Klingsch, and A Seyfried. New Data for Human Performance in Planar Corridors. In Richard D Peacock, Erica D Kuligowski, and Jason D Averill, editors, *Pedestrian and Evacuation Dynamics*, pages 61–70. Springer US, 2011. ISBN 978-1-4419-9724-1. doi:10.1007/978-1-4419-9725-8_6. URL `http://dx.doi.org/10.1007/978-1-4419-9725-8_6`. 11

[19] T Rupprecht, W Klingsch, and A Seyfried. Influence of Geometry Parameters on Pedestrian Flow through Bottleneck. In Richard D Peacock, Erica D Kuligowski, and Jason D Averill, editors, *Pedestrian and Evacuation Dynamics SE - 7*, pages 71–80. Springer US, 2011. ISBN 978-1-4419-9724-1. doi:10.1007/978-1-4419-9725-8_7. URL `http://dx.doi.org/10.1007/978-1-4419-9725-8_7`. 11, 12

[20] Rainer Mautz and Sebastian Tilch. Survey of optical indoor positioning systems. In *International Conference on Indoor Positioning and Indoor Navigation, IPIN*, number September, pages 21–23, 2011. ISBN 9781457718045. doi:10.1109/IPIN.2011.6071925. 11, 12

[21] Stefan Seer, Norbert Brändle, and Carlo Ratti. Kinects and human kinetics: A new approach for studying pedestrian behavior. *Transportation Research Part C: Emerging Technologies*, 48:212–228, 2014. ISSN 0968090X. doi:10.1016/j.trc.2014.08.012. URL `http://www.sciencedirect.com/science/article/pii/S0968090X14002289`. 13

[22] Mingsong Dou, Henry Fuchs, and Jan Michael Frahm. Scanning and tracking dynamic objects with commodity depth cameras. In *2013 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2013*, number October, pages 99–106, 2013. ISBN 9781479928699. doi:10.1109/ISMAR.2013.6671769. 13

[23] Federal Communications Commission. Enhanced 9-1-1 - Wireless Services. URL `https://www.fcc.gov/encyclopedia/enhanced-9-1-1-wireless-services`. 14

[24] Bluetooth SIG Inc. Adopted Specifications, 2016. URL `https://www.bluetooth.com/specifications/adopted-specifications`. 15

[25] Markus Köhne and Jürgen Sieck. Location-Based Services with iBeacon Technology. In *2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation*, pages 315–321, 2014. ISBN 978-1-4799-7600-3. doi:10.1109/AIMS.2014.58. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7102480`. 15

[26] W Daamen and S P Hoogendoorn. Free speed distributions Based on empirical data in different traffic conditions. In Nathalie Waldau, Peter Gattermann, Hermann Knoflacher, and Michael Schreckenberg, editors, *Pedestrian and Evacuation Dynamics 2005 SE - 2*, pages 13–25. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-47062-5. doi:10.1007/978-3-540-47064-9_2. URL `http://dx.doi.org/10.1007/978-3-540-47064-9_2`. 18, 19, 24

[27] J Ma, W G Song, S M Lo, G X Liao, and K K Yuen. Experimental Investigation on the Pair Interaction Between Pedestrians. In Richard D Peacock, Erica D Kuligowski, and Jason D Averill, editors, *Pedestrian and Evacuation Dynamics SE - 86*, pages 847–850. Springer US, 2011. ISBN 978-1-4419-9724-1. doi:10.1007/978-1-4419-9725-8_86. URL `http://dx.doi.org/10.1007/978-1-4419-9725-8_86`. 18

[28] Serge P Hoogendoorn and Winnie Daamen. Pedestrian behavior at bottlenecks. *Transportation Science*, 39(2), 2005. 19

[29] Christian Dondrup, Nicola Bellotto, Marc Hanheide, Kerstin Eder, and Ute Leonards. A Computational Model of Human-Robot Spatial Interactions Based on a Qualitative Trajectory Calculus. *Robotics*, 4:63–102, 2015. doi:10.3390/robotics4010063. 20, 23

[30] Anne-Hélène Olivier, Antoine Marin, Armel Crétual, and Julien Pettré. Minimal predicted distance : A common metric for collision avoidance during pairwise interactions between walkers. *Gait & Posture*, 36(3):399–404, 2012. doi:10.1016/j.gaitpost.2012.03.021. 20, 21, 22

[31] Thomas Ducourant, Yves Kerlirzin, and Alain Berthoz. Timing and distance characteristics of interpersonal coordination during locomotion. *Neuroscience Letters*, 389:6–11, 2005. doi:10.1016/j.neulet.2005.06.052. 20

[32] Stephane Vieilledent, Yves Kerlirzin, Stephane Dalbera, and Alain Berthoz. Relationship between velocity and curvature of a human locomotor trajectory. *Neuroscience Letters*, 305(2001):65–69, 2001. 20, 22

[33] Julien Bruneau, Teofilo B Dutra, and Julien Pettre. Following behaviors : a model for computing following distances. *Transportation Research Procedia 2*, 2(2014):424–429, 2014. doi:10.1016/j.trpro.2014.09.049. 20

[34] E Hall. *The Hidden Dimension*. Doubleday, New York, 1966. 20

[35] Andrea Gorrini, Kenichiro Shimura, Stefania Bandini, Kazumichi Ohtsuka, and Katsuhiro Nishinari. Experimental Investigation of Pedestrian Personal Space Toward Modeling and Simulation of Pedestrian Crowd Dynamics. *Transportation Research Record: Journal of the Transportation Research Board*, 2421:57–63, 2014. doi:10.3141/2421-07. 21

[36] CROW and ASVV. Recommendations for Traffic Provisions in Built-up Areas. Technical report, 1998. 24

[37] PN Daly, F McGrath, and TJ Annesley. Pedestrian speed/flow relationships for underground stations. *Traffic engineering and control*, 32(2):75–78, 1991. 24

[38] FHWA. *Manual on Uniform Traffic Control Devices*. US Department of Transportation, 1988. 24

[39] BD Hankin and RA Wright. Passenger Flow in Subways. *Operational Research Quarterly*, 9(2):81–88, 1958. 24

[40] LF Henderson. THe Statistics of Crowd Fluids. *Nature*, 229:381–383, 1971. 24

[41] LA Hoel. Pedestrian Travel Rates in Central Business Districts. *Traffic Engineering*, 38:10–13, 1968. 24

[42] Institute of Transportation Engineers. *Transportation and Traffic Engineering Handbook*. Prentice Hall, New Jersey, 1969. 24

[43] H Knoflacher. *Fußgeher und Fahrradverkehr: Planungsprinzipien*. Böhlau Verlag, Vienna, 1995. 24

[44] PM Koushki. Walking Characteristics in Central Riyadh, Saudi Arabia. *Journal of Transportation Engineering*, 114(6):735–744, 1988. 24

[45] HK Lam, JF Morrall, and H Ho. Pedestrian flow characteristics in Hong Kong. *Transportation Research Record*, (1487):56–62, 1995. 24

[46] JF Morrall, LL Ratnayake, and PN Seneviratne. Comparison of Central Business District Pedestrian Characteristics in Canada and Sri Lanka. *Transportation Research Record*, 1294:57–61, 1991. 24

[47] FPD Navin and RJ Wheeler. Pedestrian Flow Characteristics. *Traffic Engineering*, 39:30–36, 1969. 24

[48] CA O'Flaherty and MH Parkinson. Movement on a City Centre Footway. *Traffic engineering and control*, 13:434–438, 1972. 24

[49] SJ Older. Movement of Pedestrians on Footways in Shopping Streets. *Traffic engineering and control*, 10(4):160–163, 1968. 24

[50] J Pauls. Calculating Evacuation Times for Tall Buildings. *Fire Safety Journal*, 12:213–236, 1987. 24

[51] M Roddin. A Manual to Determine Benefits of Separating Pedestrians and Vehicles. Technical report, Transportation Research Board, 1981. 24

[52] AK Sarkar and KSVS Janardhan. A Study on Pedestrian Flow Characteristics. In *Transportation Research Board*, Washington, DC, 1997. 24

[53] RB Sleight. *The Pedestrian: Human Factors in Highway Traffic Safety Research*. Wiley-Interscience, 1972. 24

[54] Y Tanariboon, SS Hwa, and CH Chor. Pedestrian Characteristics Study in Singapore. *Journal of Transportation Engineering*, 112(3):229–235, 1986. 24

[55] Y Tanariboon and JA Guyano. Analysis of Pedestrian Movements in Bangkok. *Transportation Research Record*, 1294:52–56, 1991. 24

[56] PR Tregenza. *The Design of Interior Circulation*. Van Nostrand Reinhold Company, New York, 1976. 24

[57] MR Virkler and S Elayadath. Pedestrian Speed-flow-Density Relationships. *Transportation Research Board*, 1438:51–58, 1994. 24

[58] SB Young. Evaluation of Pedestrian Walking Speeds in Airport Terminals. *Transportation Research Record*, 1674:20–26, 1999. 24

[59] W. Herschel. Experiments on the Refrangibility of the Invisible Rays of the Sun. By William Herschel, LL. D. F. R. S. *Philosophical Transactions of the Royal Society of London*, 90:284–292, 1800. ISSN 0261-0523. doi:10.1098/rstl.1800.0015. 25

[60] Michael Heidler. German infrared driving and fire control equipment. *Small Arms Review*, 19(5), 2015. URL `http://www.smallarmsreview.com/display.article.cfm?idarticles=2981`. 25

[61] Stephen George Porter. Sensors using detector arrays, 1998. 26, 34

[62] Stephen George Porter. Sensors using detector arrays, 2001. 26, 34

[63] Neil Sumpter. New People Counter Communications Protocol. 2003. 26

[64] Anthony Holden. Applications of pyroelectric materials in array-based detectors. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 58(9):1981–1987, 2011. ISSN 0885-3010. doi:10.1109/TUFFC.2011.2041. 26, 31, 32, 33, 34, 35

[65] Mahmoud Massoud. Thermal Radiation. In *Engineering Thermofluids Thermodynamics, Fluid Mechanics, and Heat Transfer*, chapter IVd., pages 561–600. Springer-Verlag, Berlin, Heidelberg, 2005. 26, 28, 30

[66] Claudia Kuenzer and Stefan Dech. Theoretical Background of Thermal Infrared Remote Sensing. In *Thermal Infrared Remote Sensing: Sensors, Methods, Applications*, volume 17, chapter 1, pages 1–26. Springer Netherlands, 2013. ISBN 978-94-007-6638-9. doi:10.1007/978-94-007-6639-6. URL `http://link.springer.com/10.1007/978-94-007-6639-6`. 26, 28

[67] A. Rogalski and K. Chrzanowski. Infrared devices and techniques (revision). *Metrology and Measurement Systems*, XXI(4):565–618, 2014. 26, 27, 34, 36, 37, 38

[68] Ignat Ignatov, Oleg Mosin, and Chavdar Stoyanov. Fields in Electromagnetic Spectrum Emitted from Human Body . Applications in Medicine. *Journal of Health, Medicine and Nursing*, 7:1–23, 2014. 28

[69] Rikke Gade and Thomas B. Moeslund. Thermal cameras and applications: A survey. *Machine Vision and Applications*, 25(1):245–262, 2014. ISSN 09328092. doi:10.1007/s00138-013-0570-5. 30

[70] C. Y. Wong, S. C F Lin, T. R. Ren, and N. M. Kwok. A survey on ellipse detection methods. In *IEEE International Symposium on Industrial Electronics*, pages 1105–1110, Hangzhou, 2012. IEEE. ISBN 9781467301589. doi:10.1109/ISIE.2012.6237243. 39, 40

[71] Sung Joon Ahn, Wolfgang Rauh, and Hans Jürgen Warnecke. Least-squares orthogonal distances fitting of circle, sphere, elipse, hyperbola, and parabola. *Pattern Recognition*, 34(12):2283–2303, 2001. ISSN 00313203. doi:10.1016/S0031-3203(00)00152-7. 40

[72] Ludwig von Seidel. Über den Einfluss der Theorie der Fehler, mit welchen die durch optische Instrumente gesehenen Bilder behaftet sind, und über die mathematischen Bedingungen ihrer Aufhebung. *Abhandlungen der naturwissenschaftlich-technischen Commission der Bayerischen Akademie der Wissenschaften*, pages 289–332, 1856. 41

[73] F Devernay and O Faugeras. Straight lines have to be straight. *Machine Vision and Applications*, 13:14–24, 2001. ISSN 0932-8092. doi:10.1007/pl00013269. 42, 43, 48

[74] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4), 1987. ISSN 0882-4967. doi:10.1109/JRA.1987.1087109. 42

[75] MATLAB R2014b, 2014. URL `http://uk.mathworks.com/products/matlab/`. 45, 109

[76] *EIA standard RS-232-C: Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange.* Electronic Industries Association. Engineering Dept., Washington, DC, 1969. 52

[77] G. Clark. Telnet Com Port Control Option, 1997. URL `https://tools.ietf.org/html/rfc2217`. 53

[78] P.H. Welch and J.M.R. Martin. A CSP model for Java multi-threading. *Proceedings International Symposium on Software Engineering for Parallel and Distributed Systems*, pages 114–122, 2000. doi:10.1109/PDSE.2000.847856. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=847856`. 55

[79] Peter H Welch, J R Aldous, and J Foster. JCSP.net - A Network Extension for JCSP. In *PDPTA '02: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 318–324, 2002. 55

[80] Sarah Clayton and Jon M Kerridge. Active Serial Port: A Component for JCSPNet Embedded Systems. In Ian R East, David Duce, Mark Green, Jeremy M R Martin, and Peter H Welch, editors, *Communicating Process Architectures 2004*, pages 85–98, sep 2004. 55

[81] R.R. Sillito and R.B. Fisher. Semi-supervised learning for anomalous trajectory detection. In *Proc. BMVC*, pages 1035–1044, 2008. URL `http://www.anc.ed.ac.uk/dtc/administrator/components/com_publish/files/publications/sillitobmvc08.pdf`. 59, 61, 68, 110

[82] R.R. Sillito and R.B. Fisher. Parametric Trajectory Representations for Behaviour Classification. In *Proc. BMVC*, volume 12, pages 227–238, 2009. URL `http://scholar.google.com/scholar?hl=en&btnG=Search&`

q=intitle:Parametric+Trajectory+Representations+for+Behaviour+Classification#0. 59, 60, 110

[83] Michael Unser, Akram Aldroubi, and Murray Eden. B-Spline Signal Processing: Part I-Theory. *IEEE Transactions on Signal Processing*, 41(2):821–832, 1993. 60

[84] Michael Unser, Akram Aldroubi, and Murray Eden. B-Spline Signal Processing: Part II-Efficient Design and Applications. *IEEE Transactions on Signal Processing*, 41(2):834–848, 1993. 60

[85] Robert Fischer. CAVIAR: Context Aware Vision using Image-based Active Recognition, 2001. URL http://homepages.inf.ed.ac.uk/rbf/CAVIAR/. 61

[86] Eric W Weisstein. French Curve., 1997. URL http://mathworld.wolfram.com/FrenchCurve.html. 61

[87] Alistair Townsend. On the spline: a brief history of the computational curve. *International Journal of Interior Architecture and Spatial Design*, 3, 2014. URL http://www.alatown.com/spline-history-architecture/. 61, 62, 63

[88] Gerald Farin. A History of Curves and Surfaces in CAGD. In Gerald Farin, J Hoscheck, and MS Kim, editors, *Handbook of CAGD*, pages 1–22. Elsevier, 2002. 62, 64

[89] Teun Koestsier. Ludwig Burmester (1840-1927). In Marco Ceccarelli, editor, *Distinguished Figures in Mechanism and Machine Science*, volume 7 of *History of Mechanism and Machine Science*, pages 43–64. Springer Netherlands, 2010. ISBN 978-90-481-2345-2. doi:10.1007/978-90-481-2346-9_3. URL http://link.springer.com/10.1007/978-90-481-2346-9_3. 63

[90] Les Piegl and Wayne Tiller. *The NURBS Book.* Springer-Verlag Berlin Heidelberg, 1995. ISBN 978-3-642-97385-7. doi:10.1007/978-3-642-97385-7. 64

[91] Carl de Boor. *A Practical Guide to Splines.* Springer-Verlag, revised edition, 2001. 65, 66, 67, 68, 72, 73, 80, 81

[92] Qiujuan Tong, Sanyang Liu, Quan Lu, and Junfeng Chai. A fast algorithm of Moore-Penrose inverse for the symmetric Loewner-type matrix. In *International Conference on Information Engineering and Computer Science, 2009. ICIECS 2009.*, pages 0–3, Wuhan, 2009. IEEE. ISBN 9781424449941. doi:10.1109/ICIECS.2009.5364014. 70

[93] Joe Hicklin, Cleve Moler, Peter Webb, Ronald F. Boisvert, Bruce Miller, Roldan Pozo, and Karin Remington. JAMA, 2012. URL http://math.nist.gov/javanumerics/jama/. 70

[94] N Bronshtein, A Semendyayev, G Musiol, and H Muehlig. *Handbook of Mathematics.* Springer-Verlag Berlin Heidelberg New York, fifth edition, 2007. ISBN 9783540721215. 79

[95] Yannis Theodoridis and Nikos Pelekis. *Mobility Data Management & Exploration*. Springer, 2012. ISBN 978-1-4939-0391-7. doi:10.1007/978-1-4939-0392-4. 83, 89, 113

[96] Sunglok Choi, Taemin Kim, and Wonpil Yu. Performance Evaluation of RANSAC Family. *Procedings of the British Machine Vision Conference 2009*, pages 81.1–81.12, 2009. doi:10.5244/C.23.81. URL `http://www.bmva.org/bmvc/2009/Papers/Paper355/Paper355.html`. 85, 88

[97] Ramsey Faragher. Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]. *IEEE Signal Processing Magazine*, 29 (5):128–132, 2012. ISSN 10535888. doi:10.1109/MSP.2012.2203621. 85, 86, 87

[98] Andrea Vedaldi, Hailin Tin, Paolo Favaro, and Stefano Soatto. KALMANSAC: Robust filtering by consensus. *Proceedings of the IEEE International Conference on Computer Vision*, I:633–640, 2005. ISSN 1550-5499. doi:10.1109/ICCV.2005.130. 88

[99] Serge P Hoogendoorn and Winnie Daamen. Free Speed Distributions for Pedestrian Traffic. In *TRB 2006 Annual Meeting*, volume CDROM, 2006. URL `http://katana.hsrc.unc.edu/cms/downloads/FreeSpeedDistributionsforPedestrianTraffic.pdf`. 94, 95

[100] Wolfgang Koch. *Tracking and Sensor Data Fusion*. Springer Berlin Heidelberg, 2014. ISBN 9783642392702. URL `http://link.springer.com/content/pdf/10.1007/978-3-642-39271-9.pdf`. 107, 108

[101] Zbigniew Kotulski and Wojciech Szczepinski. *Error Analysis with Applications in Engineering*. Springer, 2010. ISBN 9789048135691. 108

[102] B.T. Morris and M.M. Trivedi. A Survey of Vision-Based Trajectory Learning and Analysis for Surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(8):1114–1127, aug 2008. ISSN 1051-8215. doi:10.1109/TCSVT.2008.927109. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4543858`. 110

[103] Pau Baiget, Eric Sommerlade, Ian Reid, and Jordi Gonzàlez. Finding prototypes to estimate trajectory development in outdoor scenarios. In *Proceedings of the First International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS2008)*, 2008. URL `http://iselab.cvc.uab.es/themis2008/ProceedingsTHEMIS2008.pdf#page=35`.

[104] David Ellis, Eric Sommerlade, and Ian Reid. Modelling pedestrian trajectory patterns with Gaussian processes. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1229–1234. Ieee, sep 2009. ISBN 978-1-4244-4442-7. doi:10.1109/ICCVW.2009.5457470. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5457470`. 110

[105] Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Computational movement analysis. In *Springer Handbook of Geographic Information*, pages 725–741. 2012. ISBN 978-3-540-72678-4. doi:10.1007/978-3-540-72680-7_22. 113

[106] Nikos Pelekis and Yannis Theodoridis. Hermes. URL `https://hermes-mod.java.net/`. 113

[107] A Seyfried, M Chraibi, U Kemloh, J Mehlich, and A Schadschneider. Runtime Optimization of Force Based Models within the Hermes Project. In Richard D Peacock, Erica D Kuligowski, and Jason D Averill, editors, *Pedestrian and Evacuation Dynamics*, pages 363–373. Springer US, 2011. ISBN 978-1-4419-9724-1. doi:10.1007/978-1-4419-9725-8_33. URL `http://dx.doi.org/10.1007/978-1-4419-9725-8_33`. 113

[108] Joachim Gudmundsson, Marc Kreveld, and Bettina Speckmann. Efficient Detection of Patterns in 2D Trajectories of Moving Points. *GeoInformatica*, 11 (2):195–215, jan 2007. ISSN 1384-6175. doi:10.1007/s10707-006-0002-z. URL `http://www.springerlink.com/index/10.1007/s10707-006-0002-z`. 113

[109] Patrick Laube and Stephan Imfeld. Analyzing relative motion within groups of trackable moving point objects. In *Geographic information science*, pages 132–144. 2002. ISBN 978-3-540-44253-0, 978-3-540-45799-2. doi:10.1007/3-540-45799-2_10. 113