Abstract

Autonomous, Collaborative, Unmanned Aerial Vehicles for Search and Rescue

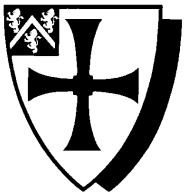
Andrew Michael Luke Ambrose-Thurman

Search and Rescue is a vitally important subject, and one which can be improved through the use of modern technology. This work presents a number of advances aimed towards the creation of a swarm of autonomous, collaborative, unmanned aerial vehicles for land-based search and rescue. The main advances are the development of a diffusion based search strategy for route planning, research into GPS (including the Durham Tracker Project and statistical research into altitude errors), and the creation of a relative positioning system (including discussion of the errors caused by fast-moving units). Overviews are also given of the current state of research into both UAVs and Search and Rescue.

Autonomous, Collaborative, Unmanned Aerial Vehicles for Search and Rescue

by

Andrew Michael Luke Ambrose-Thurman



Submitted in conformity with the requirements
for the degree of Doctor of Philosophy

School of Engineering and Computing Sciences
University of Durham

Contents

List of Tables

List of Figures

Glossary

A-GPS ..... Assisted GPS
ALSAR ..... Association of Lowland Search and Rescue
AOA ..... Angle of Arrival
APRS ..... Automatic Packet Reporting System
ARCC ..... Aeronautical Rescue Coordination Centre
BCRC ..... British Cave Rescue Council
C/A Code ..... Coarse/Acquisition GPS code
CA ..... Cellular Automata
CCPP ..... Complete Coverage Path Problem
COTS ..... Commercial Off The Shelf
DARPA ..... (US) Defense Advanced Research Projects Agency
DEM ..... Digital Elevation Model
DGPS ..... Differential GPS
ED50 ..... European Datum, 1950 (based on the Hayford Ellipsoid 1909, aka International Ellipsoid 1924)
EGM96 ..... Earth Geodetic Model
ELT ..... Emergency Locator Transmitter, for aircraft use
EPIRB ..... Emergency Position Indicating Radio Beacon, for maritime use
ETRS89 ..... European Terrestrial Reference System, 1989 (based on GRS80)
FAA ..... Federal Aviation Authority
GEOSAR ..... Geostationary Earth Orbit Search and Rescue
GNSS ..... Global Navigation Satellite System
GPRS ..... General Packet Radio Service

GPS ..... Global Positioning System
GPX ..... GPS Exchange Format
GRS80 ..... Geodetic Reference System, 1980
GSM ..... Global System for Mobile Communications
HTTP ..... Hypertext Transfer Protocol
I^2C ..... Inter-Integrated Circuit (two-wire interface)
IMEI ..... International Mobile Equipment Identity
IMU ..... Inertial Measurement Unit
INS ..... Inertial Navigation System
IQR ..... Interquartile Range
IR ..... Infrared
ISRID ..... International Search and Rescue Incident Database
JTAG ..... Joint Test Action Group (IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture)
LED ..... Light Emitting Diode
LEOSAR ..... Low Earth Orbit Search and Rescue
MAV ..... Micro Aerial Vehicle
MCC ..... Mission Control Centre
MM ..... Maximum-minimum range
MoD ..... Ministry of Defence
MRCofS ..... Mountain Rescue Committee of Scotland
MREW ..... Mountain Rescue England and Wales (formerly the Mountain Rescue Council of England and Wales)
MSP ..... MSP430F169 - See Section 5.3.1.2
NASA ..... US National Aeronautics and Space Administration
NMEA .. National Marine Electronics Association (Commonly, the format for raw data provided by GPS devices, amongst others)
NTP ..... Network Time Protocol
OSGB36 ..... Ordnance Survey of Great Britain, 1936 revision (based on the 1830 Airy ellipsoid)
OSM ..... OpenStreetMap

P(Y) Code ..... (Encoded) Precision GPS code
PCB ..... Printed Circuit Board
PLB ..... Personal Locator Beacon
POA ..... Probability of Area
POC ..... Probability of Containment (AKA Probability of Area)
POD ..... Probability of Detection
POS ..... Probability of Success
PPS ..... Precise Positioning Service
RAF ..... Royal Air Force
RBS ..... Reference Broadcast Synchronization
RPA ..... Remotely Piloted Aircraft (UAV with a human in the loop)
RTK ..... Real Time Kinematic
SA ..... Selective Availability
SAR ..... Search and Rescue
SARDA ..... National Search and Rescue Dogs Association
SARF ..... RAF Search and Rescue Force
SIM ..... Subscriber identity module
SMS ..... Short Message Service
SPI ..... Serial Peripheral Interface Bus
SPS ..... Standard Positioning Service
SRTM ..... Shuttle Radar Topography Mission
TDOA ..... Time Difference of Arrival
TOA ..... Time of Arrival
TOF ..... Time of Flight
TRF ..... Terrestrial Reference Frame
UART ..... Universal Asynchronous Receiver/Transmitter
UAS ..... Unmanned Aircraft System (UAV plus ground stations, communications network, etc.)
UAV ..... Unmanned Aerial Vehicle
VDOP ..... Vertical Dilution of Precision
WAAS ..... Wide Area Augmentation System

WGS84 ..... World Geodetic System, 1984 (based on GRS80) — A best-fit global ellipsoid, used by GPS

WiSAR ..... Wilderness Search and Rescue

WPS ..... WiFi Positioning System

Y Code ..... Encoded P code

Declaration

The material contained within this thesis has not previously been submitted for a degree at the University of Durham or any other university. The research reported within this thesis has been conducted by the author unless indicated otherwise.

Copyright Notice

In memoriam Val Vitanov

Acknowledgements

First and foremost to Alice, for putting up with me while I frantically finished writing up, and for walking very slowly very consistently.

To my supervisors, Alan Purvis, Val Vitanov, and Peter Baxendale, and my examiners.

To my family and friends — those who offered to carry trackers; those who helped to proofread; those who gave ideas and advice; and, just as much, all the others — you've all been a great help, and you've all helped me enormously to get through the last few years, whether you realise it or not.

Chapter 1

Introduction

An experienced walker has gone on holiday with his dog, walking in the highlands of Scotland, and one morning leaves the hostel for a long, all day walk. They've taken a relatively well walked trail, with a reasonably clear path, but there aren't many other walkers — it's out of season and the weather's turned to drizzle. At one point the path passes through an area of woodland, halfway up a mountain. The dog suddenly notices a rabbit, and gives chase. The walker catches the dog again without too much trouble, but in the few minutes he's spent away from the path he's lost his way, and in trying to find it again he slips on the wet leaves and injures his leg. There is no mobile phone reception, and although he left word at the hostel to say where he'd gone no one knows exactly when he was planning to return, and so the alarm isn't raised until quite late at night.

As someone who enjoys walking in remote countryside myself, this is a situation which I hope to never experience. For the people left behind it is of vital importance that the missing person is found quickly, and it can be a matter of life and death how long it takes.

Search and Rescue in the UK — under the overall control of the Department for Transport — is split between a number of different groups.[1] Incidents which occur at sea fall under the remit of the Coastguard; incidents which occur on land fall under the remit of the Police; while the RAF's Aeronautical Rescue Coordination Centre (ARCC) responds to calls for assistance from both the Coastguard and the

Police, coordinates all search and rescue aircraft (RAF, Navy, Coastguard, Mountain Rescue, etc.), and (as the UK Mission Control Centre) monitors and responds to emergency beacons on the Cospas-Sarsat system which fall within the UK Search and Rescue Region (see Appendix A).[1]

While the police have overall responsibility for land-based search and rescue, the actual incidents are in many cases covered by different organisations — particularly in more specialised cases, such as on mountains. In the case of urban search and rescue, where people need to be found in collapsed buildings, the Fire Service is usually used. The RAF Search and Rescue Force (SARF) — of which the ARCC is a part — also contains the RAF Mountain Rescue Service. This is able to provide search and rescue in mountainous terrain. There are also numerous volunteer search and rescue organisations, which form umbrella groups such as Mountain Rescue England and Wales (MREW, formerly the Mountain Rescue Council of England and Wales), the Mountain Rescue Committee of Scotland (MRCofS), the British Cave Rescue Council (BCRC), the Association of Lowland Search and Rescue (ALSAR), and the National Search and Rescue Dogs Association (SARDA), amongst others.

If a person is reported missing to the police a search is likely to take place. Volunteers can be alerted in a short period of time, and will soon be on the scene to assess the situation and decide how to split up the available resources to perform the search.

Sometimes the reason for a person going missing is that they were in a hazardous environment. Whether this is a potholer being injured by falling in a cave, or a walker being caught by floodwater or an unexpected blizzard — sending more people on foot into that same environment can be putting them at risk. It is therefore of great importance that the correct decisions are made for the search plan.

As mentioned above, if any aircraft are being used in the search and rescue effort then they are coordinated via the RAF's Aeronautical Rescue Coordination Centre. As part of the research for this thesis a visit was made to the ARCC base at RAF Kinloss, including interviews with some of the personnel. This covered many aspects of search and rescue, including search strategy, the creation of probability

[1] This only covers civilian search and rescue; military search and rescue comes under the remit of the MoD

maps, the Cospas-Sarsat system, and the potential use of UAVs in search and rescue operations. Some of the information obtained is covered in Chapter 2, and the visit has helped to shape much of this work.

Aircraft are of great use in a search and rescue mission.[2] They give a birds eye view of the situation. While they are unable to search through the undergrowth, they complement ground based searchers by being able to cover the ground at a far greater speed to look for larger clues — a body, the remains of a fire, a landslip, etc. They are also able to provide transport — either to move personnel (search teams, sniffer dogs, paramedics, etc.) to areas more quickly, or to move injured persons back to safety.

They have some associated problems, however. The nearest search and rescue helicopter to a given situation might be some distance away, meaning that precious time is wasted. Their clearance is limited, so that in many cases they cannot approach the ground too closely. The helicopters are expensive to run, and it would be unlikely that multiple aircraft are summoned to one incident. Manned aircraft are unable to fly indefinitely; the people inside them get tired and must stop. They are also unable to fly in adverse weather conditions — in snow, high winds, fog, etc. This is particularly true of civilian search and rescue aircraft, which have more restrictive flying conditions.

There are also limitations to the effectiveness of ground based searchers — primarily from their often limited numbers, their relatively low speed when on foot, and their personal safety.

Some of these problems may be overcome by using a swarm of autonomous, collaborative unmanned aerial vehicles (UAVs).[2] Smaller than manned helicopters, these would be cheaper to run, able to run for longer periods, and would be able to fly considerably closer to the ground, including through areas with many obstacles (such as sparse woodland). While such a swarm would be of limited help with the rescue part of the operation they would greatly ease the load on human searchers.

Land-based search and rescue has had considerably less research than other forms of search and rescue. Many of the ways in which search is carried out, particularly

[2]Flying robots; see Chapter 4 for more

by volunteer organisations, suffer from inefficiencies. Technology is not being used as well as it might. In particular, new advances in robotics in general and in UAVs in particular might cause great increases not only in the speed at which missing people might be found but also in the safety of the human searchers.

The main overarching thesis for this work is therefore:

> **Land-based Search and Rescue can be substantially improved through increased use of technology, and through use of UAVs in particular.**

Given the above discussions, this work will be tackling the following themes:

> **1. Improving algorithms — Current search strategy is suboptimal for use with a swarm of collaborative UAVs, and can be improved upon**

This is mainly covered in Chapter 3, which looks at the use of a diffusion based algorithm for calculating routes for UAVs to optimise their searching of high probability areas faster than lower probability ones.

> **2. Improving hardware — Autonomous, collaborative UAVs for search and rescue could be developed which would improve the work of SAR groups**

This is touched upon in many sections (including an overview of UAV technology in Chapter 4), but most particularly in Chapter 5. This discusses the creation and modification of a GPS tracker, both as part of an autonomous UAV control system and as a standalone tracker which could be used by search and rescue teams.

> **3. Improving localisation — UAVs require accurate positioning in order to successfully complete their tasks, and GPS can be improved upon in some cases**

GPS errors are covered in Chapter 6, and Chapter 7 describes the creation of a relative positioning system for fast-moving UAVs when GPS is unavailable.

The remainder of this thesis is organised as follows:

Chapter 2 will cover search and rescue, focussing on search strategy for land-based wilderness search and rescue but also covering other forms. Chapter 3 will then describe a diffusion based method for improving upon current search and rescue methods, comparing a variety of algorithms for plotting a search path across a probability map. Chapter 4 will document the current situation of UAVs (of varying levels of autonomy), covering a number of research groups but focussing on work related to disaster management and search and rescue. This is followed by Chapter 5, where the Durham Tracker project is documented. This tracker was intended to be connected to a UAV control system to provide both a position to the UAV and reports back to a control centre, and can also be used for a number of other relevant applications — including personal safety for walkers, or plotting the areas covered by searchers. Chapter 6 talks about the errors inherent with GPS, and aims to quantify the altitude error range by comparison of a large number of GPS altitudes against another dataset. Finally Chapter 7 covers the creation of a relative positioning system for UAVs and researches the effects of their rapid motion on standard equations.

Chapter 2

Search and Rescue

2.1 Introduction

Search and Rescue (SAR) can be split into three broad categories — land, sea and air. Search in the air is, usually, extremely simple; if something is in trouble and in the air, it will very soon be on sea or land.[1] Search in the sea is for the most part very logical; the tides and currents can be calculated to some degree of accuracy for any given place and time, and most floating things that are in trouble will move in a way that can be computed.[2] Land-based search and rescue, by comparison, is considerably harder. A missing person on the land will be moving by means that cannot be calculated: they might be walking; they might have fallen down a cliff; they might have been kidnapped; they might have caught the bus — or they might just be sitting in a pub.

The field of Search and Rescue was first formalised during WWII by B. O. Koopman[3] while working for the US Navy. His search methods were originally designed for use in searching for enemy submarines, but the underlying theory[3] is applicable to any search situation — whether finding electronic signals, diseases, or (as in this case) missing persons.

Traditional search methods for search and rescue can be split into two sections.

[1] With some exceptions, such as out of control balloons, etc.
[2] This is a simplification — see Section 2.2.2.1
[3] Search theory is part of the broader topic of Operations Research[4]

The first is to produce a probability map for the missing person — displaying on it which areas are most likely to contain the person and conversely which are less urgent to search. Secondly, once this has been created, the resources available can be assigned to search across this probability map.

The remainder of this thesis gives a brief overview of search theory and of search and rescue.

2.2 Creating probability maps

There are three important terms which need to be understood for working out where best to search. They are Probability of Area (POA),[4] Probability of Detection (POD), and Probability of Success (POS). For a given area:

POA is the probability that the missing object is inside this area.[6]

POD is the probability — for a given object and a given searcher / sensor — that the missing object will be detected, **providing** that the object is inside this area.

POS is the combined probability that searching this area for this object with this searcher will result in the missing object being found.[7] POS is calculated using the formula

$$POS = POA \times POD \tag{2.1}$$

The value of POS for a given area allows you to decide where is most sensible to search first.[3] Searching the area with the highest POS means that you have the highest probability of finding the object, as the probabilities both of the area containing the object and of the object being found are maximised. This was initially used in military search situations, for example by the US military to find sunken ships and lost hydrogen bombs, but is highly applicable for search and rescue.

[4] Also known as Probability of Containment (POC)[5]

2.2.1 Calculating POD

The POD is dependent on a large number of factors.[8] If you are searching for a person then you will have a greater probability of detecting them than you will of detecting a small clue, such as a cigarette butt. You will have a higher POD in terrain which is flat with no coverage (say, a tarmaced playing field) than in terrain with hidden hollows and large amounts of undergrowth (for example in dense woodland). Speed will also have an effect; driving past an object in a car will give less time to see and recognise it than walking past slowly.

A significant factor in estimating a POD is the sensors being used. Without going into detail with the underlying mathematics, here follows a brief overview of how this is calculated.. If a human is walking through an area they might have a near certain probability of detecting the object if it is directly in their path; this probability might drop off as the object lies further away from the search path, so that a few metres away the object had only a 50% chance of being seen, then a little further only a 25% chance of being seen, and finally at a certain distance no chance of being seen. The same path could be taken by a low flying UAV, overhead. This might hypothetically have a camera which was able to detect the object with near certainty if it fell anywhere within its field of view — but with a zero probability outside of it. For both of these a probability distribution could be plotted, showing the likelihood of detection at a variety of distances. This, however, is only able to give a probability for a single view of the object at the point at which the path comes closest to it. In order to allow for the possibilities of seeing the object when the searcher is further away along the path (where the probability of detection is lower, but the amount of time spent at that distance is higher) it is possible to integrate along the route. Doing this for all object distances gives a Lateral Range Curve — the probability of detecting the object at some point during the search along that route, based on its distance from the path.

The Lateral Range Curve makes several assumptions, however, such as that the object is unmoving and the path taken by the searchers does not overlap. A more realistic metric is the Sweep Width, which is the area underneath the Lateral Range Curve. This represents the number of objects that would be found if passing

through an area filled with randomly placed objects. In this case, the aircraft may have the same sweep width as the human — the former would find every object it passed which was within a certain distance of the path; the latter would detect a lower proportion of the objects, but over a wider distance from the path, giving the same overall total. In real life situations, where there is path overlap and the object being detected cannot be assumed stationary, the Sweep Width gives a more realistic quantity for calculating POD.[9; 7; 5]

There are several different algorithms which can be used for converting between sweep width and POD. The most useful is the exponential or 'random' detection function, which gives a minimum POD for the effort expended and does not require perfectly straight, parallel sweeps across the area (which is unrealistic in most real life situations, especially with moving targets where the frame of reference may be shifting[10; 4]):

$$POD = 1 - e^{-\frac{vWt}{A}} \tag{2.2}$$

where W is the effective sweep width of the sensor used, v is the average velocity, t is the time taken to search the area, and A is the area size.[3; 11]

Much research[5] has been done on attempting to quantify sweep widths for a range of searchers and sensors — including ground based searchers,[9; 12; 8] aerial sweeps,[13] and UAVs.[14]

2.2.2 Calculating POA

The POA is the probability that a given area actually contains the missing person. After it has been set initially it needs to be regularly updated based on changes; whether that is new evidence, or the area having been searched and nothing found. In this latter case the new values are calculated using Bayes theorem. Bayesian Search Theory has been in use for almost 50 years for locating items, initially for objects lost at sea (such as a missing hydrogen bomb lost off the coast of Spain and a missing US submarine, the Scorpion).[11]

Given that sensors are never perfect, it is almost always possible that despite

[5] Much of which, sadly, is mathematically incorrect — Cooper et al. 11

being searched an area still contains the missing person. This means that there are two possibilities — given that nothing was found, either the person wasn't there, or they were. The prior probability that they were in that area (the previous POA) is known, as is the probability that they would be discovered if they were (POD), so the new POA can be worked out.[15]

Bayes theory states that

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{2.3}$$

For the area searched:

$$P(C \mid N) = \frac{P(N \mid C)P(C)}{P(N)} \tag{2.4}$$

where $P(C)$ is the prior probability that the area contained the person (the previous POA), and $P(N)$ is the prior probability that the person wasn't detected in the area. POD — the probability that the person would be detected given that they are in the area — would be $P(\bar{N} \mid C)$.

$$POA_{new} = \frac{(1 - POD) \times POA_{old}}{P(N)} \tag{2.5}$$

$$P(N) = P(N \mid C)P(C) + P(N|\bar{C})P(\bar{C}) \tag{2.6}$$

$$P(N|\bar{C}) = 1 \tag{2.7}$$

$$P(N) = (1 - POD) \times POA_{old} + (1 - POA_{old}) \tag{2.8}$$

$$P(N) = 1 - POD \times POA_{old} \tag{2.9}$$

$$POA_{new} = \frac{(1 - POD) \times POA_{old}}{1 - POD \times POA_{old}} \tag{2.10}$$

The probabilities of all other areas need to be updated as well, to reflect the new information — showing that they are now comparatively more probable, now that a search elsewhere has proved fruitless.

For each other area x:

$$P(C_x \mid N_S) = \frac{P(N_S \mid C_x)P(C_x)}{P(N_S)} \tag{2.11}$$

where $P(C_x)$ is the prior probability that area x contained the person (the previous POA for that area), and $P(N_S)$ is the prior probability that the person wasn't detected in the area which has just been searched. $P(N_S \mid C_x)$, the probability that the person wouldn't be found in the area searched if they are actually in this different area, will logically be 1.

$$POA_{x,new} = \frac{POA_{x,old}}{P(N_S)} \tag{2.12}$$

where $POA_{x,new}$ is the new POA for the area x. It should be noted that this has the same denominator as Equation 2.5 — the calculations required can therefore be simplified by not performing this normalisation step, as all relative probabilities will be unchanged. This means that, after a search, the areas not searched can retain their old POA and the searched area changes its POA to:

$$POA_{new} = (1 - POD) \times POA_{old} \tag{2.13}$$

Not normalising gives a number of advantages, including a reduction in the amount of processing required (only one area needs updating at a time rather than every area on the map), and an increase in the amount of useful information that can be obtained — if all areas are normalised (such that the POAs from all areas sum to 100%) then there is no way of knowing when the overall probability of the areas under consideration drops to such an extent that the missing person must be outside those areas. Either the search must be expanded, or the search suspended — there is no point in continually searching an area when it is unlikely that it contains the person.[11]

Note that calculating POA using the algorithms above may lead to a searched area that was found to not contain the person having a higher POA than an entirely unsearched area. This is logical — an area which is very likely to contain the person but which is very hard to search effectively is going to remain likely for a long time, while an area that is unlikely to begin with will remain unlikely, regardless of

whether or not it has been searched.

The creation of the initial land-based probability maps is covered in Section 2.2.2.2, after a brief description of marine probability map creation for comparison.

2.2.2.1 Sea

The location of an object lost at sea depends on several factors, but one of the most important ones is whether or not it is under human control. Ships which have failed to arrive may have left their predicted course at any point, and might have done so intentionally and under full control. The movement of a floating object, however, can to some extent be modelled mathematically. This is extremely useful, as in most cases an object requiring search and rescue will not be in control of their own movements; e.g. a life raft, floating away from a sinking ship.

A life raft (or similar floating object) will drift predominantly under environmental factors — wind, tides, current, etc.[11] — and the survivors themselves often have little or no control over their course by comparison. All of the main effects can therefore be calculated from known information, such as the weather, and information from systems available worldwide to accurately calculate currents.

Once the data has been collated a Monte Carlo simulation can be performed. Particles are placed at the last known position of the missing object, moved based on drift estimates, and their accumulated positions give a set of probabilities for where the actual object may be.[16]

Much more could be written on marine search and rescue, but it lies outside the scope of this thesis.

2.2.2.2 Land

There are various ways of defining land SAR areas. Koester[6], for example, splits them into wilderness, rural, suburban, urban, and water. In this thesis we are focussing on Wilderness, which is also referred to as Mountain, and to some extent on Rural. Goodrich et al.[17] describes wilderness search and rescue (WiSAR) as 'finding and providing assistance to humans who are lost or injured in mountains, deserts, lakes, rivers, or other remote settings'. Koester describes Wilderness and

Rural respectively as 'An area essentially undisturbed and uninhabited by humans. Although even in a wilderness area a network of trails and roads may be found', and 'An area relating to the countryside, marked by farming or raising livestock, which is sparsely populated. It is often mixed with open spaces and woods.'.

The search and rescue techniques required for this terrain are notably different from others. In Urban SAR,[6] for example, any missing persons would be either inside the damaged building or they would be safe — and although some areas are likely to be more fruitful to search than others (for example bedrooms in a house late at night, or offices in an office building during the day), these decisions are based more on the types of area and less on the types of individual.

The probability map for a missing person on land is calculated from a range of information — including expert opinion from people who know the area, historical search statistics, and possible speed of travel over the terrain, amongst others.

On land a missing person's movement is an unknown; they may move or they may stay still, and the direction, speed and distance are all based on a range of aspects which are hard to pin down — such as the person's state, condition, age, etc. Fortunately some of this information can be obtained from statistics gathered from historical searches.

There is a large body of statistical knowledge available relating to land-based search and rescue, collated by organisations in many different countries. This information is split into missing person types. The large variation in behaviour of missing persons depending on their type is significant — a hiker will behave very differently to a child, or to someone who is suicidal, or to someone who has dementia — and the results for each person type as shown by each country are very similar.

The importance of this data is summed up by Perkins et al.[18]: 'It is always useful to tell the search party prior to deployment how the missing person is likely to behave. How will they react to being the subject of a search? Will they try to hide, or to attract attention to themselves? If they went missing deliberately, how will they try to remain undetected? If they are lost unintentionally, how are

[6] 'Urban', in this context, is being used to denote searching for missing people in buildings, usually after some form of disaster. In other contexts it can refer to people who are missing in a city

they likely to react, what actions will they take to remedy the situation?' — the more information available to searchers, the better equipped they will be to find the missing person.

There are many organisations collating SAR statistics in different countries. In the UK both the Police and the Mountain Rescue Council of England and Wales produce statistics, and the latter releases regular reports — the latest of which was published in 2011.[19] Individual countries have only limited amounts of data to produce statistics from, however. One of the best methods for obtaining statistics, therefore, is to use the International Search and Rescue Incident Database (ISRID).[7]

ISRID was started in America in 2002 on the back of a research grant from the US Department of Agriculture, building on many previous sets of statistics,[6] and now contains data collated from more than 40 countries. By 2008 the database contained over 50,000 SAR incidents, and it is still growing.

The ISRID database first splits the missing person into a category (e.g. Child (with different categories for different ages), Hiker, Climber, Despondent — see Appendix B for a full list), and then further divides the statistics by area type (temperate, dry or urban). For this more specific scenario it provides a number of statistics, including probability of different distances found from initial point; elevation change (probabilities of the person moving up, down, or neither from the initial point); mobility in hours; dispersion angle (how far they are likely to spread from a straight line); common locations where found (e.g. by water, in woods, on roads, etc.); survivability (how likely they are to be found alive, for various lengths of time); distance travelled away from a track; and finally any other useful information (such as that people suffering from dementia will avoid crossing boundaries (roads, fences, etc.)).

Used properly, these statistics can provide the basis for a probability map. It is important to note, however, that these statistics don't mean that all similar situations will be exactly the same — they're just a good place to start, and making too many assumptions can be dangerous. As Koester[6] said, 'The most dangerous planner is one who has seen a situation once or twice and tends to think all future

[7] Into which both the Police and MREW submit data, along with similar organisations in many other countries

subject behavior will be just like the last one'.

One of the most useful statistics is the distance at which the missing person is usually found. This allows a series of concentric circles to be plotted on a map, showing the differing levels of probability that the person is within each circle. This is known as the Bicycle Wheel Method. A missing person is most likely to be found within the 'hub' of the wheel. They are almost certain to be found in the area covered by the entire wheel, although that may be a very long distance from the hub. Radially out from the centre of the wheel are 'spokes' — natural pathways that the missing person may have followed. Depending on the other statistics, the search teams will probably want to concentrate initially on these paths within the 'hub', then fill in the areas between them, and then repeat for the remainder of the wheel.

It should be noted that statistics do not alone make up a new probability map. A lot of weight is given to the opinions of experts, who come up with a variety of scenarios for what might have happened to the missing person based on the available evidence. These scenarios are likely to be based on the statistics, but may entirely disregard some as not being applicable or important. The opinions of many experts can be collected, and an average weighting for each search area obtained using a Mattson Consensus. It is these opinions which create the probability maps, but it is the statistics (along with personal experience of the area and any unusual factors, such as the weather) which inform their opinions.

2.2.2.3 Criticism

Marine search and rescue has taken classical search theory into account for a long time, but land-based search for missing persons has in the past lagged behind somewhat. The probabilities for land-based SAR are affected by a wide range of sources and their calculations have often been misunderstood by researchers, leading to many people conducting plausible but ultimately incorrect studies in this field.

Because of this, the US Department of Homeland Security and US Coastguard commissioned a review of land-based search strategy due to the lack of 'a comprehensive attempt to apply the science of search theory to the development of land

search planning and techniques'.[11] Published at the end of December 2003, this extensively reviewed previous methodology and provided a framework for the areas requiring research. Although this and similar papers produced much controversy at the time,[20] since that point there has been much work carried out to bring land-based searches in line with search theory. This history of improper research, however, has marred the opinion of some groups of current land search theory.

In mid 2009, as part of the research for this thesis, some questions were put to Mark Harrison (at the time the National Police Search Advisor). Regarding the body of research produced internationally by volunteer organisations, he questioned the relevance and reliability of their datasets — in particular, the use of data from other countries for searches in the UK — and hence the validity of the conclusions that can be drawn from their work. The UK police have conducted extensive (unpublished) research into their own data, profiling by behaviour instead of by activity. This research is used by the police both when searching for missing persons and when searching for criminals.

Despite such criticism, the historical data used by volunteer groups around the world (profiling by activity) is in everyday use[6] and there is much research based around it. The ISRID database does include some data provided by the British police;[21] similarly, if the use of international statistics was problematic, the statistics from Mountain Rescue (England and Wales) are also available.[19]

2.2.3 Division of areas

Traditional land SAR is based around the abilities of the searchers. Creating a probability map would typically start by splitting the area into sections, each of which is sized so as to be searchable by a team in one search period.[11] The areas need not be regular, and need not all be the same size. A search period might last anywhere between a few hours and a day. Between search periods the teams would all report their findings back to base, be given their next area, and travel to it ready to begin searching again. The teams are assigned so as to maximise the probability of success (POS), and after each search period the probability of areas are recalculated in the light of their findings (see Section 2.2.2).

This method does have some problems. Having the search area boundaries specified in terms of the search party means that they bear little relation to the actual search probabilities of the missing person — an area may actually have a range of probabilities across it but only be given one POA.

One way in which this could be solved would be by splitting the search area into smaller segments; reducing the search areas to a minimum, for example a metre square. This would allow for faster area searches, with search teams being allocated multiple search regions to cover (either being given a route across them or being left to decide for themselves how to cover them). If a POA covering a large area changed such that some parts were now at a very different level of probability to the rest, this could be more easily updated without having to split up regions or modify area boundaries.

This does, however, require some changes in the methods used. Assigning the probabilities may become more time consuming, although the areas covered by each POA can still be large (just covering multiple search regions). Searchers — who before spent most of their time searching, with time for travel and reporting findings being considered negligible — would now be spending most of their time travelling from one search region to the next, with the time taken to search each being considered negligible.

There would also be a problem with the searchers knowing which search region they were in. Large regions with notable boundaries (e.g. a wood, with the boundary being the edge of the wood) are easy to tell when the searcher is inside them; knowing that the searcher is inside a particular square metre in the middle of the wood, on the other hand, would be very difficult.[8] This can, however, be considered as a larger problem — if searchers are unable to tell where they are with any degree of precision then they will be unable to entirely search an area regardless of its size.[22]

In the case of UAVs, having a fine grained probability map would be an advantage. Instead of having search areas based on the abilities of a team of slow-moving human searchers with limited communication, they would have search areas based around movement across the map. They would be able to take a coarser initial

[8]This could in part be solved by issuing the searchers with GPS — see Chapter 5

probability distribution and fly across it, updating each area as it passed and giving an increase in precision. There has been some research into SAR UAVs which can update POA based on their findings.[23; 24; 25]

2.2.4 Computer systems

There are several computer programs in existence to aid in search and rescue. On land, these are mainly for managing calculations of probability (POA, POD and POS), and resource allocation[26; 27; 28] — taking the available searchers and splitting them across the various search areas which have been input by hand; assigning a number of people to search each area using methods such as the Washburn allocation[29] or the Charnes-Cooper algorithm.[26; 4] Some exist, however, which aid in the creation of probability maps. This has been common for marine SAR for some time[16; 30] but (as discussed in Section 2.2.2.2) on land it is far harder to calculate probabilities mathematically, which has meant that the software has lagged behind somewhat.[6] The problem is indeed tricky — as O'Connor and Frost[20] said, 'implementing any sort of reasonably realistic land motion model in a computer simulation would be extremely difficult since things like terrain, game trails, vegetation, and other features have considerable effects on how lost persons move but are often difficult or impossible to represent with the available information'. Historical statistics can, however, help to overcome some of these problems.

Programs which create probability maps take inputs about the missing person, terrain information, and other mapping details.[31] They then attempt to give a first approximation of a probability map which can be modified by experts, based on possible movement and historical statistics from the ISRID database (see Section 2.2.2.2).

Jones and Twardy[32] have developed a system for comparing these, and scoring their effectiveness at producing probability maps against actual data. Their MapScore server allows researchers to compare their lost person behaviour models against a great many ISRID cases to ensure that they work in many real life situations.

Early on in this PhD some minor work on automated probability map creation was undertaken, creating a program which took a terrain map and last known point

and attempted to create a containment map. The map was treated as a grid of cells, with a moving boundary expanding from the last known point where each cell next to the boundary was given a higher value than its lowest neighbour in a manner similar to Dijkstra's algorithm.

In Dijkstra's algorithm,[33] a route is found from point A to point B by moving radially outwards from point A until point B is reached, giving each new neighbour at the boundary at each step a value that is one higher than the smallest of its neighbours (see Figure 2.1). The shortest route is then obtained by working backwards from B to A, moving down the steepest gradient. This works well with obstacles.

In this probability map creation work the increase in value as the boundary increased (1, in Dijkstra's algorithm) was set by the terrain gradient — a flat path would provide a faster route than a steeply sloping one (see Figure 2.2). By setting a speed and time limit a maximum distance travelled could be obtained, giving an area in which the missing person was likely to be with flatter routes proving more likely than steep ones. This was to be extended to include more terrain data (positions of rivers, buildings, undergrowth, etc.) and to allow for increased probability for routes which travelled in straight lines than ones which doubled back, as well as taking in statistics for person type. However, as the creation of probability maps is not in itself directly relevant to the topic of UAVs for search and rescue, this was never fully completed.

2.3 Searching probability maps

However the probability map has been created, it is important to search it in a way that maximises the probability of finding the missing person in the minimum time. This requires both completeness and efficiency — completeness, in that every part of the map needs to have been (eventually) searched, and efficiency, in that the searchers find the missing person in the minimum possible time.[17] A search algorithm should aim for the most probable areas to be covered first.

Given a probability map (a map of values of POS) and a set of limited resources, it can be a difficult job for the person in charge of the search to assign those resources across the probability map.

15	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
14	13	14	15	16	17	18	19		23	24	25	24	25	26	27
13	12	13	14	15	16	17	18		24	25	24	23	24	25	26
12	11	12	13	14	15	16	17		25	24	23	22	23	24	25
11	10	11	12	13	14	15			24	23	22	21	22	23	24
10	9	10	11	12	13	14		24	23	22	21	20	21	22	23
9	8	9	10	11	12	13		23	22	**B**	20	19	20	21	22
8	7	8	9	10	11	12		22	21	**20**	**19**	18	19	20	21
7	6	7	8	9	10	11		21	20	19	**18**	**17**	18	19	20
6	5	6	7	8	9	10			19	18	17	**16**	17	18	19
5	4	5	6	7	8	9	10					**15**	16	17	18
4	3	4	5	6	7	8	9	10	11	12	**13**	**14**	15	16	17
3	2	3	4	5	6	7	8	9	10	**11**	**12**	13	14	15	16
2	1	2	3	4	5	6	7	8	**9**	**10**	11	12	13	14	15
1	**A**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	9	10	11	12	13	14
2	1	2	3	4	5	4	7	8	9	10	11	12	13	14	15

Figure 2.1: Dijkstra's algoritm. To find a route from A to B, work radially outwards from A giving each new neighbour at the boundary a value one higher than the value of the cells at the boundary. When B is reached, work backwards by taking the lowest numbered neighbour back to A.

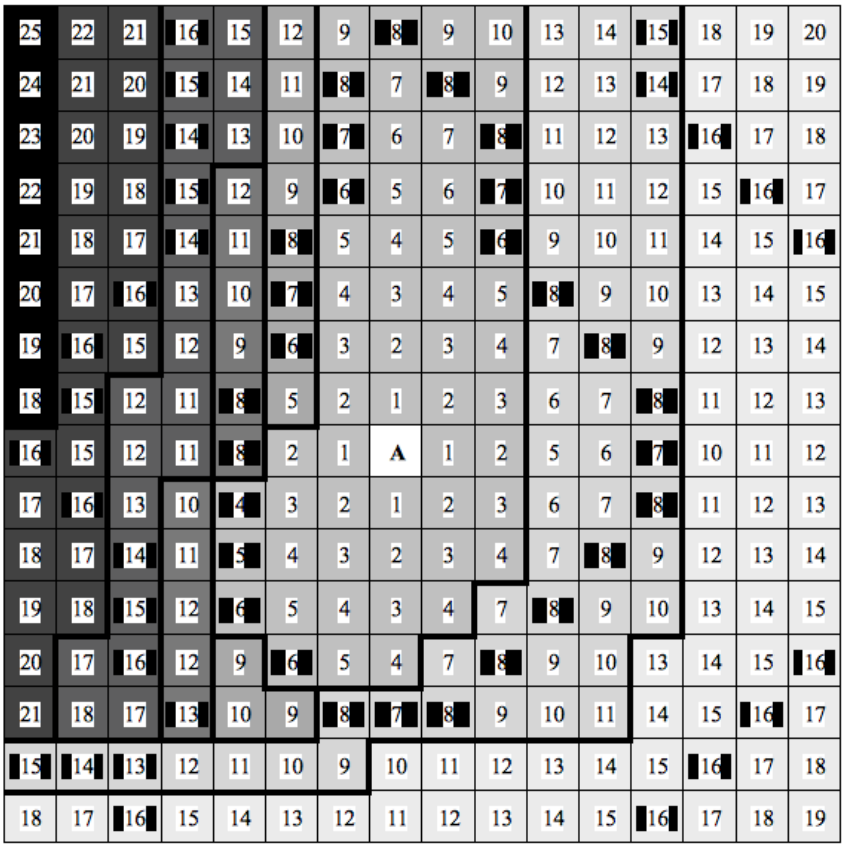


Figure 2.2: Probability map creation based on terrain gradient. Working radially outwards from A (the last known point), each new neighbour at the boundary is given an increased value based on the gradient. No gradient change gives an increase of one; crossing any gradient contour gives an increase of three. An estimate of the distance travelled can then be found for a given time. Two example times — 8 and 16 — are marked.

2.3.1 Current methods for human searchers

As mentioned in Section 2.2.3, current land search methods will typically involve splitting the map into large areas each of which is assigned a team of searchers. These searchers will travel to the area — a negligible amount of time compared to the many hours they may spend searching it — and will then search the area for one search period. At the end of that search period they will return to base and report any findings, along with an approximation of their POD. This will be used to calculate the probability map for the next search period, and the process will be repeated until either the person is found, or the search is called off.

When searching their assigned area a team of searchers may do a variety of things, depending on circumstances.[22; 7; 5] These include checking along trails and pathways, searching likely looking areas (e.g. places offering shelter, or places which historical statistics say are likely), calling for the missing person, or walking across an area in a line — keeping a roughly equal separation. Different search patterns exist for covering an area,[5] especially for searches by aircraft[13] (where, for example, a spiral pattern may prove more effective to cover the area than a boustrophedonic sweep search[17]). The exact search method used for an area is often largely up to the team assigned to it, however. According to the staff at Kinloss, for aerial searches the routes taken by SAR helicopters in the UK are entirely at the pilot's discretion — based more on experience than on any search strategy.

2.3.2 Distributed systems

The common setup for search and rescue — repeated cycles of teams being assigned areas, sent out to search them, then returning to report — give some problems when transferred to large numbers of autonomous units. It is designed for the situation where there is one control room into which comes all information found, and out of which come all orders for which resources should investigate which areas. The system may not have all information, but it may be assumed to have all known information. This leads to the problem that all units must either retain contact with the base at all times for reporting back information and receiving instructions (limiting their range, especially in hazardous conditions where communication over

a distance may be difficult), or else periodically return to the base — taking up time that could have been spent in searching. It also requires that there be a base; a specialised unit that contains all information, which creates a single point of failure.

The alternative — a distributed system[34; 35; 36] — while offering some improvements over a centralised control structure is not without problems. Resources may move in or out of range of one another at any point, which makes keeping the probability map entirely up to date across all resources likely to be impossible due to the limited communication. It also leads to a far more complex system; each unit will need to decide upon a course of action individually, as opposed to a central station which can aim to optimise routes for all resources.

This is essentially the difference between coordination and cooperation[37] — whether the various searchers are considered individual entities which work together for a common goal, or as extensions of a central controlling intelligence.

2.3.3 Greedy lookahead algorithm

For an individual searcher crossing a probability map it is important to maximise the POS of the route that it travels. One simple way that this is done is to use a 'greedy' algorithm. This takes the set of possible movements for the next step, and then the set of possible continued movements for each of these, continuing for a number of steps. After a number of steps, the highest valued route is selected and the searcher travels along this path. While travelling, the path can be being continuously updated (a 'receding horizon') — recalculating the next few steps for each step taken, to check that there is not a new route which would be better now that the algorithm is looking another step ahead. For multiple searchers this can be extended to take into account the cumulative POS from all of the searchers.

This method is used by many SAR UAVs,[38; 37; 24; 36] but it has a number of flaws. While the algorithm will produce better results the more steps ahead it searches, searching for increasing numbers of steps will give an exponentially increasing number of calculations. On a low powered system such as is common on a UAV, this would prevent more than a small number of steps ahead being calculated. Indeed, in several cases a single step lookahead is used — which is the

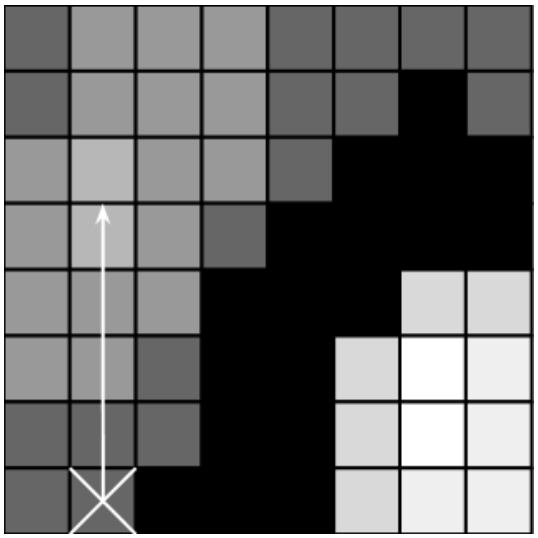


Figure 2.3: A greedy lookahead algorithm being caught on local maxima. With a 2 step lookahead a searcher starting at X will never be able to cross to the high probability areas in the lower right corner, as the boundary with an extremely low probability is at least 2 cells wide at all points. It will instead search a comparatively low probability area. This example could represent, in a real life situation, a wide boundary such as a river — a boundary which the missing person could cross, but which is near certain to not contain the missing person now.

same as using a simple hill climbing algorithm.

A larger flaw is that depending on the probability map being searched, it is entirely possible for a greedy algorithm (which only considers points relatively close to the UAV and doesn't take a global view of the map) to get caught on local maxima which are themselves considerably worse than the global maxima. In extreme cases the algorithm could plot a route which travels through the areas with the lowest probabilities, never finding the high probability areas (see Fig 2.3).

This is clearly, then, a suboptimal method for searching across a probability map. It is of great importance to use a search algorithm which optimises the time taken to search the highest probability areas on the map, taking into account travelling time. Such a method is described in the next chapter.

Chapter 3

Search and Rescue as a Weighted Cleaning Problem

3.1 Background

Robots, depending on their use, can be either stationary or mobile. Mobile robots in general — and autonomous ones in particular — come in a variety of types, for a large number of tasks, but their motion can generally be categorised into one of three types: General motion based on a set of rules; travelling to a specific point; or covering an area to visit a number of points. One specific example of this third is the Complete Coverage Path Problem (CCPP) — aiming to visit every point in an area, with a route based on certain criteria (such as to optimise duration or path length).[39] This is a requirement for several applications — from painting, to vacuuming, to searching — and several methods have been discussed[40; 41; 39] to solve it. An extension of this is to use multiple robots to cover the area; splitting the task into sections so that it can be completed in the minimum time possible.

A common method for completing the CCPP is to use boustrophedonic motion. In situations with no obstacles, this entails covering the entire area in a series of parallel sweeps (see Fig 3.12). This will easily guarantee total coverage. This method becomes harder to use when obstacles are included, and much work has been produced on breaking down complex areas into sections which can each be solved using

boustrophedonic motion.[42; 39] Other methods include algorithms based on the motion of ants, using virtual or actual pheromone trails,[43] or using methods from graph theory. Zelinsky et al.[44] suggest a method based on Dijkstra's algorithm, where instead of travelling from the start point to the goal the robot attempts to climb the gradient away from the goal while being restricted to predominantly travelling on cells that have not yet been visited. The robot therefore covers the furthest sections of the area inwards until the entire area is completed.

A not unconnected problem to the CCPP is the Cleaning Problem. In this, areas are marked as either being 'Clean' or 'Dirty'. The task of the robot is to convert 'Dirty' cells to 'Clean' ones.[43] In some variants — the Dynamic Cleaning Problem[45] — the Dirty tiles are able to contaminate the Clean tiles around them, spreading at a set rate. This allows for many situations to be modelled by this problem — from spreading fires (where Dirty represents being on fire) to chasing a fugitive (where Dirty represents cells where there is a possibility of the person being). Again, there have been several attempts at solving this,[45; 40; 46] and this, too, can be extended into the Collaborative Cleaning Problem, where multiple robots are available.

The problem being described in this chapter is the Weighted Cleaning Problem — a subset of the Cleaning Problem, where all tiles are given a value based on how important it is for the robot to visit them. The robot is then required — as in the Cleaning Problem — to visit every cell, while optimising its route to cover the cells of a higher value in the minimum time and, where possible, before the cells of a lower value (noting that the important factor is that the more high valued cells are covered as soon as is possible — not that the low valued cells are left until last). This allows for a more realistic way of describing many situations, for example in the case of a vacuum cleaner — the optimum solution for a vacuum cleaner is to clean the dirtiest areas first, such that if interrupted it will have cleaned the most important areas and have only left the less dirty sections uncovered.

The situation for which this problem is being used is that of aerial searching for missing persons; deciding a route plan for UAVs based on a probability map (a dynamically changing map showing the probability of the missing person being

in any particular area. Figure 3.1(a) shows an example probability map, where the lighter the point the more probable it is for that point to contain the missing person). Clearly in this situation time will be of the essence, and so searching the more likely places first is of great importance. For the purposes of this chapter this problem will be simplified to a static map[1] and will only use a single robot to cover it. In a final solution, the map would be dynamic and would be input from a separate system designed to calculate the probability of any particular area containing the missing person based on known data.

Several methods have been suggested previously to solve this problem or similar problems. Jin et al.[47] for example pose a similar problem, which is solved by splitting the problem into 'tasks' — which can be seen as points with values of importance that the UAVs must visit. A central processor uses a bipartite matching algorithm to assign UAVs to tasks. The assignment is based on the distance to the closest UAV, and multiple UAVs may be associated with one task, competing for the assignment. These UAVs will travel towards the closest task they are associated with, continuing until one is assigned to each task (by moving within a threshold range). Unassigned UAVs without associations travel towards unsearched areas (areas where tasks may or may not exist). This could be extended to solve the Weighted Cleaning Problem; the downside to doing so would be the great increase in computation required if all cells had an associated task. This method also requires a central processor to decide routes and assignments, which gives a single point of failure and requires complete contact from all UAVs at all times.

Polycarpou et al.[48] show a situation where multiple UAVs are searching an area, and their future routes are decided step by step using a weighted cost function which is maximised between the units. The elements of the cost function are many; the three which are most relevant to this work are: (1) moving to the point which has the lowest certainty (i.e. has been searched least), (2) moving in the direction of the point of highest uncertainty within the entire map, and (3) moving in the direction which would overlap least with the other units. This method could easily be modified

[1] I.e. the probability of a cell containing the missing person is only based on the initial values and the movement of the robot; there are no changes based on external influences. In a real life situation this would not always be the case.

to solve the Weighted Cleaning Problem by replacing the map of uncertainty levels with a probability map. The method as outlined does, however, have potential flaws — for example that having all units aiming towards the most important point on the map may lead to undue clustering as they all aim to compete. One element of the cost function that works towards (3) takes into account for each UAV all other UAVs' direction of travel; however, this assumes that the robots are most likely to continue moving in a straight line. While this is often a reasonable assumption (especially when fixed-wing aircraft are being used), in many cases it may hinder potential routes which may prove more optimal being considered.

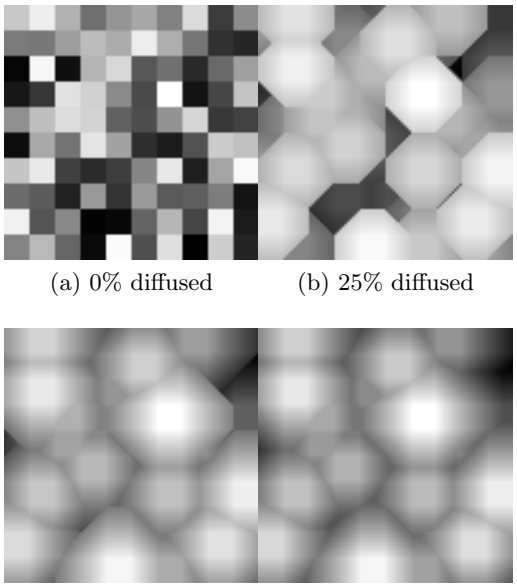
Flint et al.[49] take a dynamic programming approach for a very similar problem, in which a number of UAVs are to search an area. The area is split into cells, and each cell has a range of values relating to the probability of it containing one of the targets to be searched for. The method then used to cover this is to use a recursive function based (among other things, including potential threats) on the probability of the immediate neighbours at each stage. This reports positive results and could be used for the Weighted Cleaning Problem, but the use of a complex recursive function will tend to increase the processing power required of the robots. While processing can in some situations be run on a remote, centralised system, in many real life cases this would not be possible — and the processing power of small robots is typically very limited. In an earlier paper,[50] the authors suggest reducing the required processing by only permitting a limited number of recursions per step, to remove the need for 'unlimited computational power'.

This is similar to the Greedy Lookahead method (see Section 2.3.3), where robots look at the probabilities immediately around them and work out the best route only taking into account the next few steps.

3.2 Method

The method under consideration for solving this problem is that of Diffusion.

Starting with a Base Map, showing the raw probabilities (i.e. the value of every point on the Base Map directly represents the actual probability of that point containing the missing person), a second, overlay map is created. This overlay is known



(a) 0% diffused (b) 25% diffused

(c) 50% diffused (d) 100% diffused

Figure 3.1: One map from Set A, at four stages in diffusion using the Highest Neighbour method

as the Navigation Map, and gives values which are used by the robot for deciding its route. The values on the Navigation Map do not represent the actual state of probabilities in real life, but instead provide a means for the robot to plan future movement. The robot will pass over the Navigation Map using a simple hill climbing algorithm, and the values on the Navigation Map over which it is passing — the Navigation Levels or Values — are being recalculated continuously. The value of any point on the Navigation Map for a particular iteration is based on the corresponding Base Map value at the same point and on the neighbouring Navigation Map values from the previous iteration, using a diffusion algorithm.

An algorithm must be selected which will allow the values of the Base Map to diffuse across the Navigation Map, creating gradients which will lead up to the highest points for the robots' hill climbing algorithms to follow. This can be seen in Fig 3.1, where one probability map is shown at various states of diffusion, from one of the simulated algorithms.

A selection of potential diffusion algorithms are given in Section 3.2.1.

3.2.1 Diffusion methods

Diffusion is defined by the Oxford English Dictionary as 'dispersion through a space or over a surface' and 'dissemination (of ... knowledge)'.[51] There are many models of diffusion, and many equations which would fulfil this description.

3.2.1.1 The Highest Neighbour method

The first method under consideration for disseminating the existence of the high valued cells across the map would be for the Navigation Level of a cell to be set to either its Base Value or to a percentage of the highest of its neighbours' Navigation Values, whichever is larger. The range of the diffusion produced by this method — the distance over which a base value will be propagated — is dependant on the percentage diffusion value (D in Equation 3.1), allowing the system to be tailored either at one extreme towards searching for the base value regardless of its distance from the robot's current position (a high diffusion), or at the other towards searching for the highest of the cells in the immediate vicinity to the robot.

$$l_t(i,j) = \max \begin{cases} l_{t-1}(i^N, j^N)D \\ L_{t-1}(i,j) \end{cases} \tag{3.1}$$

In this diffusion equation, $l_t(i,j)$ is the Navigation Value of the cell (i,j) at iteration t, $L_t(i,j)$ is the Base Value of the cell (i,j) at iteration t, and D is the diffusion level, where $0 \leq D \leq 1$. (i^N, j^N) represents all cells neighbouring (i,j); i.e. $(i-1, j-1)$, $(i, j-1)$, $(i+1, j-1)$, $(i-1, j)$, $(i+1, j)$, $(i-1, j+1)$, $(i, j+1)$, and $(i+1, j+1)$. The changing value of D allows for the behaviour of the robots to be weighted towards at one extreme seeking high probability regardless of distance, and at the other seeking close cells regardless of probability. The gradient at any point will show the direction of travel to the 'closest', 'high probability' cell (where the definitions of 'closest' and 'high probability' are dependant on the diffusion level D).

For comparison, the fully diffused Navigation Map for the first map in Set A (see Section 3.3) using Equation 3.1 is shown in Figure 3.2.

This method has a complexity (for a full diffusion over the whole map until a

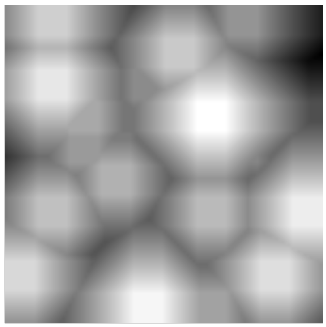


Figure 3.2: Diffusion using the Highest Neighbour method (Equation 3.1)

stable state is reached) of $O(n^{1.5})$.

3.2.1.2 The Average method

$$l_t(i,j) = \max \begin{cases} \frac{\sum l_{t-1}(i^N, j^N)}{8} \\ L_{t-1}(i,j) \end{cases} \tag{3.2}$$

One advantage of the previous method is that it will reach a stable state in a short period, as all changes are one way — a higher valued cell can affect a lower valued cell, but not vice versa, and any base map changes would propagate radially outwards until all cells are updated. A diffusion based on average values on the other hand, as shown in Equation 3.2 (setting each cell to being the arithmetic mean of its neighbours), would cause cell updates to have a reciprocal effect. A change in a single cell's value would cause all neighbours of that cell to update, and each of those updates would then in turn cause the original cell to change again. This can be overcome through use of a threshold, or a limit on the number of iterations, but the total number of iterations is still likely to be high due to the number which are self induced.

For the purposes of creating diffusion example images a difference threshold value internal to the program of 0.001 was used, which maps to a value of $3.92 \times 10^{-4}\%$.[2]

The threshold would also have a further effect, in a system based purely on neighbour averages — that of preventing the entire Navigation Map from averaging to a single value. This is clearly a situation to be avoided; the purpose of the

[2]I.e. A cell will not be updated by a neighbour if the neighbour has a value within $3.92 \times 10^{-4}\%$ of the cell

Navigation Map is to assist the robot in navigation, and hill climbing algorithms will not work if there is no gradient to climb. Equation 3.2 is not, however, purely based on neighbour averages — as the Navigation Layer is creating gradients up to high points from the Base Layer it is logical to limit the minimum Navigation Value for each cell to its Base Value, preventing the Navigation Map from averaging to a single uniform level.

A further downside to the Average method, noted experimentally, is that there is no internal mechanism for deflation. A cell's Navigation Value will become the average of its neighbours', and a low valued cell surrounded by high valued cells will therefore retain its high valued status. This causes problems for the robot moving across the map — the Base Value of the cell the robot has just passed may now be considerably lower than it had been, but this will not be being reflected in the Navigation Map, as the comparatively high Base Values of the cells on either side of it will be pulling up its Navigation Value, averaging and smoothing out the sharp change in the landscape of the Navigation Map and making it appear to the robot to be as important to cover as the cells with a truly high Base Value. The only way around this is to artificially reduce the Navigation Values of the cells by introducing some form of deflation. For the purposes of the simulation it was decided to implement a small percentage reduction of the average values obtained from cell neighbours, changing the equation to that of Equation 3.3, where R is the value by which it has been reduced, and $0 \leq R \leq 1$.

$$l_t(i,j) = \max \begin{cases} \frac{(1-R)\sum l_{t-1}(i^N,j^N)}{8} \\ L_{t-1}(i,j) \end{cases} \tag{3.3}$$

The fully diffused Navigation Map using Equation 3.3 is shown in Figure 3.3.

This method has a complexity (for a full diffusion over the whole map until a stable state is reached) of $O(n^{1.5})$.

3.2.1.3 The Median method

A similar method to the Average method, with similar downfalls, would be to take the median of the neighbour cells — ordering the eight neighbours and taking the

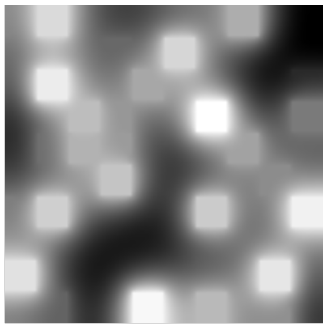


Figure 3.3: Diffusion using the Average method (Equation 3.3)

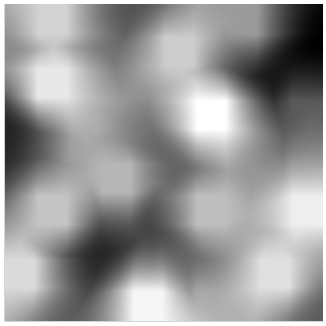


Figure 3.4: Diffusion using the Median method

mean of the two midmost values. This would similarly not reach a stable state without use of a threshold. The Navigation Map produced is that shown in Figure 3.4.

This method has a complexity (for a full diffusion over the whole map until a stable state is reached) of $O(n^{1.5})$.

3.2.1.4 The Distance method

Another potential method is to set a cell's Navigation Value using Base Values from all other cells and the distance between them, rather than relying on dissemination over multiple iterations via neighbour cell values.[3] This would, however, require many more calculations.

$$l_t(i,j) = \max \left\{ \frac{L_{t-1}(i',j')}{k} \right. \tag{3.4}$$

[3]This has some similarities to the Potential Search described by Waharte and Trigoni.[52]

Here, (i', j') is a cell whose position is not based on the position (i, j), and k is the distance between the two cells (i, j) and (i', j'), such that $k = \sqrt{(i - i')^2 + (j - j')^2}$; this would require — for a map of size $w \times h$, where $w \geq h$ — w^2h^2 comparisons, as each cell would need to be compared wh times.

Comparing this to the number of calculations required in the same situation from the recursive, neighbour based Equation 3.1, each cell is compared against its eight neighbours and itself, requiring 9 comparisons. This is repeated as the information disperses across the board. Taking a non-optimised case of updating all cells in every iteration until a stable state is reached, each iteration requires $9wh$ comparisons. The map will stabilise once — in the worst case — a highly valued cell in one corner has diffused to the opposite corner. Travelling using diagonals and perpendiculars, it will reach there in at most w iterations (where $w \geq h$). This means that the total number of comparisons will not exceed $9w^2h$ — which even in its worse case will be faster unless $h < 9$. The same would be true for any diffusion method which could not be calculated recursively.

The Navigation Map produced by Equation 3.4 is shown in Fig 3.5.

The speed of calculation can be improved by limiting the size of the area over which the calculation has taken place. If (i', j') is limited such that $k \leq R$ (where R is a constant 'radius' value) then the number of calculations is reduced to $\pi R^2 wh$ comparisons. While this would greatly reduce the amount of time required to calculate the navigation values, it would be at the expense of the amount of data available to the robot at any given point — limiting the range over which it could detect high base values, and thus preventing it from being able to seek out the highest base values on the map.

It should be noted here that the Navigation Map produced by Equation 3.4 is not entirely appropriate for a hill climbing algorithm. As has been mentioned, for each of the methods mentioned here there will be a pay off between at the one extreme guiding the robot to a high valued cell regardless of distance and at the other to a close cell regardless of value. The results of Equation 3.4 give a navigation map which is too close to the second of these extremes. Short distances will skew the navigation values calculated, and in the extreme (when comparing against a cell's

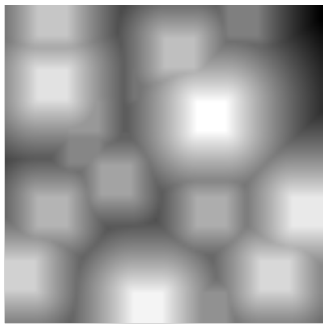


Figure 3.5: Diffusion using the Distance method (Equation 3.5)

own base value) the navigation value calculated will reach infinity. This can be prevented by adding in an offset — a constant value which is added to the distance, such that the greater the offset value the less effect the distance will have on the equation. Equation 3.4 therefore changes to Equation 3.5, where O is the offset; low values of O will give greater import to close cells than high values of O.

$$l_t(i,j) = \max \left\{ \frac{L_{t-1}(i',j')}{k+O} \right. \tag{3.5}$$

For the experiments described later a further variable is used; R, where R is the maximum radius over which Equation 3.5 is run, such that $k \leq R$.

This method has a complexity (for a full diffusion over the whole map until a stable state is reached) of $O(n^2)$.

3.2.1.5 The Low Pass Filter method

A desirable Navigation Map could be looked at as being the result of a low pass filter across the Base Map. The high frequency information — the exact delineation of base values — is unimportant to the hill climbing robot; instead it needs the low frequency information, giving a generalised landscape. One potential method for producing a Navigation Map would therefore be to take a Fourier transform of the Base Map, remove a percentage of the high frequency, outer layers, and transform back. This would produce the entire navigation map in one go — outputting a navigation map such as Figure 3.6 — and as such would be potentially faster for calculating initial values than other methods. There are, however, far fewer options for

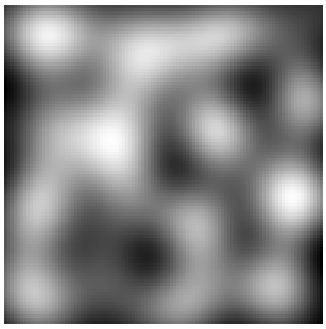


Figure 3.6: Diffusion using the Low Pass Filter method

optimisation inherent in using a Fourier based filter, which already has a complexity of $O(n \log n)$ (higher than the complexity for most other methods, of $O(n^{1.5})$). If using one of the previously discussed diffusion methods there are opportunities to optimise the algorithm (allowing the navigation map to return to a stable state after very few calculations, as the change per robot movement in the base map is small). A Fourier transform, on the other hand, would need to be run over the entire Base Map after every change in base data; and every transform would take the same amount of time.

Another difficulty with using a Fourier transform in this way is that the Navigation Map produced will include fringe effects. This means that there will be many false positives — hills which do not lead to high points on the Base Map. The robot will have no means of knowing that these are false; they will not disappear once visited (as they are not caused by the cells at their peaks), and so robots will become stuck at the summit with nowhere else to go.

In order to mitigate this, the output from the Fourier transform is multiplied by the Base Map in order to artificially raise the value of actual positives. This is discussed further in Section 3.4.5.1.

It was later decided, after the experiments had been completed, that the false peaks may have been caused by ringing from the harsh frequency filtering — cutting off all frequencies above, and retaining all frequencies below, a certain threshold. A better solution to this would have been to use a Gausian filter, to prevent the ringing effects.

This method has a complexity (for a full diffusion over the whole map until a

stable state is reached) of $O(n \log n)$.

3.2.1.6 The Conservative method

Most of the diffusion methods covered so far have been focussing on the dissemination of the base information across the navigation map. Many real life situations in which diffusion occurs — such as chemical diffusion or heat transfer — are instead based on conservative diffusion. For this method the Navigation Values sum to a constant number, which will remain constant throughout diffusion. Cells with a high initial value will reduce as the cells surrounding them increase, and this leads to a potential problem. If left unchecked this would, over time, raise up the low navigation values and lower the high ones — eventually resulting in a Navigation Map with no discernible features. While this problem has been encountered in previously discussed methods (such as the Average Method) they could be solved by forcing a limit on the minimum Navigation Value of a cell to be that of its Base Value. The conservative nature of this method, however, precludes this solution — as one cell's Navigation Value goes up, another must go down, and with a starting state of the Base Values this would produce an impasse.

Instead, this problem is solved by imposing a minimum gradient across the Navigation Map, below which no change will occur. Lesser gradients would of course be permitted if they occurred naturally, but the algorithm would not be able to create them itself. The equation used for this method is therefore:

$$\left.\begin{aligned} l_t(i,j) &= l_{t-1}(i,j) - G/8 \\ l_t(i^N,j^N) &= l_{t-1}(i^N,j^N) + G/8 \end{aligned}\right\| \text{if } l_{t-1}(i,j) - l_{t-1}(i^N,j^N) > G \tag{3.6}$$

where G is the minimum gradient, and (i^N, j^N) is each of the cells neighbouring (i, j). The Navigation Map created is shown in Figure 3.7.

This method has a complexity (for a full diffusion over the whole map until a stable state is reached) of $O(n^{1.5})$.

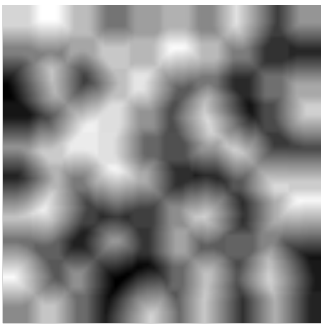


Figure 3.7: Diffusion using the Conservative method (Equation 3.6)

3.2.1.7 The Neighbour Difference method

While Repenning[53] does not cover the Complete Coverage Path Planning problem or the Weighted Cleaning problem directly, the diffusion algorithm he describes is certainly applicable to the situation.

The paper describes the concept of diffusing signals across an area to allow systems to climb the gradient, using it for many situations and focussing on computer games — Pacman, Soccer, etc. — where units are required to travel towards a set target or set of targets. This is also extended to allow for different diffusion layers from different objects, and different diffusion values when passing through different sprites, allowing for goal obfustication — the situation where the diffusion from one target is masked from most units by the closest unit being in the way, preventing them from all following the same route. Goal obfuscation is a useful technique in many situations but is somewhat less usable in this context, with a small number of units in a large open area with few obstacles.

Repenning's calculations are made from the point of view of the cells (or 'antiobjects' — essentially cellular automata representing every space not covered by an object). The paper states that 'Collaborative Diffusion is not directly relevant to robotics because in robotics there is no other way than putting all the computational intelligence inside the robots'; I would dispute this claim, however, as the use of virtual cells would permit the robots to use this method to calculate a gradient — indeed, if this was the case then none of the methods outlined in this thesis would be usable for real robots.

The equation used by Repenning, as given, is

$$u_{0,t+1} = u_{0,t} + D \sum_{(i=1)}^{n} (u_{i,t} - u_{0,t}) \tag{3.7}$$

where n is the number of neighbours, $u_{0,t}$ is the 'diffusion value' of each cell being updated at time t, $u_{i,t}$ is the 'diffusion value' of the ith neighbour (where $i > 0$), and D is the 'diffusion coefficient' (a term of 'speed of diffusion', where $0 \leq D \leq 0.5$). Converting this into the terminology used above gives

$$l_t(i,j) = l_{t-1}(i,j) + D \sum \left(l_{t-1}(i^N, j^N) - l_{t-1}(i,j) \right) \tag{3.8}$$

It should be noted that this chapter has been taking a basic case of eight neighbours per cell, including diagonals as well as adjacents, as specified in Section 3.2.2. Repenning assumes a basic case where each cell has four neighbours, although it is similarly extensible to any arbitrary number.

Repenning's diffusion algorithm works by summing the differences between each cell's Navigation Value and its neighbours', multiplied by a 'diffusion coefficient' which is described thus:

> 'The diffusion coefficient controls the speed of diffusion. A larger value will result in quick diffusion. In material science each material has a specific heat diffusion coefficient. Silver, for instance, has a much larger heat diffusion coefficient than a thermal insulator such as wood. Consequently Silver will spread heat much more rapidly than wood.'

The use of a 'diffusion coefficient' to set diffusion speed is interesting. In the situation put forward in this chapter it is to be assumed that information should be disseminated as quickly as possible — there is no advantage in information not being received by the robots immediately, and no reason for delaying the total propagation of information such that a stable state is reached before updating the hill climbing algorithm.[4] The main use of D in the paper is to simplify equation 3.7 — if $D = 1/n$ then the equation reduces to

[4] With the possible exception of reducing processing power

$$u_{0,t+1} = \frac{1}{n} \sum_{(i=1)}^{n} u_{i,t} \tag{3.9}$$

which no longer takes into account the previous Navigation Value of the cell itself; instead, all Navigation Values are based on neighbour Navigation Values from the previous iteration.

Converting to the terms used elsewhere in this chapter, Equation 3.9 becomes

$$l_t(i,j) = \frac{1}{8} \sum l_{t-1}(i^N, j^N) \tag{3.10}$$

which bears a marked similarity to the Average method of Equation 3.2, and therefore shares all of its pitfalls. This simplification is not used in this chapter; instead a range of values for D are considered. The navigation map produced is shown in Figure 3.8.

This algorithm has a number of potential downfalls. The first is shared with others such as the Average method, in that its reciprocal nature means it will not reach a stable state without use of a threshold. The equation also doesn't take Base Values into account; there is no separation between Base Values and Navigation Values, and all changes are based solely on the previous iteration's Navigation Map.

Another difficulty is the lack of opportunity to reduce the Navigation Levels. Once a high value has propagated it has no means of reducing down to a lower value, other than through the influence of lower valued neighbours. This means that there is no means to reduce the overall value of the system.

Further description of this work, including a discussion on instability caused by the use of a diffusion coefficient, can be seen in Section 3.2.4.1.

This method has a complexity (for a full diffusion over the whole map until a stable state is reached) of $O(n^{1.5})$.

3.2.2 Neighbours

Many of the proposed methods base the Navigation Value of a cell for the next iteration on the Navigation Values from the previous iteration of its neighbours. For this, the term (i^N, j^N) was defined to include eight neighbours of the cell (i, j). All

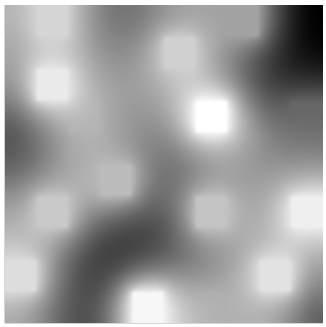


Figure 3.8: Diffusion using the Neighbour Difference method (Equation 3.8)

methods use this same definition of 'neighbour'. An investigation of the effect of differing neighbour definitions lies outside the scope of this work, but would make for interesting future investigation.

3.2.3 Computing power

One aspect of the proposed diffusion methods which must be taken into account is that of computing power. Collaborative swarms of robots will in general comprise of a large number of small, inexpensive and low powered systems, compared to standalone robots, which are typically larger and more powerful.

As the systems being used will be constrained by processing power, memory and communication it is of great importance that the real time processing required is kept to a minimum. This must be taken into account when developing a method for this kind of situation.[41]

The use of diffusion allows for the problem of navigation to be split into two sections. The first — short term route planning — will require a very low level of processing power, as it is merely a hill climbing algorithm. The second — the creation of the navigation levels to form a series of gradients over which the hill climbing can take place — can be performed in a more ad hoc fashion. A small delay in the hill climbing would cause the robot to immediately stop or continue in a straight line regardless of gradient, while a delay in the diffusion might cause the robot to remain at an old peak after having searched it.

It's also possible for the calculations required for diffusion to be distributed between robots, allowing them to pool computing resources. This depends on com-

Method	*Complexity*
Highest Neighbour	$O(n^{1.5})$
Average	$O(n^{1.5})$
Median	$O(n^{1.5})$
Conservative	$O(n^{1.5})$
Neighbour Difference	$O(n^{1.5})$
Low Pass Filter	$O(n \log n)$
Absolute Distance	$O(n^2)$

Table 3.1: Computational complexity of each method

munication, however, and assumes that all robots have the same diffusion map. As the algorithms are extended to cover multiple agents it is possible that individual diffusion maps would be required — allowing each robot to 'repel' other robots by affecting the diffusion through the map.[53]

The methods described above have a range of complexities, and Table 3.1 shows the methods ranked by complexity.

3.2.4 Previous work

3.2.4.1 Repenning[53]

The method outlined by Repenning (described in Section 3.2.1.7) was originally designed to produce navigation towards discrete targets. The primary case given is that of ghosts tracking a Pacman — there is a single, point source (Pacman) which multiple robots (ghosts) are travelling towards, with obstacles (walls) to avoid. Once any robot touches that source, the game is over.

Other examples given include a simulated person looking for food — in this case there is a single robot (the person) travelling towards one of multiple sources (each representing a piece of food). Some sources may be at higher levels than others, with the level determined by the attractiveness of the food, but all sources are liable to be separated in space and only a limited quantity of food is required.

The third example given is that of a football match. In this context there are several sources — including the ball and the two goals — which diffuse independently; that is, they each have individual Navigation Maps over which their presence is dif-

fused, such that the diffused value of one will have no effect on the diffused value of another. Once the ball has been brought into contact with any part of the goal source the situation is completed.

One of the major differences between Repenning's work and the situation considered here is that this situation can be seen as more a continuous map of values to be diffused, rather than the highly discrete points in the three examples above.

The use of a diffusion coefficient in the algorithm put forward causes several potential problems, including instability, if it does not meet the exact value of $D = 1/n$ used for simplification to Equation 3.9.

Using Equation 3.8, for values of D which are low ($D < 1/n$) the range of diffusion would reduce as $D \to 0$ (i.e. as $l_t(i,j) \to l_{t-1}(i,j)$).

For values of D which are high ($D > 1/n$) the response demonstrates a degree of instability. Repenning gives a limit for D of [0..0.5]. Neither cell values ($l_t(i,j)$) nor number of neighbours (n) are limited; although four neighbours are assumed in the paper.

If Equation 3.7 is rewritten as

$$u_{0,t+1} = (1 - nD)u_{0,t} + D \sum_{(i=1)}^{n} (u_{i,t}) \tag{3.11}$$

and it is assumed both that there are four neighbours ($n = 4$, as described by Repenning) and that the maximum value is taken for the diffusion coefficient ($D = 0.5$), it can be shown that

$$u_{0,t+1} = \frac{u_{1,t} + u_{2,t} + u_{3,t} + u_{4,t}}{2} - u_{0,t} \tag{3.12}$$

In other words, the algorithm subtracts the value for each cell from the previous iteration from twice the average of its neighbouring cells. This can lead to large fluctuations.

If an extreme example is taken of alternating cells at iteration t, half (A cells) with a value of 0, and half (B cells) with a value of 100 (such that each cell at 0 has four neighbours at 100 and vice versa). Stepping through the iterations:

$$\begin{array}{lll} t+0 & A & = \frac{4\times B_{t-1}}{2} - A_{t-1} = 0 \\ & B & = \frac{4\times A_{t-1}}{2} - B_{t-1} = 100 \\ t+1 & A & = \frac{4\times 100}{2} - 0 = 200 \\ & B & = \frac{4\times 0}{2} - 100 = -100 \\ t+2 & A & = \frac{4\times -100}{2} - 200 = -400 \\ & B & = \frac{4\times 200}{2} - (-100) = 500 \\ t+3 & A & = \frac{4\times 500}{2} - (-400) = 1400 \\ & B & = \frac{4\times -400}{2} - 500 = -1300 \end{array} \tag{3.13}$$

The values will continue to increase, fluctuating between positive and negative and becoming increasingly unstable. It is therefore probable that D should be constrained such that $0 < D < 1/n$.

3.2.4.2 Yang and Luo[40]

Yang and Luo describe a diffusion method for solving the Cleaning Problem, representing each cell on a map as a neuron in a neural network. The change in value of each neuron is given by a shunt equation. This shunt equation contains an excitatory term (based on both the cleanliness of the cell and that of its neighbours), an inhibitory term (based on the existence of an obstacle in the cell), and a passive decay (to allow the cell values to drop over time, thus preventing the dynamic landscape from saturating). While 'diffusion' is never explicitly mentioned in this paper, the neural network is diffusing the existence of dirty tiles over the entire map. The robots climb the diffusion gradient, with their direction of travel weighted in favour of a minimal change in angle, and continue until the map is clear.

This paper has many similarities to the work described here, with a few differences. Most notably, Yang and Luo do not cover the Weighted Cleaning Problem — cells have one of three possible states (Dirty, Clean, or Obstacle), with associated values E, 0 and -E respectively (where E is 'a very large positive constant'). Yang and Luo's use of neural networks in which — for the basic case — all neurons have only connections to their immediate neighbours could be realised as above through Cellular Automata (CA). The value of CAs are based on their current value and

that of their immediate neighbours, based on a simple algorithm.

The shunt equation (diffusion algorithm) is given as:

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)\left([I_i]^+ + \sum_{j=1}^{k} w_{ij}[x_j]^+\right) - (D + x_i)[I_i]^- \tag{3.14}$$

Values are given for most of the variables for a particular simulation,[5] giving:

$$\frac{dx_i}{dt} = -20x_i + (1 - x_i)\left([I_i]^+ + \sum_{j=1}^{8} w_{ij}[x_j]^+\right) - (1 + x_i)[I_i]^- \tag{3.15}$$

where x_i is the neuron value of cell i. I_i is specified as being E if cell i is dirty, $-E$ if cell i is an obstacle, and 0 otherwise. $[a]^+ = max(0, a)$ and $[a]^- = max(-a, 0)$. w_{ij} is μ divided by the distance between cells i and j, which (as $r_0 = 2$ so the connected neurons are the 8 immediate neighbours) will either equal 1 (for adjacent neighbours) or $\frac{1}{\sqrt{2}}$ (for diagonals).

This can be simplified by taking a basic hypothetical situation. On an infinite grid with no obstacles, $[I_i]^- = 0$ in all cases, and $[I_i]^+ = I_i$. All cells are clean ($I_i = 0$) save one (cell A, $I_A = 50$).[6]

$$\frac{dx_i}{dt} = -20x_i + (1 - x_i)\left(I_i + \sum_{j=1}^{8} w_{ij}[x_j]^+\right) \tag{3.16}$$

Logically, the neural network should propagate out the dirty cell. Looking at some of the values over successive iterations:

Iteration	Cell A	Adjacent	Diagonal
1	0	0	0
2	50	0	0
3	-3,400	35	24.7
4	734,597	3,551.1	658.7

[5] $A = 20$, $B = 1$, $D = 1$, $\mu = 0.7$, $r_0 = 2$ (therefore $k = 8$), $E = 50$

[6] Initial conditions aren't specified, but as the algorithm is described as being able to 'deal with arbitrarily changing environments' an initial value of $x_i = 0$ appears a reasonable assumption.

As can be seen, this appears to be leading to instability. Values are fluctuating between positive and negative, and increasing greatly between each iteration.

This equation is also used in follow up papers (which state that the system is stable)[54; 55], and the bounds of the neural activity are described as being limited to the range [-D, B] (in this case, [-1, 1]) — but as the exact values used for a working experiment are not given, and as I was unable to obtain a stable set of variable values for this algorithm through experimentation, it unfortunately cannot be compared against the other methods described in this chapter. Further work would be required to find a set of variable values for this method to allow it to be used.

3.3 Experiment

To compare the effectiveness of the range of diffusion algorithms they were each tested in a range of simulations.

In order to conduct these simulations it was first necessary to obtain a dataset over which to run them. Two sets of Base Maps were used. All maps are 100 units square (10,000 cells), and each set contains 10 unique maps.[7]

Map Set A was randomly generated. The individual maps were split into a 10x10 grid (each square being 10x10 units, or 100 cells), and each of these was set — at random — to a different value from the range 1–100. Fig 3.9 shows an example map from this set.

Map Set B was created using terrain maps from the NASA SRTM (Shuttle Radar Topography Mission) data set, with high ground levels being mapped to high base values, creating Base Maps which were more likely to replicate those found in a real search and rescue situation. It also allowed for an interesting comparison with the more regularly arranged maps of Set A. Fig 3.10 shows an example map from this set.

In both cases the values of all maps were scaled from 0 to 255 for internal program use, but are represented throughout as a value from 0–100 inclusive. Figure 3.11 shows the average number of cells at each value for the two map sets. For clarity,

[7] See Appendix C for images of all 20 maps

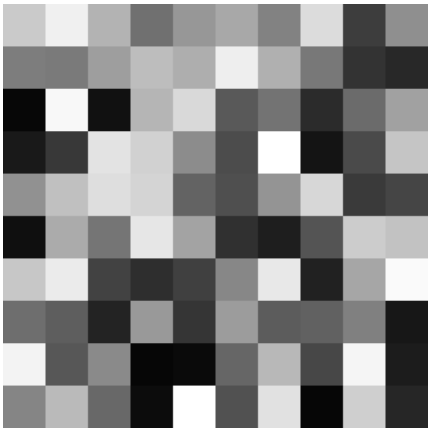


Figure 3.9: First probability map from Set A; randomly generated (100x100 units)

Figure 3.10: First probability map from Set B; based on SRTM data (100x100 units)

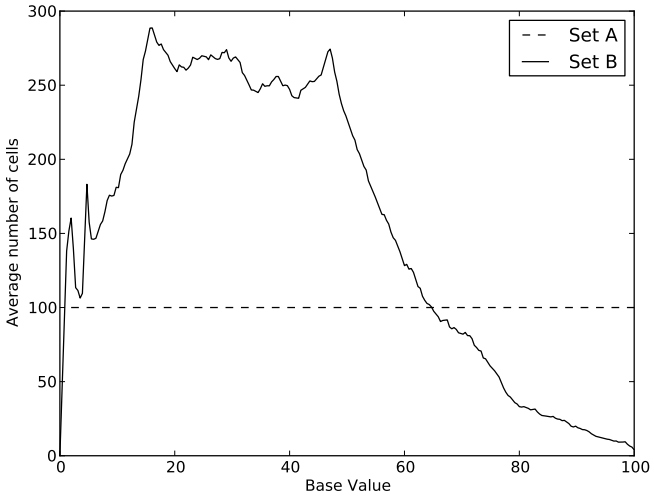


Figure 3.11: Averaged number of cells, from the total of 10,000 per map, which are at each Base Value for the two map sets

values at which there are no cells have not been showed. The number of cells at each value is constant for Set A; as specified, each map in that set contains 100 cells at each value.

The distribution in Set B is much more weighted, with more cells having lower values. It should be noted that while the area under the graphs (i.e. the total number of cells on the map) does not appear to be the same between the two map sets, this is an artefact from the averaging process as well as from the removal of points where no cells exist at a particular probability.

3.3.1 Cleaning

In the Weighted Cleaning Problem, two forms of cleaning are possible — Perfect Cleaning and Imperfect Cleaning.

With Perfect Cleaning, whenever a cell is cleaned (i.e. when a robot passes over that cell) the cell is entirely cleaned. Whatever the cell's current value, it will be set to zero.

The assumption of perfect cleaning is flawed in many cases where perfect cleaning is not possible — in other words, where a cell still requires coverage after having

been visited.[56]

With Imperfect Cleaning, whenever a cell is cleaned the cell's value is set to a percentage of its current value. Repeated cleaning is required to set the cell value entirely to zero.

In many contexts imperfect cleaning is a more realistic method — if a vacuum cleaner is unable to remove all dirt on the first try; or a painting robot requires multiple coats to cover an area; or a searching robot is able to miss objects when looking at a particular area — very few sensors are perfect,[3] and there is likely to be a chance that something has been missed.

3.3.2 Perfect cleaning

The first experiment involved a simulation where a route is calculated using Navigation Maps created from each of the diffusion methods outlined in Section 3.2.1, at a range of variable values. A robot then follows the routes, setting the Base Map value of each cell passed to zero, and continues until the entire Base Map is uniformly at zero (the entire map is clean). This is performed using all maps from each map set, and the results compared.

Each method is run at a variety of different variable levels, with the results from these compared against each other to find an optimum set of variables for the algorithm. Once the best option for each method has been discovered they are tested against the other methods to find which provides the best performance.

As a further comparison, each method outlined above is also compared against a boustrophedonic method — a 'brute force' solution involving travelling across the area in a parallel sweep pattern designed to cover all cells in a minimum time but not taking into account the values of the cells it covers (see Fig. 3.12).

3.3.3 Imperfect Cleaning

The first experiment was modified to include Imperfect Cleaning; that is, that the Base Value of the cells covered by the robot would not reduce entirely to zero, but instead would lower to a percentage of their previous value. This in turn necessitates a second change, to the end condition of the simulation as it would take a

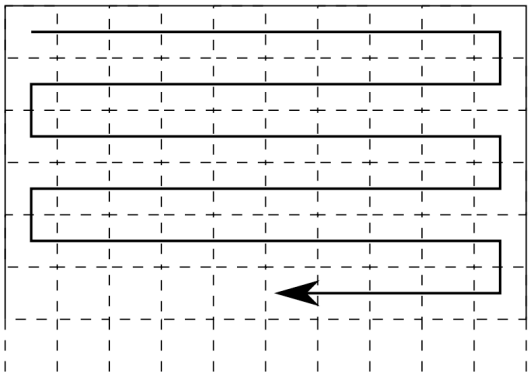


Figure 3.12: Boustrophedonic motion across a map. The robot travels along one entire row and then travels back along the row below, guaranteeing full coverage in the minimum time

considerable length of time for the Base Map values to entirely approximate zero. In all other aspects the Perfect and Imperfect Cleaning experiments are identical.

The end condition of this experiment is instead to 'find' a certain proportion of 'people' from the map. The positions of these people are calculated dynamically, and people may be found in any of the cells with a known and calculable probability. This probability is based both on the likelihood of there being a person in that space (taking terms from search theory, the Probability of Area, or POA) and the likelihood of the robot being able to find them if there is (the Probability of Detection, or POD). For any cell, therefore, a robot will have a possibility of finding a person; a robot searching more probable squares is more likely to find a greater number of people in a set time, however.[8] For this experiment the POD (Probability of Detection) has been taken as being a constant 0.2. In real situations this would be liable to change depending on location, but for the purposes of the simulation a constant POD is acceptable.

The different methods are compared at different variable levels to see how long each takes to find 500 people.

[8]It should be noted here that in a real search scenario the goal would be to plan a route optimising on Probability of Success (or POS, where POS = POA × POD) — the likelihood of finding a person in a particular cell, rather than the likelihood of the person being in that particular cell.[11] For this simulation a constant POD is used (i.e. the imperfection of the cleaning is constant irrespective of location), so the area over which the robot will be passing will be proportionately the same for POA as for POS.

The change in the Base Value of each cell after being visited is given by the equation:[15][9]

$$\mathrm{POA}_{s,n} = \mathrm{POA}_{s,n-1} \times (1 - \mathrm{POD}_{s,n}) \tag{3.17}$$

Given that nothing is found after a search of cell s at time n, $\mathrm{POA}_{s,n}$ (the Probability of Area — the probability of the person being in a particular cell — for cell s at time n) after searching is based on $\mathrm{POA}_{s,n-1}$ and $\mathrm{POD}_{s,n}$ (Probability of Detection — the probability of the robot finding the person given that they are there — of cell s at time n).

Converting this into the terms used elsewhere,

$$L_t(i,j) = L_{t-1}(i,j)(1 - \mathrm{POD}) \| \text{ when visited, if nothing found} \tag{3.18}$$

where POD ($0 \leq \mathrm{POD} \leq 1$) is a constant value of 0.2 for the purposes of this simulation.

As each cell retains a Base Value after searching (see Equation 3.18), it is possible for multiple detections to occur from the same location, albeit with a reducing probability each time.

3.4 Results

3.4.1 Incomplete data sets

In this section the following terms are used:

Iteration Running the simulation for one algorithm with one set of values for variables on one map

Run Running the simulation for one algorithm with one set of values for variables on all maps, with the results averaged

[9] This equation is one used in the field of Search and Rescue — for other scenarios the equation for the change in value may be different

Set Running the simulation for one algorithm for all combinations of variable values on all maps, with the results averaged

Not all of the diffusion methods are able to complete in all circumstances. Several of them are able to produce situations where the robot becomes stuck — either staying in the same position, or moving along a limited and repeating path.

This can be caused by several reasons, the main one being the creation of false local peaks — high values on the Navigation Map which are not created by corresponding high values on the Base Map, so the robot reaching the top of the peak will not affect its cause and so will not move away.

This would normally cause the simulation to run indefinitely; for this reason the program will attempt to detect a stuck robot and end the simulation early. In a real life situation a stuck robot would potentially cause the mission to fail; it is possible to use additional complexity to detect and counteract them, but that is outside the scope of this work.

This can have effects with both Perfect and Imperfect Cleaning. With Perfect Cleaning, if the time taken to entirely clean the map of all cells at each initial Base Value is recorded, then an unfinished simulation would only allow us to record the time at which the last cell at that Base Value was visited. This will clearly be a shorter time than it would take to visit all cells at that Base Value.

With Imperfect Cleaning it is possible to measure the time the stuck robot took to find up to a certain number of people (the number that it was able to find before getting stuck).

There are several ways in which the effect of the incomplete data sets could be mitigated.

If an iteration becomes stuck then it could be removed entirely. The downside to this is that it would skew the results; entirely removing iterations which are sufficiently bad will make the algorithm look unnaturally good.

The actual time taken to reach the next person for that simulation (in the case of imperfect cleaning) should theoretically be infinite — the next person was never reached, as the robot became stuck. A second option would therefore be to trim all sets to the length of the shortest iteration. The downside to this is that if, for

example, one iteration only found a small number of people before getting stuck but all other iterations in that set found all people then a lot of useful data would be removed.

A third possibility would be to directly include the times for that iteration up to the point where the robot became stuck in the averages for that set; that is, included into the average time taken to find each number of people found up to that number. The downside to this is that the average time taken will then drop for later numbers, as the robot is liable to be running slowly in the run up to it becoming stuck.

A final option would be to continue the data for the incomplete iterations through extrapolation — extending the results to estimate the times at which later people would have been found had the robot not become stuck (for imperfect cleaning), and the time taken to clean all cells at a particular initial Base Value (for perfect cleaning). This has the downside that the data being used is artificial, and does not come wholly from actual simulation results.

In the results shown below it was decided, for imperfect cleaning, to extrapolate all data which had found 100 or more of the 500 people. Iterations which had failed to find 100 people were discarded, with the average for that run calculated from the other iterations.

The algorithm which is most affected by this, by some margin, is the Low Pass Filter method.

For perfect cleaning, the majority of methods failed to complete to such a degree that only a subset of algorithms are presented.

3.4.2 Computational complexity

The time axis on all graphs is based on the number of movements between cells taken by the robot. It does not take into account the amount of time the algorithm took to compute — some algorithms are more computationally intensive, but still able to complete the task in a smaller number of movements.

While this is a difficult metric to measure (as the speed of completion is based on many factors, including the speed of the processor and the number of concurrent

jobs), use of software profiling tools[10] allows for the collection of data on how long a program spends executing various parts of its code.[11]

For each algorithm, at the best measured variable values (as listed in Section 3.4.5), the program was profiled while converting a map to a fully diffused state ten times. The times are compared in Figure 3.17; it should be noted that as times are based on processor speed these should not be taken as exact times, rather as comparisons of relative speed.

It should also be noted that these are timing a complete change from the Base Map to the Navigation Map. For most methods this is not required for every iteration — updates are instead made based on small changes to the Base Map.

These graphs can be compared to Table 3.1, which shows that the Low Pass Filter and Absolute Distance methods have a greater computational complexity than the other methods. This makes the Low Pass Filter result unexpected, as it gives one of the fastest results for one of the most computationally complex methods. This may be due to implementation — the Low Pass Filter method uses a Fast Fourier Transform, which is liable to be better optimised than most of the other methods used.

3.4.3 Perfect cleaning

This simulation was run at a range of variable values for all of the different diffusion methods. Of the seven methods trialled, four failed to complete a single iteration. They did not get stuck in all cases (i.e. moving around a small, repeated area); they merely failed to visit all cells on the map. Of the remainder, two (the Highest Neighbour and Average methods) completed the entire set successfully, and one (the Low Pass Filter method) partially completed, becoming stuck on some iterations. Unfortunately, as the majority of the iterations failed to complete, this data set has also been discarded from the final results.

The two methods, then, for which there was a complete set of data were the Highest Neighbour and the Average algorithms. Data was collected at a range

[10] In this case, GNU gprof was used

[11] This measures the actual time spent in execution, not including time taken by the processor running other applications.

of variable values (D in Equation 3.1 and R in Equation 3.3). The best results from each were found to be at $D = 99\%$ (maximum value) and $R = 0.1\%$ (minimum value) respectively. The two were tested on two sets of maps, as described in Section 3.3, and the results from each map set were relatively consistent. Fig 3.13 shows the average from all maps across both map sets, comparing the best results from the two methods against those from Boustrophedonic motion. This shows that the Highest Neighbour method outperforms the Average method across the board — in that it succeeded in visiting all cells at any initial Base Value (representing the level of importance in visiting a cell quickly) in a lower time — and that both the Average and Highest Neighbour methods were improvements on the Boustrophedonic method for base values greater than 50% (i.e. the 50% most important cells to visit).

As would be expected, the boustrophedonic curve is relatively horizontal. The curvature is predominantly due to the uneven distribution of base values across the maps in the second set, as can be seen in Fig 3.11. As the method does not take the Base Values into account it is as likely to reach one probability at a particular time as another. It also covers the entire map in a minimum time, and so the area under the boustrophedonic curve is going to be lower than for all other curves.

Unfortunately there is little more that can be gained from this experiment, due to lack of data from the other algorithms.

3.4.4 Imperfect cleaning

The seven algorithms were tested using the same simulation, described in Section 3.3.3, allowing for a direct comparison between them. Each was run on maps from both map sets; the majority were tested against all maps from both sets, but due to time constraints some of the slower running algorithms were only tested against a subset. Table 3.2 lists the map / algorithm combinations which were tested.

Several algorithms have incomplete sets of data. To allow for comparison the incomplete results have been discarded if less than 100 people were found, or extrapolated and averaged otherwise.

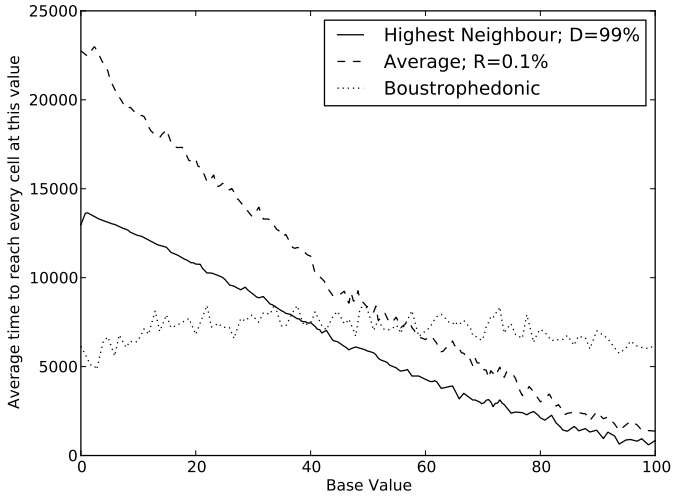


Figure 3.13: Complete cleaning — comparing Highest Neighbour, Average and Boustrophedonic methods, across both map sets. Graphs for Map Sets A (maps 0–9) and B (maps 10–19) followed a similar line. Data from other methods was unavailable.

Method	*Set A*	*Set B*
Highest Neighbour	All 10	All 10
Average	All 10	All 10
Median	All 10	6 of 10
Distance	6 of 10	6 of 10
Low Pass Filter	All 10	All 10
Conservative	10 of 10	6 of 10
Neighbour Difference	3 of 10	6 of 10

Table 3.2: Number of maps covered by each algorithm

3.4.5 Variable values

There now follows a table of all the variable values tested for each equation, with the values which gave the best results highlighted.

Method	Variable	Values
Highest Neighbour	Diffusion (D)	10, 50, 95, **99%**
Average	Reduction value (R)	20, 10, <u>1</u>, **0.1%**
Median	Threshold	**10**, 1, <u>0.1%</u>
Absolute Distance	Offset (O)	1, 10, <u>50</u>, **100** cells
	Radius (R)	5, 10, *20*, **30** cells
Low Pass Filter	Threshold	3.9%, 7.8%, *15.6%*, **39.1%**[12]
Conservative	Gradient (G)	**50**, 20, 10, 5%
Neighbour Difference	Threshold	10, 1, **0.1%**
	Diffusion (D)	**0.050**, 0.075, 0.100, 0.125[13]

(Emboldened values were found to be the best over all maps; underlined values were best for Set A (if different); italicised values were best for Set B (if different))

It is notable that for some methods it is dependent on the map set used which variable levels give the fastest response — implying that some values are better for a randomly generated landscape consisting of disjointed values with sudden variation, while some are better at more gently varying values. In almost all cases, however, the overall average has been more influenced by the better options for the second set than the first.

3.4.5.1 Notes on individual methods

(This is a brief overview of the methods, mainly covering unusual results — fuller notes, along with full result graphs, can be found in Appendix D)

As suggested in Section 3.2.1.5, the Low Pass Filter method failed to complete for all iterations bar one (39.1% threshold on Map 11); in all cases this was because the robot became stuck on peaks which did not exist, created by the Fourier transform.

[12] These correspond to the internal program values of 5, 10, 20 and 50, respectively

[13] Higher values were not tested, as values above $\frac{1}{\text{number of neighbours}}$ were found to be unstable

For this reason it was decided to modify the algorithm to be the Fourier transform of the Base Layer multiplied by the Base Layer. These are the results shown below; results for the pure Fourier transform are given in Appendix D.5.

The Conservative method failed to complete any maps in Set A where the gradient was 5%. Despite being the fastest gradient over maps in Set B (but by a sufficiently small margin to be negligible), it was decided to remove this option from all results from both sets.

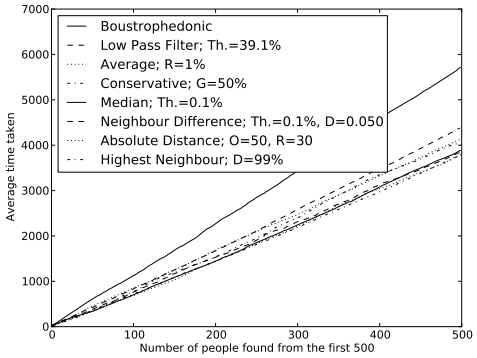
The Distance method is unusual for having an option that is better for maps from both sets which is different from that for each set individually. The results of the three options are extremely close, however.

The Neighbour Difference method failed to complete any of the maps from Set A with a 1% threshold and a diffusion of 0.100, or with a 0.1% threshold and a diffusion of 0.075, 0.100 or 0.125. These diffusion levels have therefore been entirely removed rather than extrapolated.

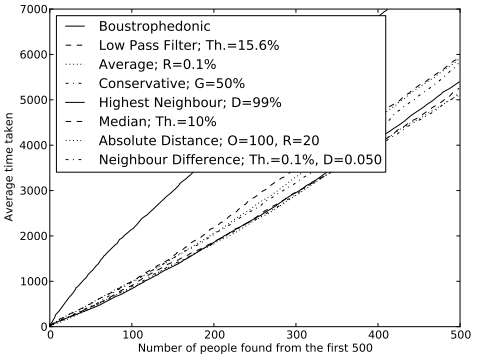
Both the Median and Conservative methods gave results which on first inspection are unexpected. The Median method gave its best results when the threshold was high, and the Conservative method when the minimum gradient was high — this appears somewhat counter intuitive; both give a Navigation Map which is close to the unchanged Base Map. There are minimal changes — but there are changes, and the resultant maps are sufficiently diffused to allow all of the simulations to complete (which was not the case when no diffusion is applied at all). Furthermore, having a higher threshold or gradient means that far fewer calculations are required per iteration, speeding up the time spent in calculation.

3.4.6 Comparison between map sets

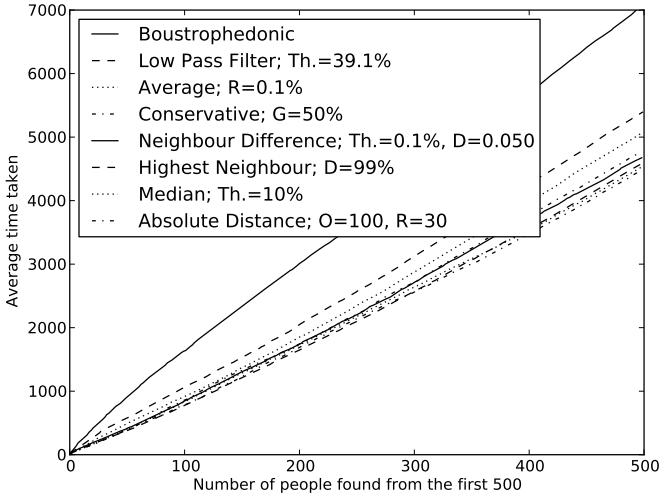
The methods are compared in Figure 3.14, comparing their fastest average across: (a) Maps from Set A. (b) Maps from Set B. (c) All 20 maps. Corresponding boxplots are shown in Figure 3.15.



(a) Set A



(b) Set B



(c) Average from both sets

Figure 3.14: The time taken to find 500 people, the probability of whose discovery is based on both the likelihood of their existence and on the likelihood of their being found if this is the case. The data is averaged over 10 base maps in each of two map sets, both individually and collectively.

3.4.6.1 Sets A and B, combined

Averaging across all 20 maps (see Figure 3.14(c)), the fastest methods in terms of the number of iterations taken to find 500 people are the Distance, Median and Highest Neighbour methods, closely followed by Neighbour Difference and Conservative. The difference between these methods is not large; it's actually far less than the range of results making up any of the averages shown (see Figure 3.15 to see the spread of data).

The Boustrophedonic gives — as is to be expected — a bad result, as it has no knowledge of Base Values. The Low Pass Filter and Average methods are slower than most; their average values still fall well within the range of the fastest method, however.

3.4.6.2 Set A

This map set is characterised by the maps having been created randomly. While this is less likely to be similar to the Base Map produced for a real life situation, it does give a far more difficult landscape to move over.

The Highest Neighbour and Distance methods are equal fastest, with the Neighbour Difference and Median methods not far behind.

The Low Pass Filter method is the worst of the diffusion methods, again with Boustrophedonic motion far behind.

In this set, unlike the combined set above, the boxplots (Figure 3.15) appear notably smaller — the speeds of the methods over the 10 graphs were therefore fairly consistent.

3.4.6.3 Set B

This map set was created using data taken from NASA's Shuttle Radar Topography Mission, to give a more realistic spread of data than that of Set A.

The Neighbour Difference and Distance methods come a joint first, with the Median and Highest Neighbour methods close behind, then after a short gap the Conservative, Average and Low Pass Filter methods. As above, the control using Boustrophedonic motion is considerably slower.

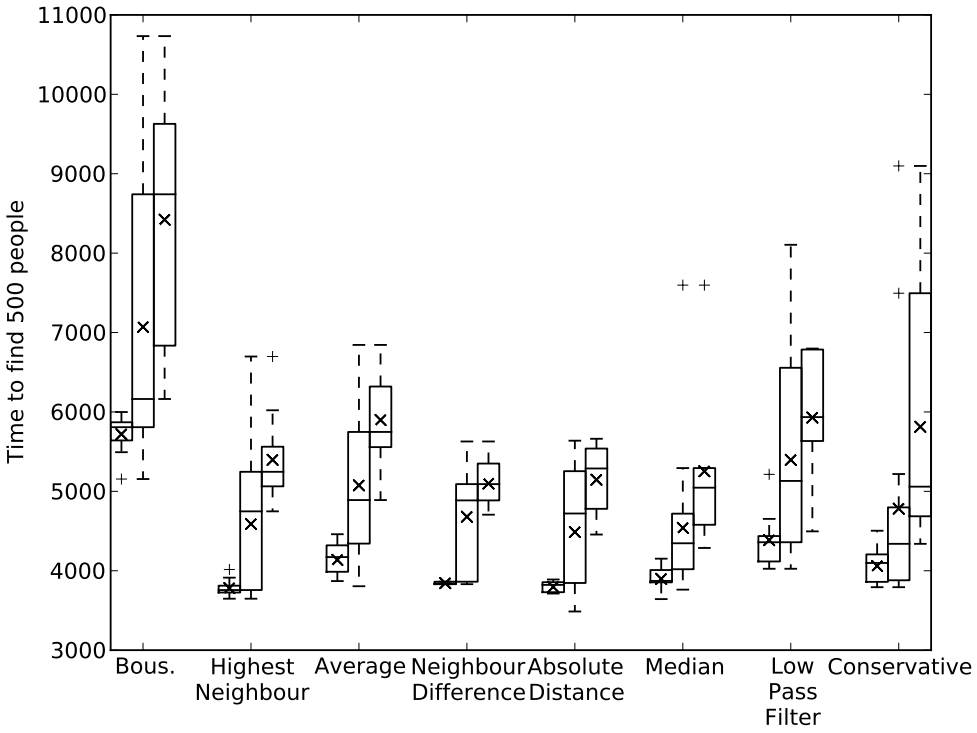


Figure 3.15: Boxplots of the range of times to find 500 people for each method, over Set B, all 20 maps, and Set A respectively. The average for each method is shown by a cross.

3.4.6.4 Differences

The various methods have been compared purely based on average speeds. This obscures, however, the spread of speeds produced by each — how consistent each method is across a number of maps.

Figure 3.15 shows the boxplots for each method for Set A, both sets combined, and Set B. Set B is notably slower in all cases than Set A, and predominantly has a larger spread of values.

Various diffusion methods have been compared against each other and against Boustrophedonic motion; it is important to check the effectiveness of the hill climbing algorithm without any diffusion, to make sure that it is the use of diffusion which is causing the improvement and not the use of a hill climbing algorithm. This is shown in Figure 3.16, where a hill climbing algorithm was used to navigate the robot over

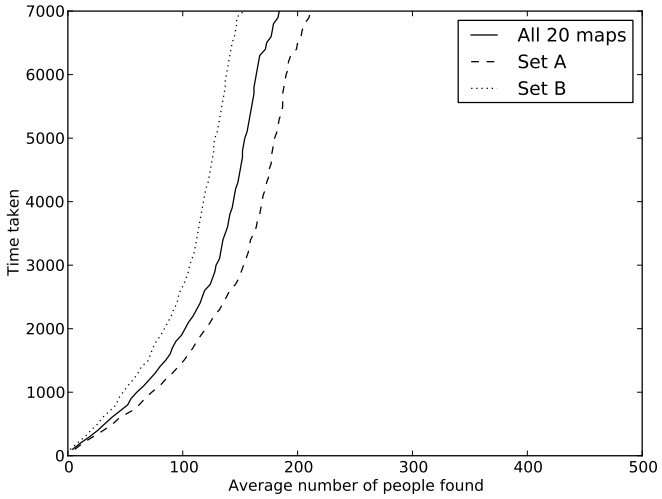


Figure 3.16: Average traces from a hill climbing algorithm across both map sets with no diffusion. No iterations complete to finding 500 people.

the Base Map instead of the Navigation Map.[14] As can be seen, no maps managed to complete to finding 500 people. It is interesting to note that the average was better for Set A than Set B; this is probably because if the robot managed to reach a high valued set of cells it would instantly have access to 100 at the same level, while in Set B it would only have as the high valued cells at the top of a peak.

3.4.7 Comparison between methods

While the various methods give little in terms of improvement in time between each other — the averages over all 20 maps are well within 1,000 iterations of each other, and the average of the slowest is within the error bounds of the fastest — it does demonstrate that use of diffusion is a sensible idea; using boustrophedonic motion (or a hill climbing algorithm without any diffusion) leads to far slower times, if the simulations completed at all.

Part of this may be because some maps are more difficult to search than others; this would therefore give a spread of data for all methods between the easier to search maps (in the extreme, this would be where all high valued cells are in close

[14] This is essentially a one step lookahead algorithm (see Section 2.3.3)

proximity) and the harder to search maps.

While there is no great difference between the methods, some have consistently appeared faster and some slower than others. The slowest diffusion algorithms for either map set are the Conservative, Average and Low Pass Filter methods. The fastest four — usually with little margin between them — are the Distance, Median, Highest Neighbour and Neighbour Difference methods. Of the methods, the one which has changed order the most is the Highest Neighbour method — fastest in Set A and 4th in Set B.

3.4.7.1 Incomplete data

Several of the methods failed to complete all maps at all variable levels. The table below shows the proportion of maps from each set that were not completed. (Other than on completion, simulations were stopped when the robot had remained within a 10 cell radius for 400 iterations, or had taken more than 10,000 iterations and was still incomplete. For compiling statistics, all data incomplete to under 100 people was discarded, and incomplete to over 100 was extrapolated).

Method	Variable values	Set A		Set B	
		Under 100	Over 100	Under 100	Over 100
Highest Neighbour		-	-	-	-
Average		-	-	-	-
Median		-	-	-	-
Distance		-	-	-	-
Low Pass Filter	3.9%	-	-	1/10	4/10
	7.8%	-	-	3/10	1/10
	15.6%	2/10	-	**4/10**	-
	39.1%	**1/10**	-	-	-
Conservative	Gradient=5%	5/10	5/10	-	-
Neighbour Difference	Threshold=1%, Diffusion=0.075	-	1/3	-	-

Neighbour Difference	Threshold=1%, Diffusion=0.100	1/3	2/3	-	-
	Threshold=1%, Diffusion=0.125	1/3	1/3	-	-
	Threshold=0.1%, Diffusion=0.075	2/3	1/3	-	-
	Threshold=0.1%, Diffusion=0.100	1/3	2/3	-	-
	Threshold=0.1%, Diffusion=0.125	-	3/3	-	-

Emboldened rows indicate ones which were selected as the best set of variables for one or more map sets.

3.4.7.2 Computational speed

These simulations were run over an extended period on a high specification server. As described in Section 3.2.3, this is not the scenario in which these algorithms are likely to be run in a real life situation, where they are more likely to be required on a low specification system for use in a low cost mobile robot.

The various methods were profiled in order to test how fast they were to completely diffuse a Base Map. Figure 3.17 shows the average length of time taken to fully diffuse the map. It is important to note that while the times are in milliseconds they are dependent on processor speed as well as algorithm complexity; as such they should be used more as a unitless comparator between methods. It should also be noted that while some methods require recalculation from scratch for any change to the Base Map (the Low Pass Filter and Distance methods, for example), others are able to update the Navigation Map to adjust for small changes by only updating the cells which are affected by the change. It is therefore possible that the number of iterations required — and thus the speed of calculation — may be substantially different at different points in the search. The graph should also be viewed in conjunction with Table 3.1, which shows the computational complexity of

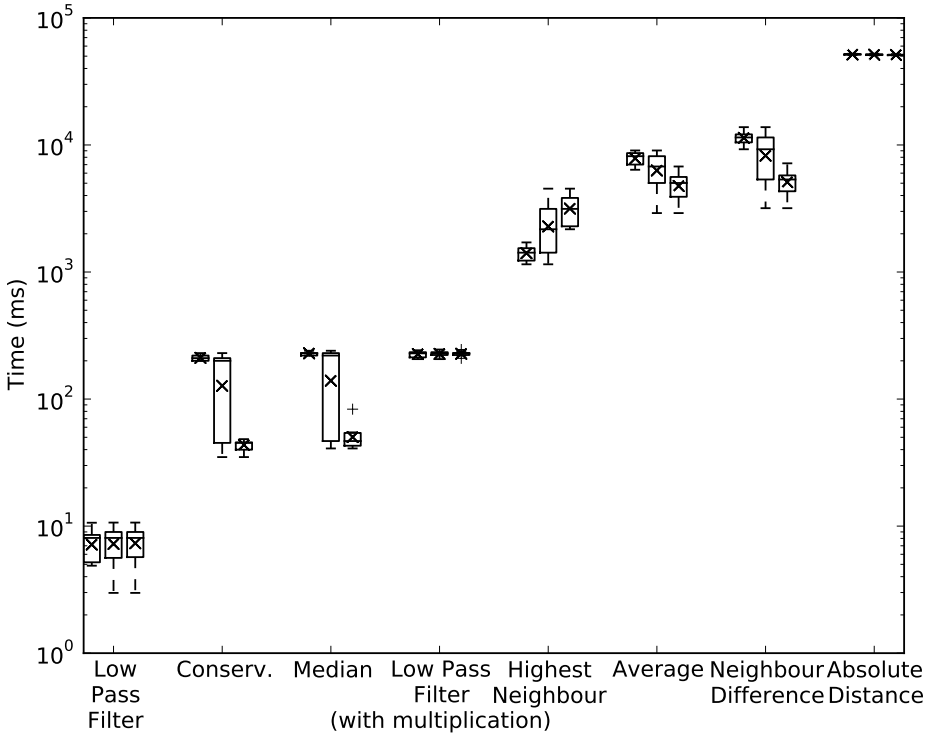


Figure 3.17: Profiling times for various methods, in ms. The three box plots per method are the times for Set A, all 20 maps, and Set B, respectively.

each method.

The variable levels picked were either the ones found best for all three of Set A, Set B and all maps together, or best for two of those three. For the pure Low Pass Filter method the variable level selected was the only one to complete a single map.

Of the methods shown, it should be noted that the pure Low Pass Filter method time, while considerably faster, failed to complete all but one map at that level.

The Conservative and Median methods are extremely fast to diffuse mainly because the variable levels which found 500 people in the fewest iterations were the minimum change from the Base Map that were tested, which required rather less processing time.

The Low Pass Filter method with Base Map multiplication failed to complete 1/20 maps at that level.

Absolute Distance is the most processor intensive method, which is understand-

able as it has a higher computational complexity than any other methods.

3.5 Conclusions

The aim of this experiment was to determine a method for searching an area such that cells with a higher Base Value were searched in the minimum time possible. Using diffusion gives a better (i.e. faster) result for the higher valued cells than brute forcing using a Boustrophedonic method. The overall time taken to search the entire map, however, increased dramatically — in some cases to over twice as long as with boustrophedonic motion, from the data available. It is therefore not the best method to use if a time limit is imposed upon complete coverage.

With imperfect cleaning, of the diffusion methods trialled the best four (across all data sets) were the Distance, Median, Highest Neighbour and Neighbour Difference methods, although the range of average times produced by the various methods was within 1,000 iterations. The Boustrophedonic method proved to be substantially worse than the best result from any of the diffusion methods.

Certain diffusion methods are more computationally intensive than others. If processing time is sufficiently important that search and travel times become negligible, the Conservative, Median, Low Pass Filter and Highest Neighbour methods become the best options. Some of the differences seen between processing times may be due to implementation, and future work should be undertaken to optimise the code used for each method. Processing time is important for low specification systems such as those typically found on inexpensive robots.

Selecting the method which is fastest (in terms of number of iterations), least computationally intensive, and most likely to complete, the Median and Highest Neighbour methods are probably best — although Median was one of two diffusion methods which gave their best results at variable levels which keep the Navigation Map very similar to the Base Map. This is unexpected, and requires further investigation.

To conclude, diffusion is a better method to use than Boustrophedonic motion; the best algorithm to select for diffusion depends on the importance of speed in the

calculation, but Median and Highest Neighbour are probably the most suitable.[15]

3.6 Future work

This experiment has made several assumptions, the removal of which would increase the scope of this work. One such assumption has been that the Base Map will remain static. In many situations (searching for a missing person who may be moving, tracking a forest fire that may be spreading, etc.) the probabilities are liable to shift over time, and the Dynamic Cleaning Problem (where the Base Map values are binary — clean or dirty) is a well researched extension to the Cleaning Problem.[45] Dynamically changing Base Maps are an important area to be investigated, and are one for which Boustrophedonic motion may prove unusable, or of considerably less use than the diffusion method.

The results from diffusion have been compared against Boustrophedonic motion, but have not been fully compared against a greedy lookahead algorithm (as described in Section 2.3.3).[16] Although this has the possibility of missing globally important areas by being caught on local minima, it would be interesting to see how much better the diffusion algorithm is in comparison when using realistic probability maps, as well as seeing how the computation requirements compare.

Comparison has been made (in Section 3.4.7.2) between the processing time required for the different methods to completely diffuse a Base Map to a Navigation Map. As in most cases during normal running only incremental changes are needed (where only a single cell in the Base Map has changed, instead of the entire Base Map), it would be interesting to profile processing time for an entire run.

As Bourgault et al.[37] said, 'any grid based approach ... is intrinsically subject to the "curse of dimensionality", and as soon as one needs to increase the search area, the resolution of the grid, or the number of dimensions in the state-space,

[15] Both are fairly evenly matched; Median is notably faster to reach a diffused state from a raw Base Map (see Figure 3.17), but Highest Neighbour is faster in terms of future updates (no quantifiable data is available for this, but Table 3.2 shows that Highest Neighbour completed all maps in a reasonable time, while Median could not be tested against all maps during the time available)

[16] Section 3.4.6.4 does consider a hill climbing algorithm without diffusion, which is equivalent to a one step lookahead, but multiple step lookaheads were not considered.

computational costs tend to get out of hand'. There are various ways in which the computational complexity could potentially be reduced, including using particle filters or Monte Carlo methods, increasing the cell size for all cells outside certain radius from the robot (that is, calculating values for multiple cells as one), or having a severe cutoff threshold for differences below which cells are not updated.

All of the methods calculate the values for the next interation based solely on past iterations, and most methods use the same calculation for each cell individually. This means that the problem can in most cases be parallelised, with each cell of the next iteration being calculated independently if a multithreaded processor were available.

Many of the methods investigated require all cells affected (each effectively a cellular automaton) to update between movement decisions. In some cases this includes most of the cells in the area. If this number could be reduced — for example by only updating cells close to the robot — then it would reduce the processing power required, but might reduce the effectiveness of the methods.

Another assumption has been that no obstacles impede the movement of the robot. Although use of diffusion should cope with obstacles without much difficulty, it would still be an important factor to test.

While much mention has been given of the use of multiple robots, this has not been included in the experiments. This extension to the problem is one which should lead to interesting further research.

Two methods — Conservative and Median — gave their best results at variable levels which gave the minimum change from the base map. This would seem to imply that their use should be avoided; but at the minimum tested variable levels they gave results comparable to those from other diffusion levels, and the small changes mean that the speed of calculation is high. Further experimentation should be made to see more precisely at what variable levels these run optimally, and at what stage they cease to complete, taking into account that entirely undiffused maps do not complete.

The results from the Low Pass Filter method — as mentioned in Section 3.2.1.5 — may not be entirely representative. The method was let down because of the

number of false peaks created; these may have potentially been caused by ringing from the harsh filter used. A future experiment to repeat the Low Pass Filter method using a less agressive filter (such as a Gaussian filter) would allow for a better comparison.

As mentioned in Section 3.2.2, for all methods described in this chapter the neighbours of a cell have been specified as being the four adjacent cells and the four diagonals. This work could be extended to compare the methods with different specifications of neighbour.

Another possible extension would be to use 3D navigation maps, to take into account the fact that UAVs are able to fly at varying heights. A UAV flying close to the ground searching for a missing person will be able to see over a smaller area than one flying at a high altitude, but it will have a higher probability of detecting a missing person — meaning that it will reduce the base values it passes to a greater extent. This could be represented by creating a 3D base map, where the values for each cell are comprised of the cumulative POS from the POA of all cells visible on the ground (taking the POD of the UAV into account at that altitude). These values would then be diffused in three dimensions, giving the UAV a flight path that included changes in altitude; automatically moving it closer to the ground for areas which are harder to see.

Chapter 4

Unmanned Aerial Vehicles

Having investigated search and rescue and a diffusion based navigation method for UAVs to plot a route over probability maps, it becomes important to understand more about UAVs — both as a concept and about their capabilities, especially in regard to situations such as search and rescue.

An unmanned aerial vehicle — or UAV[1] — is, at its simplest, a manmade device that flies and which does not contain a human. In this sense UAV covers a large range of machines, from the most basic of small scale model aircraft upwards. The term carries a variety of connotations, however, depending on the situation in which it is used.

Possibly the biggest variation in meaning is based on the autonomy of the vehicle, but UAVs also cover a wide range of aircraft types[57] — from fixed-wing, to lighter than air, to a wide variety of rotor based aircraft (traditional helicopters, multirotor systems (quadcopters, hexcopters, etc.)), as well as more novel designs (such as those in the DARPA UAVForge competition,[2] and various ornithopter UAVs based on the flight of birds or insects[59]). They similarly come in a variety of sizes and weights, from extremely small Micro Aerial Vehicles (MAVs) that can be held in the hand[3] to aircraft that are substantially larger than their manned counterparts. Different

[1] Sometimes also referred to as a 'drone', or as a UAS (Unmanned Aircraft System, which includes ground stations, communications network, etc.) UAVs can also be referred to as Remotely Piloted Aircraft (RPA) when there is a human in the loop.

[2] `http://www.uavforge.net/`; including novel UAVs such as the GremLion, X-MAUS and Quad-Shot — although no entries completed the baseline scenario.

[3] For example the 10cm DelFly Micro,[59] weighing 3g

situations will call for different UAV designs; rotor based aircraft, for example, are more manoeuvrable, while fixed-wing aircraft tend to have a longer flight time.

Nikola Tesla first described the modern UAV in 1915, saying that 'it would be possible ... to direct an ordinary aeroplane, manless, to any point over a ship or an army, and to discharge explosives of great strength from the base of operations'.[60] Most UAV development since has taken place in a military context,[57] for obvious reasons. Unmanned aircraft can be smaller, cheaper, more stealthy, and — most importantly — more expendable than an aircraft piloted by a human. They also open up new options, such as the ability to monitor large areas without tying up large numbers of personnel. Military UAVs are sometimes armed, and it is partly for this reason that the level of autonomy allowed to them is usually extremely low. This helps to reduce incidents of UAVs accidentally opening fire on their own troops, for example. Instead, they will usually have a 'human in the loop'[4][61, 'The idea that UAS are "unmanned" is a misnomer because trained and professional Soldiers operate and maintain Army UAS'] — a person who is remotely controlling the aircraft, or (more commonly) multiple aircraft. In this case the UAV will often have enough self control to keep itself flying and follow basic instructions (such as flying to a particular point, or loitering around a target), but the higher levels of control — the decision making of where to go and what to shoot — will be taken by a human,[5] and potentially by a human on a different continent to the UAV.[62] Military UAVs are used for a number of reasons, including reconnaissance, combat, and cargo transport.

Away from the military context, much research with civilian UAVs focuses on attaining complete autonomy, although in a commercial context this autonomy is still for the most part limited to takeoff, landing, and simple instruction sets such as following GPS waymarks or circling set points.[35]

There are a vast number of civilian UAV uses[57] — including aerial photog-

[4]Often referred to as a 'Pilot', 'Navigator', or 'Combat Systems Officer' (in the case of UAVs with weapons).

[5]This is known as 'semi-autonomous' — a level between teleoperation (where everything is controlled by a remote human) and autonomous (where — in a military context — the majority of decisions are made by the UAV, but a human is still required to monitor and remotely assist as necessary).

raphy, border security, police chases, detecting and combating fires,[63; 64] farming (monitoring livestock, spraying pesticides,[65] agricultural monitoring,[66] etc.), providing network coverage in emergency situations,[67] searching for fish, traffic monitoring,[68] various types of research (on climate change, atmosphere, etc.), and search and rescue (see Section 4.5.1), amongst many others.

This chapter will now cover first an overview of the current state of UAV autonomy, and then will look in more detail at the work of other UAV research groups.

4.1 Autonomy

The ideal UAV would be fully autonomous — after being given an overview of the situation and sufficient additional information it would be capable of breaking contact with the human controller, performing a suitable set of tasks (possibly in collaboration with humans and / or other UAVs), resolving any situational changes as they occurred without requiring human intervention, and then returning once the situation was at an end. This behaviour would mean that the UAV could be treated in the same way as an additional human — it would be capable of using its own initiative, and while humans could contact it to update the situation (giving further orders, say, or more information) it would not be necessary to monitor it continuously.

Full autonomy is a long way off, and there are many aspects which require research. One major aspect is Sense and Avoid. When UAVs are flying in free space there is little or no risk of collision. They can be instructed to take off, fly at a certain altitude to a given position, follow a set of waymarks, circle a point, and land — but most UAVs make the large assumption that there is nothing in the way when they fly. If a UAV was being used in closely confined areas (for example at low levels where there are trees or humans, between buildings, indoors, or in caves) then they will need to be able to navigate around obstacles. This can lead to greater problems when UAVs are flying in the proximity of other aircraft (either other UAVs or manned aeroplanes) — this could easily occur when flying close to an airport, or flying sufficiently high off the ground; alternatively the UAV may be working with the other aircraft. It is especially important in these situations, where a collision

would risk not only the UAV but also the aircraft that it hit, for the UAV to be able to sense and avoid potentially moving obstacles. The US Airforce's Airborne Sense and Avoid (ABSAA) program,[69] to produce a robust and platform agnostic system for UAV / manned aircraft avoidance, is still in its early stages.

Reliable Sense and Avoid is currently one of the biggest stumbling blocks for integration between UAVs and normal airspace.[70] In the UK work is currently underway to increase the areas available to UAVs. The West Wales UAV Centre at ParcAberporth is the world's first UAV airport and testing site, covering 500 square miles[6] which in June 2011 the Civil Aviation Authority designated for both UAV and manned flight.[71] On a wider scale, the ASTRAEA[7] scheme is a consortium of UK companies working on regulations to permit autonomous aircraft into civil airspace.

While these will open a path for UAVs to become considerably more active in the wider environment, there is still a great need for robust obstacle avoidance.[65]

Mellinger et al.[72] describe an autonomous trajectory planning system which is capable of manoeuvring quadcopter UAVs through extremely small openings (such as would be found in an indoor environment) and at high speeds — some openings require the quadcopter to rotate sideways, with less than 3" clearance on all sides. This is an impressive feat, but does require total knowledge of the positions of the UAV and all obstacles at all times — the UAV's location is taken from a motion capture system and for the most part the obstacles are preprogrammed, although some experiments do involve flying through a thrown hoop also tracked by the motion capture system.[73]

Ferrat et al.[74] have developed an omnidirectional camera which can be used for sense-and-avoid. This has a laser mounted above it which emits a cone, angled downwards, running 360° around the UAV. The closer an object is to the laser, the higher the point at which the beam intersects it. Thus the camera can plot the point at which the laser is visible, with the horizontal position in the field of view giving the direction of the obstacle and the vertical position its distance from the

[6] 500 miles from 3,320 square miles available, covering land and sea. Blocks of this are made available to UAVs when required

[7] Autonomous Systems Technology Related Airborne Evaluation and Assessment

UAV. This makes the assumption that objects are large and flat — if a thin rod were approaching the UAV, say, it would only intersect with the laser at one point. Good results are reported, but the laser being used has a maximum range of 1–2m.

Roberts et al.'s Eye Bots[75] autonomously explore an area by perching on the ceiling, checking for close obstacles, and calculating trajectories from this for any UAVs moving underneath. This does allow for fully autonomous obstacle avoidance, but requires a specific setting and the aid of multiple UAVs.

De Croon et al.[59] describe a small ornithopter (flapping wing) UAV, which is capable of fully (albeit limited) autonomous indoor flight. The DelFly is capable of obstacle avoidance by comparing object textures. Objects are assumed to have for the most part similar textures which are for the most part different from their surroundings. As an object is approached its colours and detailed texture become increasingly visible, while other objects become increasingly hidden. The UAV analyses the image in its single camera and detects as the object starts to fill its field of view, allowing it to take evasive action.

Yang et al.[76] split the problem of sense and avoid into two, and focus on the avoid part — offering collision avoidance strategies for fixed-wing UAVs, once they have detected another aircraft on a collision trajectory.

While full sense-and-avoid is currently unavailable, there are many UAVs capable of lesser levels of automation. Basic autonomous takeoff and landing (vertically for a rotor based UAV, and with a runway for a fixed-wing UAV) has long been possible for UAVs, but there are now several examples of more interesting landing and takeoff.

There are UAVs which can perch onto small surfaces such as a human hand[77] or a wire,[78] using a stalling technique which mimics a landing bird. Others are capable of attaching themselves to walls using retractable spines, which grip against rough surfaces.[79] There are even designs for UAVs which attach to walls by intentionally flying directly into them, affixing themselves using a retractable needle on the nose of the aircraft.[80] These allow a UAV to land in situations where there is a lot of debris on horizontal surfaces but the vertical walls are clear.

Aerial vehicles in general — and rotor based aircraft in particular — tend towards instability, and there has been much research in creating a stable platform; not only

from the normal destabilising effect of flying, but to a greater extent with things such as balancing inverted pendulums[81] and other top heavy objects. Palunko et al.[82] describe a quadcopter UAV designed for carrying unstable weights, which autonomously compensates and retains a high level of stability — adjusting its flight paths to reduce the swing of the object tied underneath it. Muller et al.[83] have developed a UAV with attached racquet, which is capable of hitting a ball — either returning a ball thrown by a human, or two UAVs bouncing a ball between each other, or a single UAV continuously bouncing a ball into the air.

There are also several examples of UAVs which gain autonomy through learning — either from watching an expert human pilot and learning from their responses,[84] or from trying a task repeatedly until it can take into account any uncertainties.[85] The learning of one UAV can then be passed on to other robots, saving them having to repeat the same learning process.[86]

4.1.1 Control structure

The different levels of autonomy lend themselves to a hierarchical control structure, where each level is able to instruct the level below. At the lowest point is control based around keeping the UAV flying — taking the example of a quadcopter UAV, calculating the correct settings for the rotors in order to set the UAV attitude and velocity.[65] The next level up from this would instruct the base level with the required velocities and attitudes to move as desired — say, from its current position to a GPS waymark, taking into account feedback (wind, etc.)

This continues upwards, taking increasingly complex and abstracted sets of instructions and passing them down the different levels to produce the desired effect.[87] At higher levels there start to be more decisions that the UAV will have to make. A simple example of this would be if the UAV has been given a search task but has only limited fuel, it will at all points have two options — to continue with the search, or to return to base for refuelling.[34] The exact decision over where the UAV flies next could be based on a large number of factors — the shortest path to a goal, the position of detected obstacles, the level of fuel, or the importance of reaching a position to allow reliable communication.[35] In some cases the decision may

at first appear obvious but might turn out to be more complex; it may not always be the case that a UAV running low on fuel should immediately return to base, if by potentially sacrificing itself it is able to complete its mission. A UAV that reports the position of a missing person, lands beside them and runs out of power would be better than a UAV that had full power but did not find the person; similarly, a UAV that was collaboratively carrying an object would be better off finishing the job and not being able to get all the way back to base than to release the object and cause potential damage to the other lifters. The conflicting instructions may also come from external sources — if a UAV or swarm of UAVs are given a long list of tasks to perform they must decide how to allocate them, and in what order.[88]

A hierarchical control system also allows for hardware abstraction — the upper control levels need not know exactly what signals to send to the motors; they merely instruct the lower levels to take care of it. This means that there can be some degree of platform independence — the same high level code can, in theory, be used across a range of heterogeneous UAVs.

To achieve the different levels of autonomy humans can issue commands to the various levels of the hierarchical control.[89] Depending on the context, a human may wish to use the UAV at any level of autonomy from teleoperation (for example where the UAV controls the exact rotor speeds but the user gives directional orders) to full autonomy (where the user gives a basic overview of the situation and leaves the UAV to make its own decisions as to which tasks to carry out based on its observations of context and the user's activities).

There are some contexts where full autonomy — even an idealised and error free full autonomy of the future — may be imperfect. If the autonomy is not perfect then being able to revert to a skilled human pilot for a tricky operation, or an experienced tactician to direct and divide tasks more ably between UAVs, would be advantageous. If the autonomy was perfect this may still be the case, if the user has a very specific goal in mind which they are unable to accurately put across to the UAV. Kvarnstrom and Doherty[90] describe an emergency situation where a number of UAVs are being given the collaborative task of distributing crates of medical supplies to the wounded. At one extreme, giving the UAVs full autonomy

to decide exactly what to deliver where could cause problems with human rescuers on the ground (leading to duplication of effort, etc.). At the other extreme, where the humans have given an exact plan as to what should be delivered where and when, there is more chance for problems to occur when the situation changes and the rigid plan fails to be flexible. It is important, therefore, for autonomous robots to be given the correct level of autonomy for the given situation.

4.2 Other research groups

There are several research groups worldwide working on UAVs,[91; 92; 57] and in particular UAVs working collaboratively or in close proximity. Having given an overview of the current state of UAV capabilities in the previous section, this section will give more details on some of the most significant of these groups.

The groups can be classified by the focus of their research interests — whether their UAVs fly indoors or outdoors, whether they are rotary, fixed wing or ornithopter, whether they are fully self contained or whether they require external connections, and the manner of the tasks which they undertake. These are summed up in Table 4.1.[8]

While many other universities and companies not listed are also doing research into UAVs, this covers the larger, relevant research groups.

4.2.1 STARMAC

The Stanford/Berkeley Testbed of Autonomous Rotorcraft for Multi-Agent Control (STARMAC)[93] is a multi agent UAV testbed. They have six autonomous quadcopters, equipped with sonar, IMU and GPS, and capable of flying indoors and outdoors. Stanford have done research into full autonomy. They have developed 'apprenticeship' learning algorithms, which watch a human repeatedly doing a manoeuvre and then recreate the 'perfect' line — the average of the moves performed by the human, to remove any minor differences between each attempt. This allows the

[8] In this case 'Self Autonomous' is defined as being entirely self contained, while 'Remote Autonomy' is defined as requiring an external connection, whether that is for an external positioning system or for external processing

	Environ.		Type			Auton.		Purpose					
Group	I	E	R	F	O	S	R	F	L	C	R	M	S
STARMAC	X	X	X	X		X		X	X				X
GRASP Lab	X		X				X	X			X		X
ETH Zurich FMA	X		X				X	X	X	X			X
	X		X			X				X			
UltraSwarm	X		X	X		X				X			X
DelFly	X				X	X				X			
Berkeley Aerobot		X	X	X		X				X	X		
MAGICC Lab	X	X	X	X		X			X	X	X		X
MIT — RAVEN Lab	X		X				X			X	X		X
		X		X		X					X		
Swarmanoid	X		X			X				X	X		X
ORCHID		X	X	X		X					X	X	X
SUAAVE		X	X			X					X	X	X
UAVTech		X	X			X					X	X	X
NASA		X		X		X						X	
QinetiQ		X		X			X					X	

Table 4.1: Classification of UAV research groups. The data refers to the environment in which the UAVs are used (I = Internal, E = External), the type of UAVs (R = Rotary, F = Fixed wing, O = Ornithopter), the level of autonomy (S = Self autonomous, R = Remote autonomy), and the primary focus of the tasks that they have undertaken (F = Fine control, L = Machine learning, C = Collision avoidance, R = Route planning, M = Monitoring/Duration flight, S = Swarming/Collaboration)

UAV to perform aerial acrobatics as well as an expert pilot by repeatedly watching their performances.[84]

They have also developed a method for fixed-wing UAVs to attach themselves to walls; the UAV detects the wall with an ultrasonic sensor, files up into a stall, and 'perches' on the wall using retractable spines on the underside to get a purchase on the rough surface. When it wants to move again it detaches from the wall, flies backwards and then rotates to fly off again.[79] Landing on walls is a useful skill, as walls are less likely to be obstructed than horizontal surfaces — particularly in a disaster situation — and as it is already above the ground the plane will consume less energy getting airborne again.

4.2.2 GRASP Lab

The GRASP Lab at Penn University has produced a lot of research into UAV control, with many innovative results. Their range of quadcopters — some small enough to fit in the palm of a hand — are controlled by a comprehensive motion capture system (as well as internal IMUs) which is able to pinpoint their exact position at all times with a high degree of accuracy.[94][9] Although this is an unrealistic situation for most robots in the real world (particularly if engaged in search and rescue activities), it has allowed them to focus on autonomous control algorithms, giving the UAVs great manoeuvrability. Swarms of heterogeneous UAVs are able to autonomously calculate their own trajectories using a mixed-integer quadratic program for close flying and obstacle avoidance.[95] They take each other's planned trajectories into account, and will automatically move around each other to plan routes including decisions such as which UAV should go first through a narrow gap. They also have capabilities for lifting objects together,[96] and for collaboratively building structures.[97]

4.2.3 ETH Zurich Flying Machine Arena

ETH Zurich has a 10x10x10m lab for the development and research of small UAVs. This has an 8 camera motion capture system for localisation, using retro-reflective

[9] 'The system can be run at or below 375 Hz ... [and] experimental tests show that the deviations of position estimates for single static markers are on the order of 50 microns'

markers attached to the aircraft. Their work covers aerial acrobatics,[98] UAVs capable of balancing inverted pendulums[81], and juggling UAVs.[83] UAVs are able to learn by repetition to improve their work,[85] and can pass on their learning to others.[86]

They have also developed the muFly fully autonomous micro-helicopter, which is capable of obstacle avoidance in real time using an omnidirectional camera with a laser mounted above it (described in Section 4.1)[74]

4.2.4 UltraSwarm

The UltraSwarm project — and it's predecessor, the Flying Gridswarm — at the University of Essex[99] has created a swarm of UAVs that flock[100] (based on Boid algorithms — see Section 4.4), and which run a distributed Beowolf computer cluster. This allows for a distributed intelligence, where the swarm as a whole can make decisions and perform calculations that the UAVs would be unable to do as individuals. It also gives a level of resilience, as the loss of a single UAV from a swarm is less likely to cause the task to fail.[101] UltraSwarm used helicopter UAVs, while Gridswarm used fixed-wing UAVs.

4.2.5 DelFly

The DelFly project[59] is developing ornithopter UAVs at Delft University of Technology, in the Netherlands. The flapping wing micro UAVs that they have produced are designed to imitate birds and insects. The smallest is the DelFly Micro, which is 10cm wingtip to wingtip and weighs 3g. Despite their small size and weight they are capable of autonomous flight, albeit only in a limited way — they are able to fly around an area avoiding obstacles by detecting object textures with a visual camera (described in Section 4.1).

4.2.6 Berkeley Aerobot

The hardware for the BEAR (Berkeley Aerobot) Project is based on taking commercially available helicopters and fixed-wing aeroplanes and fitting them with autonomous control systems. This fleet includes six large, fully automated helicopter

based UAVs, ranging from 2–4m in length. The UAVs communicate via 802.11 wifi, and each has an INS, GPS, onboard computer, and a variety of other sensors (laser range-finder, stereo camera, ultrasonic sensor, etc.). The large size of the platforms and their high weight capacity (some are petrol driven and able to lift a 30kg payload) mean that they are able to carry more powerful computers and more advanced sensors than the UAVs of most other research groups. They are capable of simultaneous localisation and mapping, using a laser scanner to build up a 3D map of their surroundings and plot courses to avoid obstacles.[102]

4.2.7 MAGICC Lab

The Multiple Agent Intelligent Co-ordination and Control (MAGICC) lab at Brigham Young University has performed several tests with external and internal UAVs. Smaller UAVs tend to be better at navigating small environments than larger ones, but larger UAVs tend to have much longer ranges. They have developed a system where a micro-UAV (MAV) is captured by an autonomous mothership, to allow for longer range flights. This means that the MAV could be dropped by the mothership, and then captured again for the flight back to base — the mothership releases a 'drogue' behind it, which the MAV tracks and connects to.[103]

For the internal UAVs they have developed SLAM systems[104] based around the Kinect 3D camera.[10] They have also researched into route planning algorithms that optimise the probability of detection (see Chapter 2) by optimising the visibility of objects to the UAV. This takes into account the size of the object in the camera image based on the UAV position, the phong illumination and the contrast between the object and its surroundings, and the length of time the UAV spends looking at that area, amongst others, and the UAV's trajectory is created based on the path which gives greatest cumulative probability of detection.[14] Other research includes collision avoidance (based around repulsion of intersecting trajectories), and cooperative control using multi-agent learning.

[10]The Microsoft Kinect camera uses an infrared projector with an offset IR camera. The projector projects a pattern, and from the deformation of the pattern as seen from the offset camera it can calculate the distance from the projector to the object, building up a three dimensional image of it.

4.2.8 MIT — RAVEN Lab

MIT's RAVEN Lab has a large room with 18 motion capture cameras which track reflective balls attached to the UAVs to give real time positioning.[105] The UAVs are for the most part standard off the shelf radio controlled vehicles, with no structural or electronic modifications, which are controlled by a centralised computer that interfaces directly to the trainer port on the RC transmitter.[106][11] The lab also has autonomous ground vehicles, which work in parallel with the UAVs. The controlling computer is able to set up a system of heterogenous vehicles which autonomously share tasks. The UAVs must complete a series of tasks, organising this by giving each task a set waypoint at which it can be done and a list of which UAVs are capable of doing it, along with timing constraints to say which order they must be done in. The trajectories for the UAVs are then centrally calculated using a mixed integer linear program[12] to optimise the routes taken and to take obstacle avoidance into account.[107]

MIT also has an outdoor multi-UAV testbed,[105] which is limited to simple scenarios of up to 3 UAVs. While this provides a more realistic environment for UAV testing, with real world factors such as wind and communications difficulties, it is considerably less structured than the indoor test bed, requires more people to run experiments, and cannot be used to reliably run longer experiments as it can only be used during daylight hours when weather conditions are suitable. The indoor RAVEN Lab was created partially to allow for long duration experiments.[91]

4.2.9 Swarmanoid

The Swarmanoid project was a Future and Emerging Technologies project funded by the European Commission, researching into the creation of a distributed heterogeneous robot system, where different components are individual autonomous robots

[11] Many radio controlled aircraft will have a 'trainer port' included on their transmitter, to allow connection to the controller of a more experienced pilot who can teach an untrained person how to fly the device.

[12] Mixed integer linear programming refers to a linear optimisation problem where the constraints are based on integer variables. In this case the equations to be minimised are the distances travelled by the aircraft, with constraints based on start and end positions and on the positions of other aircraft

which can collaborate on tasks — in some cases by joining together to form larger robots.[108]

Three types of robot have been created; 'Foot' (small autonomous ground vehicles); 'Hand' (autonomous robots with grippers which can climb and move objects); and — most relevantly — 'Eye' (autonomous UAVs). These work in collaboration to explore areas, find objects and retrieve them. All processing takes place on board the individual robots.

The Eye Bot swarm works collaboratively for localisation, with each UAV in one of two modes.[75] In the Beacon mode it will be magnetically attached to the ceiling, where it will map obstacles and provide directions for UAVs in Explorer mode. In Explorer mode it will fly towards the edge of the known area, and then attach itself to the ceiling to become a Beacon and extend the knowledge of the swarm; scanning the area for obstacles and items of interest, detaching from the ceiling when no longer needed and flying to a new area.

The system works without a priori knowledge of the environment, and without any external positioning system (such as GPS or motion capture cameras).[109] Each explorer/beacon can calculate their relative position using a series of IR transmitters and receivers, comparing received signal strength in an array of receivers to obtain distance and bearing. The beacons can detect near obstacles by seeing the reflection of their own transmitters, and uses this data to plot a trajectory for the explorers (either to nearby unexplored space, or on to a beacon nearer the edge).

Although this does require specific characteristics for the space (walls must rise to the ceiling to be taken into account as obstacles; ceilings must all be at a similar height, etc.), it is reported to work extremely well in many real world situations (such as in office buildings).

4.2.10 ORCHID

The ORCHID project is a large group of research institutions working on 'Human-Agent Collectives' — the ways in which humans and robots can work together towards common goals. Towards this goal they are looking at three distinct application domains: Smart Grid (covering national electricity grids), Citizen Science (crowd-

sourcing scientific research through volunteers), and Disaster Response. This third is the most relevant, in connection to this thesis.

The project covers the use of a wide number of technologies for disaster management, including UAVs for surveillance and search and rescue, with differing levels of autonomy based on the situation and amount of human intervention.

As part of this they have developed a system of autonomous, collaborative UAVs flying over a disaster area to provide live aerial imagery. The humans on the ground can request for the UAVs to travel to a specific site (a burning building, for example, or a crowd) with a level of importance. The swarm then takes the list of locations which need monitoring and, using the max-sum algorithm,[110] work as a team to cover them; taking into account things like the length of time that each site needs observation, and the battery life of the respective UAVs.[88]

4.2.11 SUAAVE

The work of the SUAAVE[13] consortium[35] shares many similarities to this thesis. Their goal is to produce a swarm of UAVs which are capable of searching for a target — specifically, a person in open countryside. This requires full autonomy, including obstacle avoidance and swarm behaviour, and distributed search strategies.[34] The UAVs are designed to have robust communication, allowing for signals to be forwarded between each other for sharing search information and for reporting back to base.[35]

Research has been tested on hardware platforms as well as in simulation, specifically on a swarm of off the shelf quadrotor UAVs[34] which have then been heavily modified to augment their original basic autopilot (capable of following GPS waymarks) with a considerably more advanced set of processors and sensors (visual and IR). The UAVs are capable of target detection based on visible images.[111]

Their work includes probability map creation,[23] and the effect of altitude on object detection — leading to a search strategy based on taking a high altitude general overview and then descending to view areas of interest in higher detail.[24] They have also researched into a variety of search strategies, specifically Greedy (where

[13] Sensing, Unmanned, Autonomous Aerial VEhicles

UAVs go to the neighbouring cell which has the highest probability of containing the target), Potential (based on virtual attractive and repulsive potential fields), and Partially Observable Markov Decision Process (where decisions are made based on their expected reward or cost over time).[52]

4.2.12 UAVTech

The Autonomous Unmanned Aerial Vehicle Technologies Lab at Linkoping University in Sweden has produced a lot of research into UAV autonomy and control.[90] Their work includes human/UAV collaboration,[89] UAV localisation when GPS is unavailable,[112] and the use of sensor fusion for detecting bodies in a search and rescue situation.[113]

4.2.13 NASA

NASA's UAV research topics have been many and varied. They have produced a number of UAVs, including the high altitude Pathfinder and Helios solar powered UAVs.[114] They have produced innovative sensors, such as the UAVSAR UAV synthetic aperture radar[115] which has been used for monitoring the effects of landslides, ground deformation after earthquakes, oil spills, and the state of polar ice sheets. Their UAV missions have also been diverse — ranging from fire detection, to agricultural monitoring of coffee ripeness[66] and detection of vinyard frost risks, to chemical sensing and airborne surveillance for use during terrorist activity. In wider context, they've produced reports into the capabilities of civil UAVs for use in Earth Science, Land Management and Homeland Security,[116] and are working towards UAV integration into civilian airspace.

4.2.14 QinetiQ

QinetiQ is heavily involved in UK UAVs, as it runs and maintains ParcAberporth (the UAV airport and testing ground in Wales). They have also produced UAVs themselves; specifically the solar powered Zephyr — a high altitude, long endurance aircraft which runs on solar power during the day and on lithium-sulphur batteries at night. The Zephyr has a 22.5m wingspan, containing the solar cells, weighs

around 50kg, and in 2010 broke the world record for endurance flight at 336 hours 22 minutes, flying to an altitude of over 70,000ft.[117]

4.2.15 Other commercial groups

There are several other commercial groups working on UAVs. Lockheed Martin, for example, produces a number of unmanned military and commercial aircraft; as does Boeing, including the Phantom Eye surveillance UAV — a high altitude, long endurance craft with a four day flight time. BAE Systems has also produced a wide range of UAVs, and is conducting research into autonomy. In 2005 BAE's HERTI aircraft was the first UAV to fly in the UK with CAA certification, and they are currently developing both semi-autonomous combat stealth aircraft (the Taranis, produced as part of the UK's Strategic Unmanned Air Vehicle Technology Demonstrator Programme along with other collaborators), and fully autonomous combat aircraft (the Mantis).

4.3 Sensors

There are many different sensor types, used both for calculating the position and trajectory of the UAV and for locating objects of interest.[57] Commonly positions are calculated by GPS outdoors, and indoors either by a range of 3D tracking sensors[94; 105][14] or by an on board SLAM based[102] system.[15] Other systems also exist, however.[109] As the complexity of the area being flown through, or the number of UAVs in the area, increases so too does the importance of accurate data from sensors.

Many different sensors can be used for both localisation and task completion, including laser range-finders,[74; 102] cameras (visual and IR), microphones, gas sensors, biological sensors, radiation sensors, etc.[57] The majority of sensors used by UAVs, however, are vision based — partly due to use cases, and partly due to

[14] 3D tracking sensors can give extremely good results, but except for extremely specific situations would be unavailable in real world situations.[75]

[15] This can cause difficulties from the typically low processing capabilities of UAVs, unless the processing is offloaded to an external computer.

the quantity of information which can be obtained for a relatively low weight and size.

4.3.1 Sensors for localisation

UAVs can use sensors to calculate their location and attitude. Mejias et al.[112] use stereo vision for calculating UAV altitude and motion. Tournier et al.[118] have created a vision based UAV positioning system using Moiré interference patterns created from pairs of gratings at a known separation. These produce different patterns depending on the position of the viewer, allowing the UAV to tell its position from analysing the grid. This works, but requires a Moiré pattern in a known location to be in view at all times to give the UAV its position.

Several groups have created SLAM systems for UAVs, with a variety of sensor options — with onboard laser range-finders;[119] with a Microsoft Kinect 3D camera and IMU;[104] or with basic low resolution visual, using image recognition to build up an aerial map of the environment.[120]

Sensors can also be used to direct route planning. Xu and Dudek[121] use a visual camera to follow outlines and boundaries, tracing shorelines and roads to plan their trajectory.

4.3.2 Sensors for task completion

Depending on the use case for the UAV, various sensors can be used to complete the tasks required. If the UAV is being used to survey an area after a nuclear accident, say, the UAV would necessarily carry a radiation counter. The projects mentioned in this section are limited to those based around search related tasks.

He et al.[122] describe an entry in the MAV'08 competition, where UAVs have to scout out an area in a hostage situation to ensure a safe path for following ground vehicles. This UAV was monitored by a human, who recognised and noted mobile objects of interest (enemy combatants, vehicles, etc.) and stationary ones (land mines, obstacles). The UAV then used a machine learning algorithm to enable it to recognise the objects in future and to track their position, allowing it to provide real time positions of enemy agents to other units.

This requires a human to constantly monitor the UAV's output. Symington et al.[111] describe a feature based target detection method which for each image from the UAV camera selects points which are likely to be robust against image changes (from lighting, scale, rotation, perspective, etc.) A set of target features are manually selected in a training phase (e.g. pictures of a person lying on the ground), and these are compared against the new images. This can have problems with generalisation, where it's capable of recognising (for example) a standing man, but not a man crouching beside a wall, even if they're in full view.[34]

Human body detection can be improved upon by using 'part based models', which decompose recognition into discrete sections of body so that partially occluded bodies and those lying in strange positions can still be detected. Andriluka et al.[123] compare several different methods for detecting human casualties from the video stream of an indoor UAV.

Morse et al.[124] describe a 'seeability' metric for images taken by search and rescue UAVs, to describe the quality of the data obtained for each location. This can then be used to discern how accurate the detection will be based on that data.

4.3.3 Sensor Fusion

A missing person could be recognised in a number of ways. They could be giving off more heat than their surroundings; they could have activated an emergency radio beacon; they could be recognised just from their shape; they could be making a noise (either shouting or blowing three blasts on a whistle); they could be giving off CO_2; they could be moving. All of these can be detected by UAV sensors, with some degree of accuracy — a visual camera can detect the shape of a person; an infrared camera can detect body heat; a microphone can detect sound; etc.

There are two difficulties with this. The first is that these sensors can give both false positives and false negatives.[125] A person might be lying in an unusual position or be partially covered, so that a visual system is unable to detect them. Alternatively, a scarecrow might lead to a false positive for a visual camera — as might the heat from a sheep for an infrared camera.

The second difficulty is that the person is, in the majority of cases, only likely

to be doing a small subset of these things. A system which can only find missing people who happen to be carrying an emergency radio beacon will be of no use if the missing person in this case isn't.

If the helicopter were able to detect a range of the different characteristics of a missing person then it could use sensor fusion to decide how probable it is that this is a person — if it looks like a human, sounds like a human, and is hot like a human, it's considerably more likely that it is a human, and not a scarecrow, a radio, or a sheep.

Rudol and Doherty[113] present a system which uses sensor fusion for detecting humans, using a colour visual and an infrared camera. It first analyses the infrared image for shapes at human temperatures, and then confirms or rejects these using the data from the visual camera, with (in the described experiment) 11 bodies found, 3 false positives, and no false negatives. Merino et al.[64] describe a similar system for fire detection, which again uses visual and infrared cameras with sensor fusion.

The main problem with using sensor fusion on small UAVs is that sensors tend to make up a large proportion of a UAV's payload, which is usually highly limited. One possibility to circumvent this would be for different helicopters to carry different sensors and communicate what they see. If two or more helicopters with different sensors believed that the person was in a particular location then the probability would be far higher than from multiple sightings with the same sensor type.

4.4 Collaboration

If the ideal UAV is fully autonomous, able to take an abstract high level situational overview and work out tasks based on them, then the ideal collaborative UAV should be able to work well as part of a group. This would involve taking an abstracted list of orders or potential jobs that may be required (e.g. searching an area, tracking an object, etc.) and working as part of the team to complete them, with varying levels of inter-UAV communication. Each UAV should be able to decide upon the importance and relevance of the jobs independently of human input based on their observance of the rest of the team — whether that team is other homogeneous UAVs, heterogeneous UAVs, or humans. The UAV should be able to take the initiative,

performing tasks such as scouting along the path ahead when travelling with a group of humans without needing to be instructed. This, as with full autonomy, is still some way in the future; but there has been much research into more basic UAV collaboration.

Collaboration can be split into two distinct concepts: Coordinated planning and Cooperative planning.[37] These lead to two very different methods of collaboration — in the latter, all the data is disseminated between units and the units then individually decide what to do based on it.[16] In the former, a central intelligence tells all the units what to do (or, in some cases, all units debate on a course of action). Cooperative solutions are unlikely to produce as good results as coordinated ones, but do have the advantage that they do not require a central connection with all the risks connected with a point of failure. They also lend themselves far more to scaling from small swarms to larger ones.

Teacy et al.[126] describe some of the difficulties involved in collaboration, when the UAVs have no control over each other and only have a limited world view. They introduce a system of Bayesian Reinforcement Learning as a method for training UAVs to work together — selecting tasks that complement each other and improve the work of the group.

Collaboration isn't just about UAVs working with other UAVs — it also covers UAVs working with other robots (such as autonomous ground vehicles), or with humans;[89] although in many cases human collaboration is more similar to human control, for some level of abstraction (see Section 4.1.1).

While collaboration is possible without communication,[127] one important way in which UAVs can collaborate is in the sharing and relay of data — both between each other and back to base.[128] Teacy et al.[35] describe a system of distributed data fusion, where data is brought together across all UAVs, rather than to a central point. This reduces communication problems, removes the single point of failure, and makes the system modular; but it can lead to situations where the algorithms combining the data on the UAVs count the same data multiple times and so put undue weight behind it. The UAVs could collaborate to combine sensor data from

[16] Using a variety of algorithms — see Appendix E for information on Game Theory, which is one way of looking at collaboration problems

various locations, or to combine SLAM maps.[129] Alternatively, the UAVs could collaborate to provide robust relay communication back to a central base.[35]

The NATO standard STANAG 4586 relates to the inter-operability of UAVs, and allows for common protocols and message standards. It's designed to allow ground stations to control a large number of different UAVs, but could in theory be used for inter-UAV communication.

Collaborative UAVs can also achieve more physical joint activities. They have been used for joint lifting of large objects,[96] collaborative building of structures,[97] and providing each other with transport (in the case of a larger UAV mothership transporting a smaller, more manoeuvrable UAV scout).[103]

A more common means of collaboration is through the use of cooperative and distributed searching.[34; 36] UAV searching is covered more in Section 4.5.1.

Another way in which UAVs can work together is in the way that they fly as a group. There are two methods by which a swarm of UAVs can fly together, as opposed to simply flying individually in close proximity with obstacle avoidance. UAVs can fly in formation,[130] which can give advantages such as optimising ground coverage (or reducing the required amount of path planning, as in follow-my-leader[65]), or they can flock,[100; 131; 132] using a system such as the Boid algorithm.[133] Boid algorithms bear several similarities to cellular automata, in that both return a behaviour based on a set of simple rules. The three original Boid rules were Separation (avoidance of others nearby in the flock); Alignment (flying towards the average heading of the others nearby in the flock); and Cohesion (flying towards the centre of the others nearby in the flock). These meant that all units kept together, without collisions, and could move forward as a single connected entity. This is easily extended to allow for other rules.

Chapter 7 covers a different means of UAV collaboration — that of relative positioning, where UAVs calculate their positions relative to other units in the swarm.

4.5 UAVs in emergency situations

There is much ongoing research into the use of technology in general for disaster management,[134] and into UAVs in particular. UAVs can be especially useful in

disasters where there is a limited number of human rescuers available, there is a high level of danger for rescuers, and speed is of the essence. Examples of this include situations such as collapsed mines, industrial accidents, and missing persons (see Section 4.5.1), as well as the aftermath of natural disasters — hurricanes, earthquakes, tsunamis, etc.[57]

In 2005 a small number of UAVs from the Safety, Security and Rescue Research Center (SSR-RC) were used in the aftermath of Hurricane Katrina to aid in the search for trapped survivors.[52] The US Air Force also requested use of UAVs for search and rescue, but due to national airspace restrictions they were unable to obtain authorisation to do so. As a result of this, on May 18th 2006 the FAA issued a certificate to say that UAVs could be used for disaster recovery and search and rescue in emergency situations.[135]

The Orchid project (see Section 4.2.10) is researching into different uses of technology for disaster management, including collaborative UAVs which work in groups and with humans.

UAVs can aid rescuers in many different ways, but the major use is currently in providing aerial imagery of the situation. One example of this is the swarm of UAVs created by Delle Fave et al., which fly over different parts of a disaster area on the request of humans on the ground, deciding where to film based on the importance of the different areas requested.[88] After the earthquake and tsunami in Japan in 2011 military UAVs were used to fly over the damaged nuclear reactors at Fukushima to ascertain their status, due to the high radiation levels.[136]

Other uses of UAVs include providing more reliable signal coverage, for various purposes. Varakliotis et al.[67] have developed a swarm of UAVs which autonomously cover an area to provide network coverage, forwarding signals between themselves back to a remote base. These are designed to be used in cases where there is a sudden increase in traffic in a network, such as in disaster situations where many people are simultaneously trying to use the mobile phone network and emergency services bands.

Another way in which UAVs can be used in disaster situations is for search and rescue — finding missing or injured persons and then guiding rescuers to them[109]

— although this can be considered as being more of a stand-alone task.

4.5.1 UAVs for Search and Rescue

While some large UAVs are capable of carrying human passengers, for the most part UAVs are limited to the search element of Search and Rescue. UAVs are useful for search and rescue for a number of reasons,[52] and in some cases smaller UAVs are better than larger ones.[137]

Search and Rescue can be split into a large number of categories (see Chapter 2). Of the land-based situations, the two main ones are Urban and Wilderness — the former tends to involve searching for missing people inside buildings, usually after some form of accident, and the latter searching for missing people in the countryside (mountains, woods, deserts, etc.)[17]

There are numerous groups who have researched into these, and there have been several robotic search competitions. RoboCup Rescue[138] is a competition primarily for ground based robots with varying levels of autonomy which have to cover a series of increasingly difficult challenges based on urban search and rescue. These entail searching large areas with constricted access, uneven floors, multiple levels, and other difficulties, searching for missing persons. Missing persons are dummies which display one or more signs of life, comprising of the human shape of the dummy, a light source (representing a torch), a heat source (representing body heat), a looped sound recording (representing calls for help or scrabbling at rubble), a CO_2 canister (representing breathing), etc. These can be placed in fairly inaccessible areas, such as behind holes high on the wall. The robots need to use sensor fusion to recognise the missing persons, as well as being able to cover the terrain to reach them and to employ a search strategy to locate them. Although the robots tend to be ground based there is some scope for use of UAVs, and a specifically UAV based variant of the competition has been discussed.[139]

Use of UAVs for urban search and rescue is limited, due to the inherent difficulties in the situation — the UAV is unable to use GPS, has many obstacles to traverse, and is unable to make the assumption that the missing person is likely to be below it.

Although the algorithms described in this thesis could be used for many different search environments, they focus particularly on wilderness search and rescue. Non- and partially-autonomous UAVs are already being used in some areas for SAR situations in open countryside to give an aerial view when full scale human piloted aircraft are unavailable,[17] and — in a similar vein to RoboCup Rescue — there are competitions for wilderness searching, such as the annual UAV Outback Challenge competition in Australia.[140] For this, teams have to locate a specific person ('Outback Joe', a dummy with a known appearance[17]) lying on the ground within a set area (just over 1 square mile), report its position to within a certain margin of error, and drop a package to land beside it.

When discussing UAV use in a wilderness search and rescue operation with the staff of the UK Aeronautical Rescue Coordination Centre (ARCC) at RAF Kinloss (which oversees all search and rescue missions in the UK) they noted an interesting difficulty in the integration of UAVs into current search and rescue methods. This was the initial location of the UAVs, and their endurance. A missing person search could occur anywhere within the United Kingdom. At the moment, a pilot can fly from the nearest air base to offer support to volunteer search and rescue groups on the ground. Manned helicopters have several hours of flight time, a range of a couple of hundred miles, and a comparatively high speed; if currently available UAVs were to be launched from the same base they would be unable to keep up, and would take considerably longer to reach the situation (although they may have a longer overall flight time). The two alternatives to this are launching the UAVs from a manned aircraft, and having them based closer to the search area. For the former, this would give problems in terms of storage space, the UAVs hitting the slipstream as they emerged, and re-collecting them when they had finished. For the latter, in order to ensure that a UAV swarm was closely available to any search and rescue situation would require an extremely large number of them to be kept ready across the country, most of which would be little used. However, this is still probably the best solution — if the cost of UAVs was sufficiently reduced then local volunteer

[17] Teams are given the following description: 'The target for the search will be a human shaped dummy wearing a high visibility shirt, jeans and an Akubra hat. The dummy will not be moving and will be positioned in a typical resting pose for a tired and lost bush walker visible from the air. A simulated heat signature will be located next to the dummy.'

rescuers would be able to buy them, especially in areas which get a high rate of missing person searches.

There are many ways in which an area can be searched for a missing person (see Chapter 2 for more details). The simplest is if the person happens to be carrying a beacon, such as a Personal Locator Beacon (see Appendix A). Hoffmann et al.[137] have developed a system for collaborative UAVs which search for avalanche victims, where the UAVs have a bearing to the missing person's transmitter, but no further information (such as a range estimate). This works using a monte-carlo simulation, where the particles are weighted based on the received direction of the beacon. Their trajectories aim to jointly reduce the area in which the missing person may exist. Similar work has also been done using UAVs to locate animals fitted with short range radio trackers.[141]

In many cases, however, there is no 'active' information about the missing person's location, such as a beacon. In these situations it is necessary to create probability maps or occupancy grids[25; 23; 38] (see Chapter 2) which are then searched across; although often the method for traversing the probability map is limited to an algorithm based on the expected return for the next few steps, giving a high bias to local maxima (see Chapter 3 for an improvement to this).

Chapter 5

GPS tracking

5.1 Introduction

If a UAV is to perform a useful task, it is likely to need to know where it is in relation to the world around it. It was therefore important to create a system which would allow the UAV to know where it is.

GNSS is the best method for global positioning in many cases — although there are disadvantages associated with it (see Chapter 6 about altitude errors, and Chapter 7 about relative positioning in situations where global positioning is unavailable such as indoors). More on GPS can be found in Appendix F, and a description of some alternative wide area positioning systems is given below.

The School of Engineering and Computing Sciences at the University of Durham has been developing an extensible GPS tracker system for many years. This was originally developed for tracking shipments in transit; intermittently calculating positions (and measuring inputs from a range of sensors) and uploading them to a website. As this would be unsuitable for a UAV — which requires very frequent positions — I've extensively modified the Durham Tracker to allow for this new use case, as well as making a number of improvements over the original system. Although the project had been running for a number of years it had a number of flaws which prevented it from being fully functional, and the improvements made for this PhD not only fixed these but also made the trackers into a configurable platform for conducting localisation experiments both for this PhD and for any

future researchers.

Modifying a tracker for this purpose gives an added advantage over simply connecting a GPS receiver to the UAV control circuitry. As well as the UAV knowing its position for flight control, the tracker has the capability for independently reporting back information about the UAV (both positions and any further input data) to a remote server. This would allow a user to watch what the UAV was doing remotely.

While being used as a standalone device (i.e. not connected to a UAV), the Durham Tracker could also be used in search and rescue situations — allowing a control centre to be able to see exactly where had been searched by the various team members, without their having to record their routes manually.

Section 5.3.1 gives a history of the Durham Tracker Project, and Section 5.4 describes the modifications made as part of this PhD. This is compared against other, commercially available tracking systems in Section 5.5 — both in terms of specifications and experimentally.

5.2 Background

To find a position it must first be specified what that position is being given relative to, and this requires external data to be received. It is not possible to gain a position without the use of external data, whether this is through observations of surroundings or through receiving radio communications.[1]

Radio navigation systems mainly started development almost a century ago, spurred on by the war effort. These started with simple beacons to guide to a point, but soon developed into systems which were able to give with some degree of accuracy an exact position relative to a base. Many of these worked on a hyperbolic navigation system. The principle behind this starts with three transmitters — one master and two slaves. The receiver is able to tell how much closer it is to the master than each of the slaves using a variety of possible methods, for example by timing the delay between synchronised radio pulses. For both master-slave combinations the receiver can then describe a hyperbole. Where the two hyperbola intersect will show the current position. This gives a good accuracy, increasing as the positions of

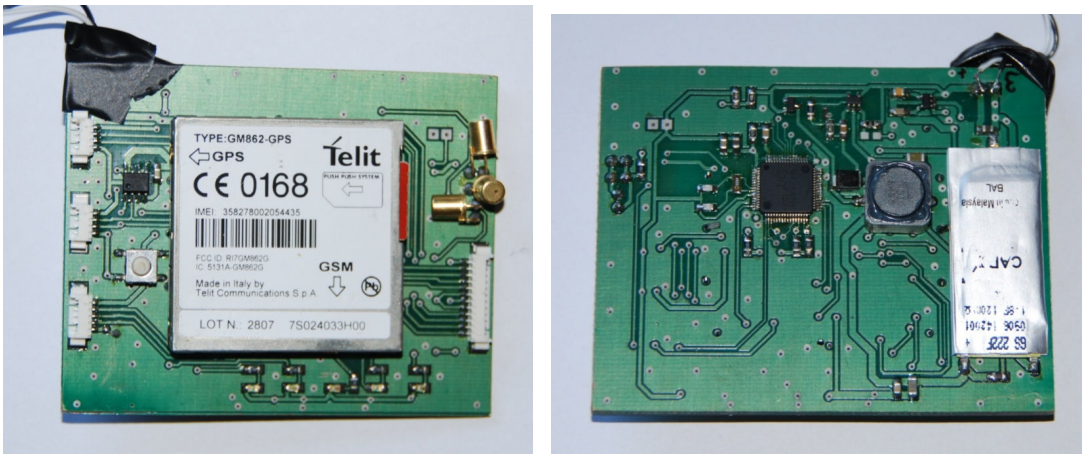
[1]The one exception to this is use of an IMU to gain a relative position to a starting point

the transmitters approach orthogonal hyperbola but decreasing over distance. The stations must be in fixed, preknown positions, and a position can only be taken from a set of three transmitters working together — this means that all three must be in range of the receiver, forcing them to be relatively close to one another unless the wavelength is long (which reduces the accuracy). The first system to use this method was the GEE system; the best known is probably the LORAN system which is still in use today. Omega — the first worldwide, continuously available radionavigation system, now decommissioned — worked in a similar way, with the exception that phase difference was used instead of time difference, giving positions only accurate to within a 'lane' due to the phase repetition.

GPS works in a slightly different way. It is also a time of flight system but does not require synchronised transmissions. Instead, signals from the constellation of satellites each contain a timestamp to show when they were transmitted. Assuming synchronised clocks these timestamps give a time of flight; time of flight from three or more satellites gives a position. The clocks are synchronised on the satellites as they are all fitted with highly accurate atomic clocks; the GPS receiver obtains a synchronised clock by using more satellites than required and working out the time from the extra information. More information on GPS is in Appendix F.

The majority of wide area localisation schemes work on a similar method; either measuring the phase or time difference between a number of transmitters to produce a series of surfaces which overlap at the location of the user.

Other wide area navigation methods include ones which map the signal strengths from a variety of different radio transmitters making up the background noise in an area. Examples of these range from the Cursor system,[142] which works by comparing the phase difference between signals from multiple background transmitters at fixed base stations against measurements taken by a roving unit to calculate a position, to the more advanced NAVSOP[143] system proposed by BAE Systems. NAVSOP (Navigation Via Signals of Opportunity) opportunistically uses background transmissions from a wide range of transmitters with a variety of signal types (GSM, DAB, DVB, 3G, MW, GSM, etc.), which need not have any synchronisation. The various transmissions have different characteristics, which means that



(a) Top (b) Underside

Figure 5.1: A v.3.1 Durham Tracker, showing the tracker when taken over for this PhD

Figure 5.2: A v.3.2 Durham Tracker, showing the current state of the project

the system uses them to calculate positions or movement in different ways.

Similar to this method are WiFi Positioning Networks (see Section 7.1.2.1) which map the signal strengths of 802.11 wireless transmitters, such as the systems offered by Skyhook Wireless, Navizon, Ekahau and Google. These fingerprint individual transmitters to build up a map.

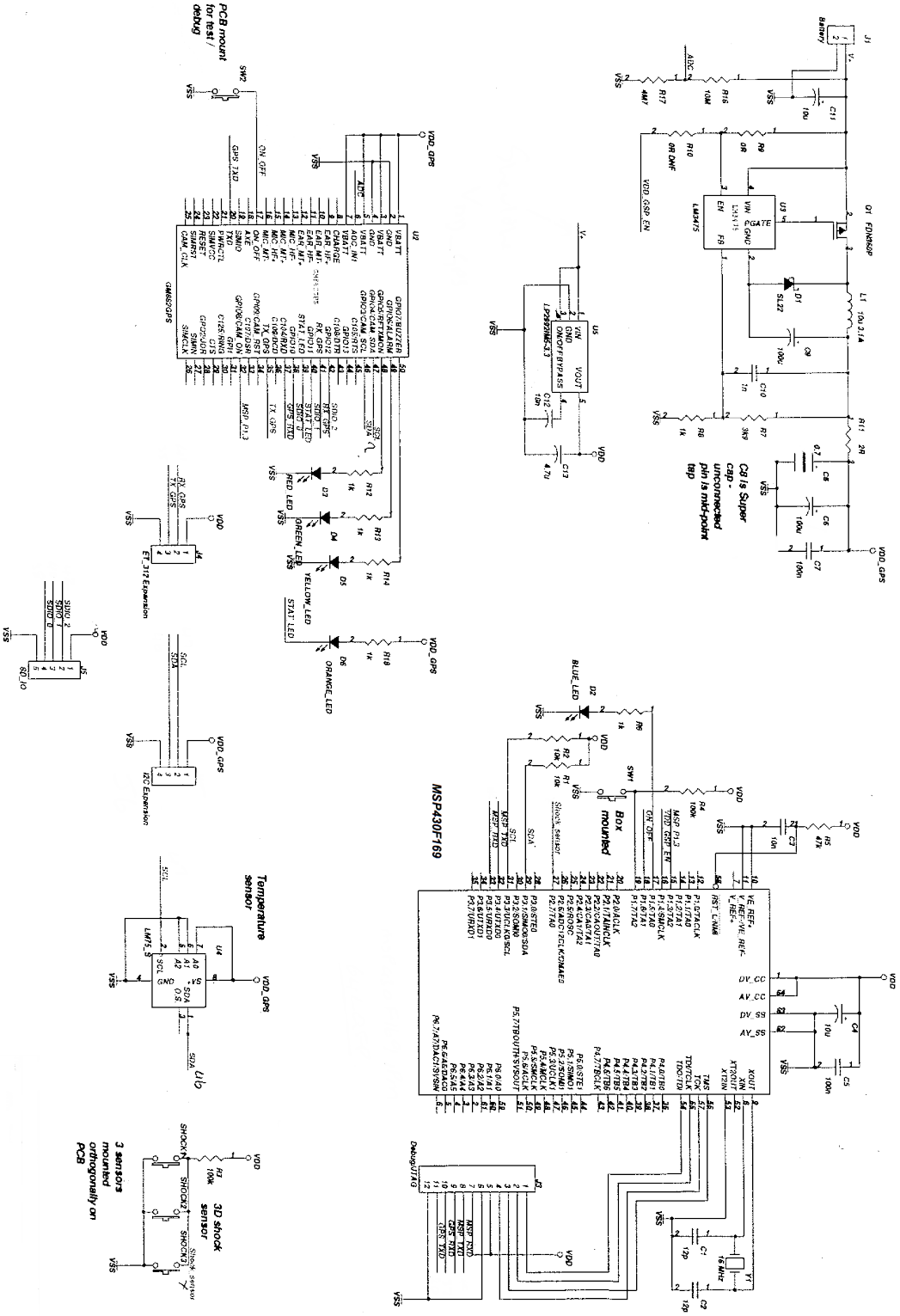


Figure 5.3: A schematic of the tracker circuit layout, showing the hardware connections.

5.3 Durham Tracker Project

5.3.1 Overview

The Durham Tracker project has been running for a number of years. The initial intention for the project was to produce a tracker for crates of produce, reporting intermittently their positions and giving additional information such as their temperature and whether they had received any undue vibrations. This would allow the user to see where their shipment was and how it was being treated. The tracker was later extended over numerous iterations to allow for a much wider set of use cases.

Unless otherwise specified the versions discussed in this paper are v.3.1 and 3.2. When the project was taken over for this PhD v.3.1 was current, but had a number of problems, most notably that the position of the shock sensors partially blocked the SIM card slot. Version 3.2 had been developed, but did not work — there was an unknown fault[144] which was discovered soon after to have been caused by a split ground plane. Connecting the two halves fixed the problem.

The tracker has been used with a number of experiments in the Engineering Department at Durham, some of which are documented in Section 5.3.2.

A hardware diagram for the tracker is shown in Fig 5.3. The main processing components of the tracker — a Telit GM862-GPS and an MSP430F169 — are described below, in sections 5.3.1.1 and 5.3.1.2 respectively.

The tracker runs sensors off an I^2C bus (see Section 5.4.2.8 for more details). A socket allows for external sensors to be connected to the tracker, giving end users an opportunity to add their own inputs. An example of this is given in Section 5.3.2.7, where readings of audio levels were taken around Durham. The sensors on the board are an LM75 temperature gauge and three 10G shock sensors mounted orthogonally to give a three axis shock sensor.

The LM75 Digital Temperature Sensor and Thermal Watchdog with Two-Wire Interface[145] is a temperature gauge designed to connect to an I^2C bus. It is rated to give temperatures from -55 to +125 °C , and is accurate to within 3 degrees (2 degrees from -25 to +100). It comes as a surface mount package, with eight connections — power, ground, overtemperature shutdown, two wire I^2C (clock, data), and three

pins for setting the three least significant bits of a 7 bit I^2C slave address (the most significant bits are set internally to 1001).

The three ASLS-10 shock sensors from ASSEMtech are 10G acceleration switches, measuring 4.6mm (diameter) by 6.8mm (length).[146] They are connected to the MSP (see section 5.3.1.2), to give them an I^2C interface.

The tracker is powered by four AA batteries, in a separate box, initially hard wired to the main PCB but later connected via a barrel connector. The reason behind this choice was to give the user an easy means of recharging the tracker when on the road.[146] The initial use case being a tracker for crated goods, the delivery driver would be more easily able to buy AAs from motorway services than to find a means to recharge an internal battery. However, this decision is inconvenient when either carrying the tracker around in person or when using the tracker on a weight limited UAV. It also means that the user gets through a large number of batteries, unless rechargeable AAs are used. Changing to using an internal rechargeable battery is discussed in Section 5.4.2.10. An LM3475 SOT23 hysteretic buck controller is used to regulate the voltage.

As the batteries are unable to provide the high currents spikes required by the Telit when transmitting over GSM,[2] a 1.8F 120mΩ supercapacitor[3] is used as buffer.[144; 146]

In order to connect the tracker to a computer, a breakout board is used. This is described more fully in section 5.3.1.3.

Before the modifications were made as described in Section 5.4, the tracker would work as follows. Once switched on it would run through a short initiation procedure, after which it would enter the main loop. This would consist of taking a GPS reading, taking a series of sensor readings, uploading these to a remote server, and then putting the tracker into a low powered sleep mode for a preset period of time, before repeating. If a connection to the server could not be made, the details would be stored in a file. If, on a future occasion, a reading uploaded successfully then it would attempt to upload up to ten readings from this file; a process known as

[2]Due to their internal resistance, there is an internal voltage drop when current is drawn from alkaline batteries.

[3]Electrochemical Double Layer Capacitor (EDLC)

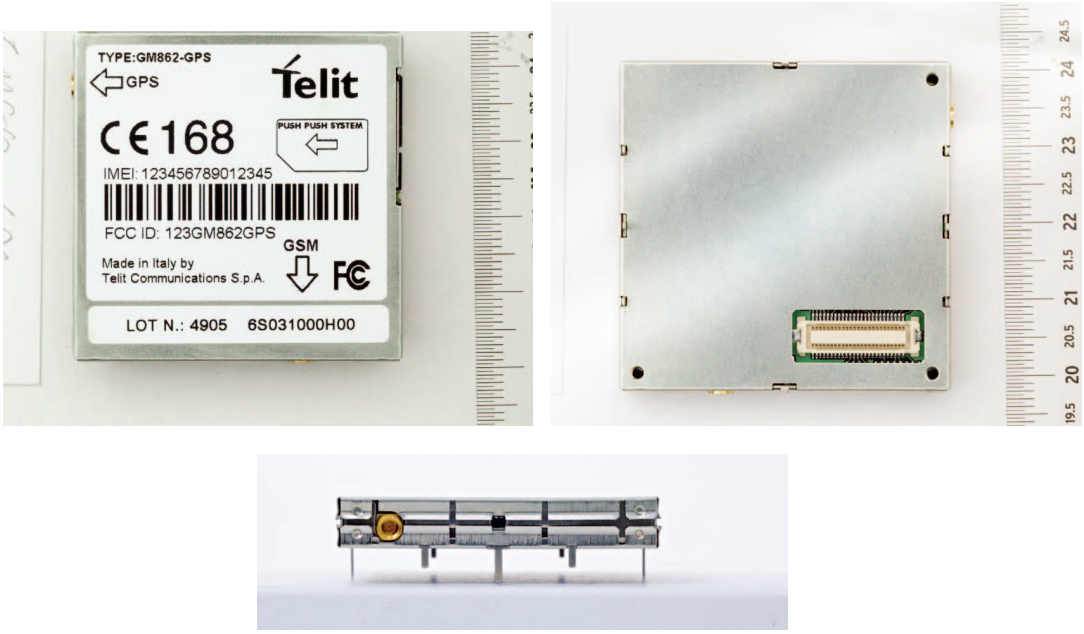


Figure 5.4: The Telit GM862-GPS

'backfilling'. When uploading to the server, the tracker could be given in reply an instruction to change between a series of configuration files; setting, for example, the frequency of readings.

The server (a single core virtual machine running Debian) receives the data from the tracker and appends it to a MySQL database. Users can then see the current and historical positions of the trackers by looking at a website, also hosted on the server, that displays the positions and other data on a map (see Section 5.4.2.9 for more details).

5.3.1.1 Telit GM862-GPS Module

The Telit module[147] (hereafter, 'Telit') is the main part of the Durham Tracker, as it supports the three main elements — a GPS receiver, the ability to make GSM/GPRS connections, and the processor which runs the code. It measures 43.9 x 43.9 x 6.9 mm, weighs 20g, and has a low power consumption.

The inbuilt SiRFstarIII single-chip GPS receiver has an advertised position resolution accuracy of less than 2.5m, with a high sensitivity for indoor fixes (up to

Power off	<26 μA
Idle (registered, power saving)	2.6 mA
Dedicated mode	200 mA
GPRS cl.10	370 mA

Table 5.1: GSM power consumption (typical values). All data taken from the GM862-GPS datasheet.

-159dBm, with an active antenna). It has a hot start of less than 3s, with warm and cold starts of less than 35s. The GPS has an operating current of 75mA.

The quad-band GSM/GPRS modem allows the module to connect to the internet via the mobile phone network. There is an integrated SIM card holder on the module itself. The GSM modem's power consumption is given in Table 5.1.

The internal processor runs scripts written in Python (Python 1.5.2; this is now a somewhat outdated version). It also is capable of taking commands given directly via serial, using an extended version of the Hayes command set ('AT' commands, based on GSM 07.05 and 07.07, with Telit-specific enhancements).

The Telit module comes with 1.9MB of non-volatile memory set aside for user scripts, and 1.2MB of RAM for the Python engine. It also theoretically supports software updates sent over the air, although this has not been tested.

The 50 pin Molex connector (visible in Figure 5.4b) includes up to 13 I/O ports, and the Telit module supports both I^2C and SPI as well as UART. It is capable of sending and receiving SMS text messages and of making and receiving telephone calls.

The Telit includes a built in Watchdog timer, which will reboot the module after a period of 10 minutes if it is not regularly reset (or temporarily disabled). This is used to prevent the system from crashing, by restarting it if it takes too long to complete a given activity. It is set to be active at all times except when sleeping (using the built in low power mode of the Telit) and when uploading points to the server.

5.3.1.2 Texas Instruments MSP430

The MSP430F169[148] (hereafter, 'MSP') is part of the MSP430 family of ultralow power microcontrollers. The device's low cost 16 bit RISC CPU, designed for embedded applications, has 60KB flash memory (plus 256B information memory) and 2KB RAM. It runs scripts written in C, and can be programmed via a JTAG interface.

The MSP is used for a number of applications on the tracker. The primary use is for the control of the three orthogonal shock sensors connected to its GPIO ports; it logs the time of each reported shock and returns the results when queried by the Telit over the I^2C bus.

It is also used for initialising the Telit. On startup it holds the ON_OFF pin of the Telit low for over 1 second to switch it on. It has additional functions for implementing a box mounted reset button, which could also be used for things such as immediate transmitting of position, but this has not been included in the current box design and is therefore unattached.

One great advantage of having the MSP in the system is that it can be interrupt driven, and can respond to interrupts without disrupting the running of the Telit. Things like the shock sensors require an immediate response (to get an accurate timestamp), and can be dealt with by the MSP and forwarded to the Telit at a later stage.

5.3.1.3 Breakout board

For connecting the trackers to a computer for programming and offloading data a breakout board was used. This was created in house, and connected onto the main PCB of the tracker to give three serial ports. The three ports (see Figure 5.5) gave connection to the raw NMEA data from the GPS, to the MSP for reflashing code, and to the Telit. This third gave a live stream of the tracker status while the code ran, the ability to upload / download files (mainly for uploading new code and downloading logs), and while code was not running a chance to enter commands directly.

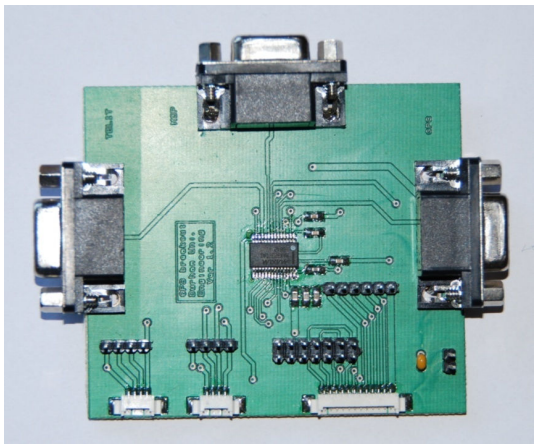


Figure 5.5: The breakout board for interfacing between the Durham Tracker and a computer

5.3.2 Previous work

The Durham Tracker Project has been running for a number of years, and there have been numerous smaller projects based around it. A selection of these are summarised below.

5.3.2.1 Joe Milbourn and David Jones

Joe and David initiated the Durham Tracker Project, designed the original incarnation of the tracker, and were heavily involved in its development until leaving the department (and the project) less a year before this segment of PhD research started. Much of the previous work on server scripts and tracker code was done by Joe, and most of the previous work on the website front end was done by David.

5.3.2.2 GPS / GPRS Tracking System — Ming[149]

This was a very early point in the tracker's development, when it consisted of little more than a Telit module attached to a development board (an EVK2 Telit Evaluation Kit). At the time it was actually using a different version of the Telit module; the GE863-GPS. Ming made two major changes to the project — the first was to connect a temperature sensor to the ADC, the data of which were uploaded to the website when it uploaded positions. The second was to implement shock

sensors. These consisted of three sensors, rather larger than those used now, and when shocked these would switch on the Telit to send a position. In order to do this a 555 timer circuit and latch were connected between the shock sensors and the Telit's ON/OFF pin (fulfilling one of the functions now performed by the MSP). After the reading had been sent the Telit could then switch off again.

5.3.2.3 A GPS Based Solution for Traffic Monitoring and Congestion Reduction — Richardson[150]

This project, in collaboration with Durham County Council, investigated traffic congestion under the 'Transport Innovation Fund' — a system of Government funding for local authorities to investigate traffic congestion and introduce trial schemes for reducing it. During the period of this project a lot of work was done on the tracker, moving it to a dedicated server and implementing the backend database. The main element of the project was to install a tracker running a cut-down version of the code (able to take readings and upload them as often as every 20s) into a car, and then to drive a series of routes through Durham at different times of day and under different traffic conditions (including at peak times and at hours when the roads were practically empty). The positions of the data points could then be used to calculate a speed, and the speeds plotted to show where the traffic flow was slowed. This allowed for traffic blackspots to be identified. Further work was also done to allow for geofencing, with several potential applications tested — including congestion zones, automated parking, and toll roads and bridges.

5.3.2.4 GPS Tracker with Temperature Sensor — Wen[151]

The main addition made by this project was to include a temperature sensor into the tracker. Three DS600 temperature sensors were attached, connecting to the Telit's ADC, and reporting to the website. These did not interface with the I^2C bus.

5.3.2.5 Remote GPS Tracking for the General Market — Watkins[146]

This report describes the design of the first custom designed in house tracker PCB, replacing the previously used SparkFun GM862 Evaluation Board. This change to

a custom design entailed some changes and augmentations, including the option for adding in a second GPS receiver which would interface with the Telit, removing the use of the internal receiver. By this point the I^2C bus had been implemented; he designed a complicated new arrangement of smaller shock sensors with a latched output designed to connect to this, although the final implementation instead connected the three sensors in parallel to the newly added MSP. Watkins also created an early box design. One of the most notable changes made was to move from using a rechargeable battery to using alkaline AA batteries. This necessitated a large number of additions, including voltage regulation and a supercapacitor — the change was made because the rechargeable battery packs needed recharging periodically, and for the current use case new AA batteries would be easier to come by than mains electricity.

5.3.2.6 Measurement of GPS Tracker's Sensitivity and Development — Zhou[152]

Zhou experimented with different GPS antennae for comparison, and ran some tests on the temperature and shock sensors to ascertain their responses to various situations.

5.3.2.7 The Integration of Environmental Audio Noise Sampling onto GPS/GPRS Trackers for Noise Mapping — King[153]

This project was based around using the Durham Trackers for noise mapping. Noise mapping is creating a map of the sound levels in a given area, and there are several potential uses of this. The two described were music festivals, where the sound levels vary across the venue but sound checks to compensate for this are currently only performed before people arrive (which changes the acoustics), and noise pollution, where a local government may wish to quantify the noise from a city, road, factory, etc. The paper focuses on the latter situation. Currently the noise maps used by Defra are created mathematically, based on sophisticated models of noise sources such as local traffic data, but using no actual noise measurements. The Durham Trackers were augmented by an additional sensor — a condenser microphone with

preamp and an LTC1968 RMS-to-DC converter, which made up the Environmental Audio Noise Level (EANL) circuit. The sound levels recorded were uploaded to the website, and plotted on the map using a system of colour coded circles. This was then calibrated around Durham, using a commercial, A-weighted[4] decibel meter.

5.3.2.8 GPS Tracking Device — Hancock[144]

Hancock was the last person to be running the Durham Tracker Project before it was taken over for this PhD. The main scope of his project was to investigate the stability of the system, and if possible to solve any bugs discovered. The newest version of the tracker hardware (v.3.2) had just been developed, and this had several problems. Several hardware faults were diagnosed, and some software faults were noted with attempts made to solve them. The most notable of these was an intermittent fault that prevented the data from being upload to the server (see Section 5.4.2.1 for more). The method proposed for solving this was to repeatedly attempt to upload all data points until they succeeded; giving up and storing the data ready for backfilling after too many attempts. While this did reduce the impact of the situation it did not solve the underlying problem, and took a considerable period of time.

5.4 Modifications

5.4.1 Reasons

As mentioned above, there were several reasons for modifying the Durham Tracker project.

The first is clear. The trackers in the form that they were received were not fully functional. They had several bugs which stopped them from being used for their original intended purpose, and if they were to be used for research they would first need these problems fixing. Many were intermittent errors which the previous people who had been working on the trackers had been unable to pinpoint, preventing them from using it as a viable system.

The second reason for making modifications was in order to generalise the scope

[4] A standard weighting designed to copy the ranges which the human ear hears most clearly.

of the trackers. They were designed with a specific purpose — that of tracking shipments — and there were several aspects of the trackers that were less generic than they might have been. It was intended to now use them to create a platform on which to run localisation experiments; both for this PhD, and for future students working on similar projects.

The third reason is in many respects the most important. If autonomous UAVs are to be developed then it is necessary for them to have some form of localisation, and the Durham Trackers offered the possibility of not only a positioning system for the UAVs themselves but also a tracking system to allow for a remote user to follow them and potentially send them commands. However, in their current form this would be impossible — they were designed for infrequent position updates, and would need modification for this new use case.

Below are documented some of the changes made to the trackers, followed by a comparison between the modified Durham Tracker and some commercial alternatives.

5.4.2 Changes

5.4.2.1 Server communication errors

The first problem with the Durham Trackers that needed to be solved was that they were being unreliable. They would function as expected for much of the time, but occasionally would suddenly stop reporting position. The logs would report various problems to do with not being able to connect to the server. In many cases the connection would be opened, but no response would be returned from the server to the data being sent (there would be no HTTP 200 'OK' response).

The problem appeared to be that the connection to the server was being closed prematurely; either the GPRS connection from the Telit to Vodafone, or the socket from the Telit to the server. Hancock[144] had attempted to solve the problem in two ways — firstly by explicitly closing and reopening the GPRS connection every time that a point was being uploaded (regardless of whether or not it was open already), and second by repeatedly uploading points a large number of times until they finally got through. If they had still not been uploaded after half a dozen attempts they

would be saved to a file (see Section 5.4.2.2) to be tried again later. Unfortunately, although this method did work on many occasions, it caused several problems. The closing of the socket caused errors when the connection was closed already, but more importantly the repeated uploads (including reopening GPRS connections that had been open and only just closed, and waiting for each upload to time out) took a great deal of time. While uploading, the tracker was unable to take GPS readings. Repeated uploading also did not solve the underlying problem — it merely increased the probability of having a successful upload.

Two things were done to solve this problem. The first was to put a check in the code to see whether or not the GPRS connection was still open when the time came to upload any data (with the 'AT#GPRS?' command); if not, it would be opened, but if it was open already it would not attempt to open it again. Also removed were any points at which the GPRS connection was closed — if the connection was open but no data was being sent across it it would not cost any more money, and the overhead for opening the connection was high (see Table 5.2). The second was to cut down the number of things done by the program in the crucial period between the connection to the server being opened and the data being sent. This was the period in which the GPRS connection was most likely to be dropped by the network; once the data was in the process of being transferred there was a greatly reduced risk of it failing. The code contained several things that could be done before the socket was opened, and moving these to an earlier point in the code greatly reduced the number of cases where the connection to the server was lost.

5.4.2.2 Rapid fire positioning and bulk uploads

One difficulty with using the trackers for monitoring the flight of UAVs was that as it was designed for infrequent measurements it was unable to take rapid readings of its position. This was fine if tracking something slow-moving or moving along a known route (such as a lorry on a motorway), but far too imprecise if tracking a fast-moving UAV.

When recording a position the tracker was set up to take a series of readings (position and sensor), upload them to a server, wait for a server response, and if

successful enter a sleep state until it was time to start the cycle again. If the upload was unsuccessful it would keep trying until it got a response, or until it had tried a certain number of times, at which point it would store the reading in a file. The ten newest locations in this file would then be uploaded point by point on the next occasion when it had a successful upload. Each point was considered separately. Any points that were not uploaded were put back into the file.

This was very time consuming. The time taken to upload a single point was around 25 seconds (approx. three times longer than it had taken to read it in), and if the unit had been without a GPRS signal for a while a large number of points could easily have built up meaning that the tracker would spend many minutes just attempting (and in some cases, repeatedly attempting if the signal was poor) to upload old data points to the website one by one. In extreme cases this extended period spent uploading caused the system to crash. Hancock[144] describes a test where locations were set to upload at five minute intervals and intervals of 30 minutes were managed instead, partly due to the 'backfilling' (uploading of old data points).

The points were stored in a Marshal file. Python's Marshal module[154] allows Python objects to be saved and loaded into a binary file, in a format which is specified as being machine independent but not version independent. Details of the format are deliberately unspecified, and there is no guarantee of changes being backwards compatible. Objects are added to the file and retrieved one by one in reverse order.

While the tracker was uploading it was unable to take readings, so it would be effectively out of action. It was therefore changed so that instead of uploading each point individually, every reading was stored in a file. This file was then periodically uploaded all in one go, uploading all the points in bulk.

This did introduce a difficulty, in that the Marshal files being created could only have the data they contained read by the tracker and not by the remote server. This meant that the format of the file had to be changed to plain text; this also gave faster read/write times, although it did take a little more disk space than the Marshal format.

As most of the upload time was taken up by overheads (see Table 5.2, which shows a 4–5x increase in upload time for a ∼150x increase in number of points), to

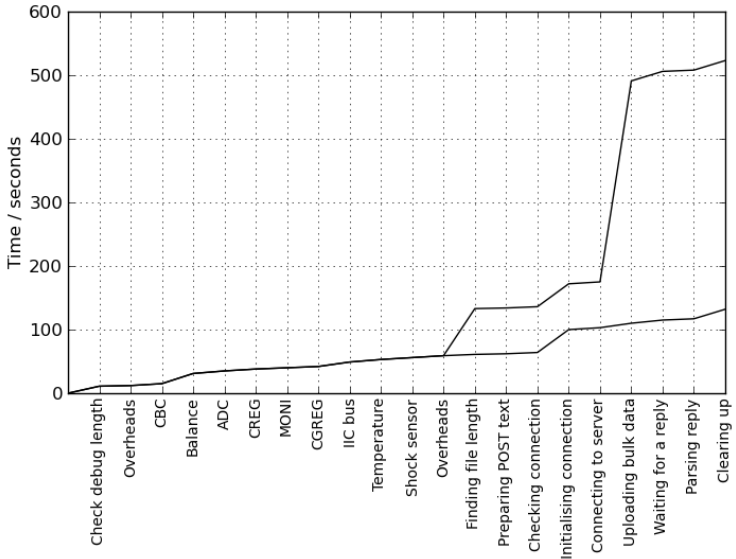


Figure 5.6: Graphical depiction of the breakdown of timings for uploading to the server, for both large (1465 readings) and small (10 readings) numbers of data points. See Table 5.2.

the extent that the upload time for a large number of positions was approximately the same as the upload time for one, uploading in bulk greatly reduced the overall time that the trackers spent unable to take readings. It also limited the times at which readings were unavailable to the infrequent uploads, meaning that for the majority of their use the trackers were able to provide readings at better than 10 second (and down to approx. 5 second) intervals.

The downside to this was that as the uploads were infrequent they would not appear on the server in real time; the cost for frequent readings was infrequent reports. This meant that while it was possible to precisely follow the route a tracker had gone, it was not possible to find out where the tracker was right now.

Nonetheless, this still gave a great improvement for this use case, and as the upload interval was now configurable it was possible to emulate the original behaviour by setting the upload interval to being the same as the reading interval.

Procedure	Time (small)	Time (large)
Check debug length	11	
Checking other data:		
Overheads	1	
CBC	3	
Balance	16	
ADC	4	
CREG	3	
MONI	2	
CGREG	2	
IIC bus	7	
Temperature	4	
Shock sensor	3	
Preparing data for bulk upload:		
Overheads	3	
Finding file length	2	74
Preparing POST text	1	
Checking connection	2	
Uploading data:		
Initialising connection	36	
Connecting to server	3	
Uploading bulk data	7	316
Waiting for a reply	5	15
Parsing reply	2	
Clearing up	15	
Total:	121	525

Table 5.2: Breakdown of timings for uploading to the server, for both large (1465 readings) and small (10 readings) numbers of data points. All values are in seconds. This gives an indication of how the timings change as the number of points increase. The tracker is unable to take any GPS readings in the period between the doubled lines (53 or 372 seconds), but could take them at intervals during the rest of the overheads. Some entries are not required every time, such as 'initialising connection'.

5.4.2.3 Multiple debug files

Another problem that the trackers showed was that they would work perfectly for a long length of time, but then would stop responding. When they returned from tests for analysis it was found that the logs showed nothing particularly notable. There was no noticeable correlation between the points in the code at which the failures occurred.

A large number of the debug logs were collected to see if there was a discernable pattern, but surprisingly the only pattern that could be found was that the log files were large and of a similar size. It seems that the module is unable to cope with file sizes above a certain limit, and so the logs were being written to until they were a maximum size before failing. The failure was noted by the module as being a write error, but as the debug logs were too large it was unable to record this.

The solution to this problem was to split the log files into several parts. The system would start by writing to the first log file. When the size of that file had reached a preset threshold it would start a second file, and so on. As there is also a limit to the total amount of storage available, a maximum number of log files were permitted. When that limit was reached the first file was deleted and recreated.[5]

This meant that the trackers had several hours of log files stored, without risking running out of space. The individual file size limit was set to approx. 1/24th of the size at which the trackers were crashing, with five log files, giving a very large margin of error.[6]

5.4.2.4 Hardware mounting problems

The Telit module used on the Durham Trackers is attached using a 50 pin Molex connector[7] (see Figure 5.4b). This gives a large number of pins in a small area, but the connector is very susceptible to becoming disconnected if any force is applied to the module. The Telit modules have eight metal legs on the outer housing to

[5]Note that the limited OS of the Telit module does not have many commands available in other OSes, including things such as 'logrotate', which would have performed this function.

[6]All limits can be modified, to allow for a longer period of log files and a reduced margin of error.

[7]Molex SlimStack CSTP 50 pin vertical SMD, part number 52991-0508 — hereafter referred to simply as a Molex connector.

anchor it to the board. However, due to a problem with the board design, the holes for these legs were unusable. In the case of version 3.1 they were in marginally the wrong place, meaning that additional force would be put on the Molex connector, causing damage. In the case of version 3.2, some holes were missing — only five holes were available, and it was not possible to add the other three as other elements would have collided with them. The holes that were available on 3.2 were also too small for the legs.

For bench tests, this wasn't a problem. The legs could be easily removed and the modules attached by the connector alone. The units could be tested and were shown to work, with a high level of reliability. They could be carried around by hand, or placed in a vehicle, and shown to work when travelling. Difficulties arose when they were tested in more difficult situations. When placed into a bag or a pocket and carried around the units would often stop working, seemingly for no discernible reason (no change in situation reported by the user, and nothing displaying in the error logs on the unit — it had simply stopped).

This was eventually diagnosed as being that the Telit modules were being shaken loose by the movement, and as the connection to the board became disconnected the system would stop responding.

The first response to this was to add a clamp. This clamp comprised of a bar which went over the Telit module and was bolted to the PCB on either side. This gave a downwards force to keep the module connected.

This helped to some extent, but did not fully solve the problem. The trackers were still occasionally stopping without any reported errors. In some cases the boards themselves would also stop working entirely, and replacing the Telit module did not fix them. The problem was eventually tracked down to the Molex connector again, which had become slightly damaged with small cracks appearing. It turned out that while the clamp had stopped the module from being dislodged vertically from the connector it had not prevented vibrations from shaking the module horizontally under the clamp. In particular, due to the position of the aerial connectors on the module, it had been given sufficient torsional force to damage the connector when submitted to vibrations from walking.

Figure 5.7: Tracker, without a box

This problem was finally solved by drilling slightly overlarge holes in as many of the correct places as could be fitted in and then affixing the Telit modules using hot melt adhesive on the underside of the legs. This fills the oversized holes, and anchors the module sufficiently that it is unable to shift under vibration, while still allowing them to be removable if the Telit module needed to be replaced. The reason for the holes to be larger than required is so that the module can be positioned by hand (with the Molex connector placed correctly) and then set into place, without any risk of the holes being marginally misaligned (the original problem with version 3.1).

5.4.2.5 Box design

Initially the GPS trackers were bare boards, with wires for GPS and GPRS aerials attached directly to the Telit module (see Figure 5.7). All status LEDs were mounted on the PCB. To make the units more user friendly a box was required.

One problematic element of the tracker design was the use of a large supercapacitor (1.8F, 120mΩ). The reason for this was to allow for a buffer in the power supply at times of high power use, such as when uploading data over GPRS. The difficulty came when powering up the unit if this capacitor was still partially charged. In this circumstance the system would not start correctly. It was necessary, therefore, to discharge the capacitor before powering on. This was initially achieved by touching across the terminals with a piece of metal. As this was not user friendly, and was hard to achieve when the board was mounted inside a sealed box, it was decided to have a button across the capacitor terminals which would perform the same task.

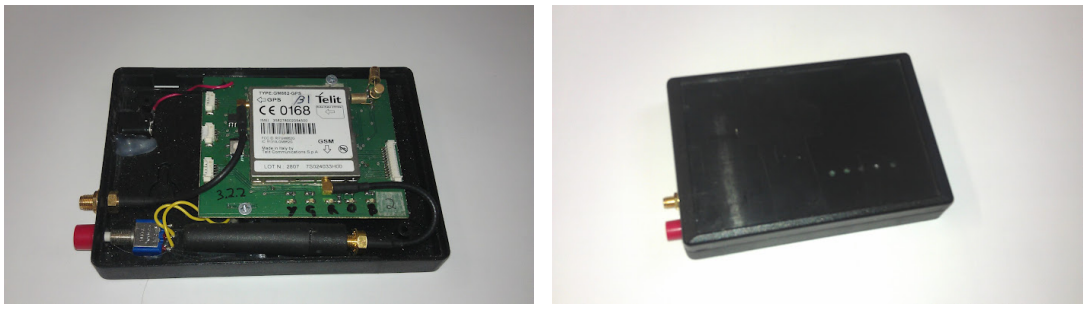


Figure 5.8: Tracker (v.3.2), in the 1st box design

Figure 5.9: Tracker (v.3.1), in the 2nd box design

The first iteration of box design (see Figure 5.8) had this button on the side, as well as a screw connector for the GPS aerial (the GPRS aerial was considerably smaller and internal to the box), a power socket (power connections had previously been hard wired to battery boxes), and holes which were aligned with the LEDs on the PCB.

The major difficulty with this design was the button for shorting the supercapacitor. This capacitor was protected by a small 2Ω resistor, which would blow if the capacitor was shorted while the main power was connected. This meant that the tracker had a large and prominent button on the side which, if pressed while turned on, would damage the PCB.

The second and current iteration of box design (see Figure 5.9) was similar, but had a small and recessed button. The LEDs were also taken off the PCB and instead mounted to the box.

5.4.2.6 LEDs

The Durham tracker has five status LEDs. One is connected to the MSP, one is tied to the Telit's status output, and the remainder can be set in the code on the Telit. Hancock[144] lists the following as descriptions of the five LEDs:[8]

Red	Constant on	Tracker is active
Green	Constant on	Data sent successfully; backfilling is occurring
Yellow	Constant on	GPS fix obtained
Orange	Short on, long off	Tracker has a valid GSM connection
	Long on, short off	Tracker is sleeping
	Short on, short off	Tracker is searching for a GSM network
Blue	Flashing	Tracker is powered on; shocks have been detected

The orange LED is connected to the STAT_LED pin of the Telit module. This is specified[147] as follows:

Orange	Constant off	Device off
	Short on, short off	Net search / Not registered / turning off
	Short on, long off	Registered full service
	Constant on	A call is active

The MSP's LED is set in code, as follows:

Blue	4 flashes	Shock detected
	16 flashes, then constant on (1.18s), then 4 flashes	Telit being initialised

[8]The meanings of the various LEDs has changed slightly in the past; other papers listed in Section 5.3.2 contain other LED descriptions. These were the meanings of the LEDs when I took over, however.

Some modifications were made to the use of the LEDs that were set in the code (red, green, yellow), so that they now fulfil multiple roles. They now work as follows:

<table>
<tr><td>Red</td><td>Constant on</td><td>Taking a GPS reading</td></tr>
<tr><td>Green</td><td>Constant on</td><td>Uploading to the server</td></tr>
<tr><td>Yellow</td><td>Constant on</td><td>Tracker is running low on power (approx. 6 hours remaining)</td></tr>
<tr><td colspan="2" rowspan="2">Red, Yellow and Green on together</td><td>When first switched on:
Tracker is initialising. Remains on until the first fix is obtained or times out.</td></tr>
<tr><td>Otherwise:
Tracker is extremely low on power (approx. 3 hours remaining)</td></tr>
</table>

5.4.2.7 Remote control via SMS

The Telit module contains many functions not directly connected to its use as a GPS tracker. It is able to make and receive phone calls, for example, and to send and receive SMS text messages.

The trackers previously had an option for sending commands to them via the website. When they uploaded data they would wait to be sent back instructions. This meant that the user could instruct a tracker to use one of a set of pre-loaded configuration files — but the tracker would only change to the new file when it next connected to the server, which might be a long time afterwards. Even if the tracker was recording positions very frequently, it might only be uploading them to the server once every few hours. This lead to a more severe difficulty — that if the user wanted to find out where the tracker was at the moment, they had no way of doing so.

In order to solve this, a piece of code was added that would check for incoming SMS text messages. If a message was received that contained a correct passcode and a command then it would act on the command. The only command that has been implemented in this way is an instruction for the tracker to immediately upload its

current position (and all previously recorded data points) to the server, to allow the user to find out where it is. Further commands could easily be added via the same method, for example to remotely set configuration options.

5.4.2.8 I²C problems

The Durham tracker comes with a number of built in sensors, including temperature and a three axis shock sensor. It is also extensible, allowing users to add on a variety of other sensors or inputs over an I²C bus. Once connected, the tracker can upload data from them to the server.

In the use case of a UAV, the I²C bus could also be connected to the flight control computer, to allow the tracker to report back flight information as well as GPS positions.

I²C is a two wire communications interface between one Master[9] and multiple Slave devices. The two wires are Clock and Data; both are pulled high with resistors, and can be taken low by devices (slave or master) using an open collector. Initially, both clock and data are high. When a master wants to query a slave it will take the data line low, then will start running the clock (a square wave). After this it will transmit a signal — the 7 bit address of the slave device followed by a read/write bit, taking the data line low or high in line with the clock (the data line is set while clock is low and read while clock is high). The slave will acknowledge this by pulling the data line low (ACK) on the next clock pulse. After this, the master or slave will transmit the data being read or written, byte by byte, followed by further ACKs. If a slave device is unable to send the data required in time it can tell the master to wait by 'clock stretching' — holding the clock line low so that the next clock pulse cannot appear. The master will detect this and wait for it to go high again before continuing. The transmission is ended by taking the data line high while the clock line is high.

When the trackers were received for this project, the I²C bus was not working. There were no errors given, but the trackers were unable to report either temperatures or shocks.

[9]The protocol does support multiple Masters, but not all devices are capable of this and only one Master is used in the case of the Durham trackers.

The sensors used have addresses set in hardware. The LM75 temperature sensor has eight pins; three of these are used for setting a slave address on the I^2C bus by tying them high or low. In the case of the Durham tracker, the LM75 was hard wired to address 73 (1001 001, where the four most significant bits are set internally).

The three orthogonal shock sensors are connected to the Telit via the MSP, which connects to the I^2C bus. The MSP has an address set in code (72); when the Telit module queries the MSP it will return a list of any shocks that have been detected since last queried.

As the I^2C was not working, the problem would logically be in either the creating of the bus in software on the Telit, the addressing, or a hardware fault. The addresses being used by the program matched those set on the slave devices, and a scan of the I^2C bus (telling all devices to report and checking for responses) returned nothing. A hardware fault seemed unlikely, as it would have to be a fault on all of the boards.

Testing the I^2C clock and data lines with an oscilloscope gave the expected square wave for the clock, and the signal 100100101 on the data (for the address of the LM75, 73). As described above, this signal starts with the address of the slave device that is being queried (1001001), followed by a read/write bit (in this case 0 for writing, but the same response was given when set to 1), followed by the ACK bit. All but the ACK were set by the master (the Telit); the ACK bit should be taken low by the slave being addressed. It remained high, however, meaning that the slave device had not responded.

As mentioned, a hardware fault on every single board seemed unlikely — unless it was a fault in design. It transpired that the clock and data lines had been connected back to front, so that all slave devices were the wrong way around. Once discovered this had a simple fix; changing the designations of the GPIO pins on the Telit allowed for the two to change places without requiring a hardware modification.

5.4.2.9 User interface

Technical abilities aside, the feature that any user of a GPS tracker would be most interested in would be having an intuitive user interface; both in terms of the unit itself, but also in terms of the website (or software) for looking at the available data.

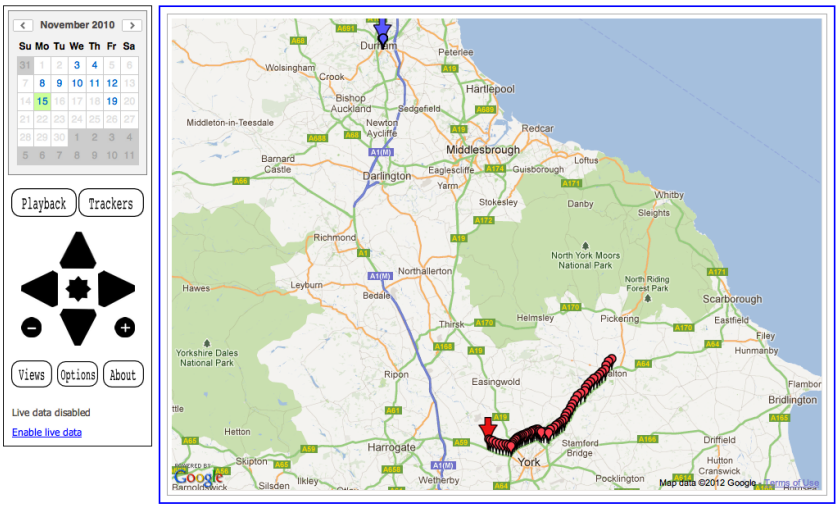


Figure 5.10: The original website design

The website that was being used with the trackers when taken over worked, and worked well, but had several flaws that could be improved upon. To start with, the interface was not entirely intuitive for some uses. It also could not be easily used by mobile devices, and did not have a good method for entering a range of dates over which the data should be viewed (points could only be shown for one particular date at a time; it was not possible to view data over a range of dates, or to view data from only a particular part of a day).

The major problem with the site came to light after the modifications were made for allowing rapid GPS readings. This change meant that the website could now be trying to display many thousands of markers onto the map, which made it unusably slow.

The old website (see Figure 5.10) was built using the Google Maps API, Version 2.[10] The interface consisted of a main section which showed the map, and a side bar with a number of controls. These controls included a calendar for selecting a date (this helpfully highlighted the dates which contained readings), pan and zoom controls (which replicated the ones available as part of the API), a button for enabling live data (where the markers were updated periodically to allow for any newly arrived points), and buttons for a range of submenus:

[10] Version 2 was officially depreciated on 19th May 2010.

Playback This sub menu has a dropdown list of 'speeds' (10x, 100x, 1,000x, 10,000x) and a 'Play' button. This was an interesting function, which displayed the markers on the screen one by one at the times at which they were recorded, allowing the user to see which order the markers were made, what speed the tracker was travelling at, and where any breaks in recording came.

Trackers This listed all the trackers which were active during the date selected, and allowed for them to be deselected so that only a limited set were visible. This seems to only affect the data shown on the map, and not any of the data tables or graphs.

Views The first three options on this menu replicated the Map Type control from the Google Maps API, and allowed for selection between Map, Satellite and Hybrid. The remaining options removed the map entirely, replacing it with other information:

- 'Data' showed a table with additional information obtained by the tracker — the time (both of the reading and of it being uploaded), tracker ID, altitude, number of satellites visible, sensor data (temperatures, shock sensors, and anything additional on the I^2C), etc. This was colour coded to match the trackers (with the background colour of the entire row matching), but in some cases the colours shown (coupled with the small text size) made reading this table difficult.
- 'Graphs' showed a range of graph options, with options for Altitude, Balance, GSM signal strength, Position accuracy, Internal temperature, Satellites seen, Shock times, Speed, External temperature, Time to GPS lock and Battery voltage. These were created using the Google Charts API, which did not cope well with large numbers of data points (returning 414 errors[11]).

Options gave a range of things that didn't fit under the other submenus, including links back to the Home screen, logging out, and a fullscreen display button

[11] HTTP status code 414: Request-URI Too Long; this is due to too many points being passed to Google's servers.

(where the map filled the entire screen, at the expense of any controls except pan, zoom and map type). Other links were:

User Management linked to a page where the various usernames on the system could be linked to a list of trackers that they were allowed to view, a list of data they are allowed to see for each tracker, and the period over which they are permitted to view.

Geofence Manager allowed for a series of areas (circles of set centre and diameter) to be specified. If enabled, they would email a group of addresses if a tracker or trackers crossed into or out of the area.

Tracker Manager allowed for commands to be sent back to a tracker when it next logged into the server to report a position. The commands were basic, and limited to selecting a 'profile' — selecting between files preloaded onto the trackers, which contained settings for how often to take readings, etc. Other than this, the main disadvantage to the method was that it would only update when the tracker logged into the server — there was no way to instruct the tracker to make an immediate change.

Some of the functions under this submenu were known to not fully work.

About gave a list of credits, with the names of people who had been working on the system and a link to the University website.

One more notable aspect of the old website was that the tracker colours were based on the trackers which were visible on the particular day being shown. This meant that the tracker represented by a particular colour was liable to change between days, if there were several trackers being used.

The website was rewritten using the latest version of the Google Maps API (Version 3), which is designed to work on multiple platforms, including mobile phones. This is a useful addition — in many use cases the user will not necessarily be sitting at home or in an office; they will be out and about, and allowing them to see the tracker positions and status on a smartphone will be better than forcing them to wait until they return.

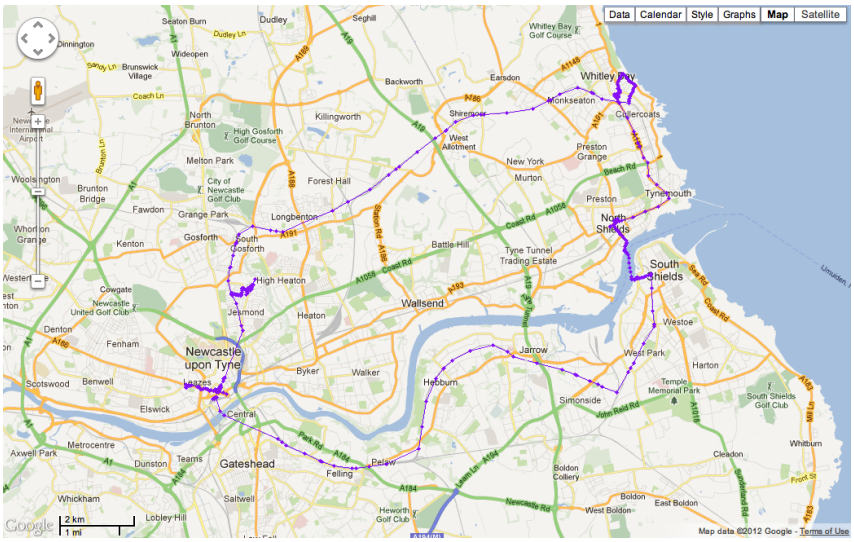


Figure 5.11: The new website design

The main basis for the design of the new site was to allow for all the data to be visible on the same screen, allowing the user to analyse points easily. The map should be large and easy to view. To this aim, the main part of the redesign (see Figure 5.11) was a map which entirely filled the window (showing markers for all points in the current timeframe, with different colours for each tracker). The map had standard Google Maps buttons for map type (map, satellite, terrain, etc.) and for navigation (zoom and pan), which would automatically adjust their style based on screen size (for example, changing the row of buttons for map type to a drop down list on a narrow screen). Onto this were added four extra buttons — Data, Calendar, Style and Graphs:

- The Data button displayed a screen overlay showing a list of all points in the timeframe, along with additional information — the time (both of the reading and of it being uploaded), tracker ID, altitude, number of satellites visible, sensor data (temperatures, shock sensors, and anything additional on the I^2C), etc. The rows were colour co-ordinated to match the trackers on the screen (with a coloured tab at the start of each line), and clicking on an individual row would highlight on the map the tracker position mentioned. The rows would also become increasingly highlighted as the battery level dropped

between two thresholds — giving an easy to read indicator that a tracker was going to be running out of power shortly. To allow for large amounts of data to be displayed without filling the web browser's limited memory, the table was limited to 500 entries. At the top and bottom of the table were forward and back buttons, allowing the user to page through the information.

- The Calendar button displayed a screen overlay showing a calendar and a pair of dates and times — the Start and End of the current timeframe. Clicking on dates on the calendar allowed for the dates to be changed, and the map updated accordingly.

- The Style button displayed a screen overlay showing various settings for the way in which the positions were displayed on the screen. Lines between points could be switched on and off. A list of all the trackers visible in this time period allowed for their selection and removal (letting the user see a limited set of trackers if an area was congested). There were also three options for the colour scheme used — 'Tracker', where each point was given the colour of the associated tracker; 'Time', where the points are given a brightness based on the time that the position was recorded (from black at the start of the time period to full colour at the end, with the colours of the trackers being the same as for 'Tracker'); and 'Battery', where the points are given a colour based on how much battery life was left (ranging from black at minimum battery to full colour at maximum).

  The colours for the markers were selected from a hash[12] of the tracker ID (in most cases the IMEI), which meant that the same tracker would always have the same colour (although not guaranteed uniquely). As a range of 20 numbers were available, all equally spaced around a colour wheel to ensure that they look different, the number of collisions is small.

- The Graphs button displayed a screen overlay with a dropdown selection box at the top. This could be used to select a range of graph options — Battery,

[12] A CRC32 redundancy check number is calculated to convert any text into a number, and the modulus taken to select one of the 20 colour options.

Altitude, Number of satellites, Speed, Temperature, Internal Temperature, and a cumulative count that let the user see when the readings were taken. These graphs were created through a custom PHP script, as the Google Charts API was unable to cope with the high numbers of points that might be viewed. The graphs can easily cope with displaying many years' worth of points.

One big change between the old and new sites was that after the redesign all the data was retrieved on the fly without having to reload the page, using AJAX calls to a background script that could pull the data from the database. This meant that making small changes to the data being retrieved was considerably faster for the user, and didn't require the entire page to be reloaded.

As mentioned above, the major problem with the old site was that it could not cope with the large numbers of points that were available when the tracker was reporting rapidly.

A standard Google Map will contain markers that are processed client side. The code for the map will contain the positions for the markers, and any additional information (their colour, for example; or the text to appear above them in a bubble (or 'infowindow') when they're clicked). Once the map has been displayed the markers will then one by one be placed onto the map. The API will manage these markers, and will take care of their events (such as clicking), displaying them on the screen, and anything else that happens to them. There is a large number of things that they can be programmed to do with relatively little difficulty on the part of the programmer.

This works well for small numbers of markers, but will rapidly start to cause problems when the numbers start to get into the hundreds. The map will take longer and longer to load, and will consume more and more of the user's memory, making it slow and unwieldy. In our use case, where many thousands of markers might be being displayed (several trackers reporting over several days), this proved unusable.

The official recommendation for use cases where large numbers of markers are required is to cluster points together. At low zoom levels (where a large area is visible) areas which contain many points are represented by a single marker, often

with a number to show how many points it represents. This reduces the total number of markers that need to be displayed at any one zoom level, and functions are available to manage this automatically.

This is a reasonable solution if suitable for the data that is being plotted — if the map shows positions of supermarkets, say, then knowing the number in a particular area and being able to zoom in to see their exact positions is very reasonable. In our case, however, where the markers represent accurate route positions that may have been recorded at different times, clustering would only show how much a tracker had been used in a given area rather than the paths that it took across it.

Instead it was decided to create a new map layer, with a tile server which calculated the positions of the points and drew them onto the map on the fly as a background overlay. This reduced the functionality of the markers (as flat points drawn on the background there were no automated functions for them), but allowed for a considerably higher number of points to be visible, raising the maximum number that could be reasonably showed on the map from a few hundred to hundreds of thousands. The size of tiles was also increased to reduce overheads, as a major problem was that there is only a limited number of connections to any particular server permitted by a browser[155, 8.1.4. Persistent Connections — Practical Considerations: 'Clients that use persistent connections SHOULD limit the number of simultaneous connections that they maintain to a given server. A single-user client SHOULD NOT maintain more than 2 connections with any server ... These guidelines are intended to improve HTTP response times and avoid congestion.'] (meaning that only a limited number of tiles can be loaded simultaneously).

The main downsides to this method, other than the lack of in built functionality, are that it puts an additional load on the server and increases the time taken for tiles to load. The delay for drawing these background tiles — even when the number of points is small — is still noticeable, and can rise to 20 or 30 seconds in extreme cases where a lot of points are being loaded. However, this is still far faster than the delay caused by using the standard client side markers (when larger numbers of markers are being displayed), and allows for a considerably larger number of points to be shown.

In a direct comparison, loading all data from one particular day in which 3091 (visible; i.e. reported with a valid GPS lat/long) points were reported took an average of 1.57 minutes to display on the old website and an average of 7.03 seconds on the new[13] (over 13 times faster). While the old site is unable to display multiple days at once, using this method over a long time period (many months, say) will be possible on the new map.

The lack of built in marker functions can in some respects be rectified by replicating them in the program. An example of this is the infowindow. Although the API was unable to create infowindows automatically, it was possible to instead catch the user clicking on the map, calculate the nearest visible point, work out whether the click was sufficiently close for it to have been intended for that point, and display its details on the screen. Another example is where the API would normally be able to position the map automatically to encompass all visible markers; instead it was necessary to calculate the bounds of the markers separately and instruct the map to be moved to fit them. These and other changes brought back most of the functions that a user would require from the map.

There are several features of the old website which have not been recreated in the new:

- The Playback button has been removed, and cannot easily be implemented as the marker positions are now drawn as one instead of being individually changeable. However, as one of the colour coding options allows for shading based on the time at which readings were taken, the same data can be obtained through other means.

- The user management system — controlling who has access to what data — has not yet been reimplemented. This can easily be added to the system in a future version.

- The ability to set geofences has been removed — again, this can be put back in a future version.

[13] Averages taken from three successive hard refreshes of each map page, with the times taken from the 'Network' tab of Google Chrome's built in Developer Tools.

- The ability to change between profiles has been removed. The general functionality provided by this (the ability to send commands to the trackers, when they report to the server) has been superceded by the ability to send commands via SMS text messages. This gives an instant response as soon as the tracker is in range of the mobile phone network, without waiting for it to communicate with the server. That being said, the specific functionality of having a range of preset profiles that can be switched between is currently unavailable — although it can be returned without difficulty.

- The live data option has been removed, but may be returned in a later version. It is somewhat less useful in the case where trackers are reporting at long intervals.

One final change which has been made is to increase the precision available, by fixing a bug on the original website. The NMEA longitude and latitude given by the Telit are measured to four decimal places (given in degrees and decimal minutes), which gives a maximum precision of approx. 2×10^{-6} decimal degrees (approx. 0.22m or less). This is stored in the MySQL database on the server, but the database will only give an output to six significant figures unless specifically instructed otherwise. This means that — for the UK — positions will be given to 5 decimal places for longitude (approx. 1m) and only 4 decimal places for latitude (approx. 10m). Changing the database query to ask specifically for as high a precision as was available gave a considerable improvement to the map, therefore.[14]

5.4.2.10 Internal battery

Relatively early in the tracker's design, a decision was made[146] to use alkaline AA batteries instead of rechargeable batteries which could be recharged from the mains. This made sense for their use case at the time, but is not useful in many circumstances — in the majority of cases it would be easier to find mains electricity

[14]The database stores floats internally as doubles, meaning that the conversion to float is approximate and only fully accurate to 7 significant figures. This means that some small errors may be introduced — these are likely to be minor, however, and will be better than those caused through imprecision.

(or another power source) to recharge the batteries than it would be to carry around a large quantity of batteries.

For UAVs in particular, the additional weight and bulk is far greater than it could be.

The Telit is capable of battery charging. The module contains two relevant pins; VBATT and CHARGE. If the Telit is being run off a LiIon/LiPoly battery, the battery needs connecting to VBATT and a constant current source[15] to CHARGE. The Telit will run from the battery, and will recharge the battery if the current source is attached to CHARGE.

Currently the AA batteries are connected (after voltage regulation) to the VBATT pin. In order to use the rechargeable batteries there are two options — either to entirely remove the voltage regulators, supercapacitor, and all other elements related to the alkaline batteries, or to use the alkaline batteries as an optional power source from which to charge the rechargeable LiPoly batteries. This latter appears to be the best option; while it would be slightly bulkier than the former it would allow for AA batteries to continue to be used in circumstances where that would be beneficial, while allowing for them to be easily removed without power loss to the tracker.

Although a circuit design for this has been made it has not yet been implemented, as this modification would require a redesign of the PCB.

5.4.3 Results from changes

5.4.3.1 Previous trials

Several tests were made of the trackers before they were used as part of this PhD research (i.e. without any of the modifications described above); these are mainly documented by Hancock[144], and are described below. The general conclusion reached was that 'some parts of the code were ineffective, and there were a few flaws to be looked into and corrected.'

[15] Recommended 400mA; will take up to 1A.

Val Thorens Val Thorens is a ski resort in the French Alps, and the aim of this experiment was to test how well the tracker worked with prolonged use. This trial is highly comparable to that described in Section 5.4.3.3, where a tracker was taken up the Merrick in Scotland.

The tracker was first placed in a fixed location for two hours. During this time, only four data points were recorded. Of these four, three are scattered over an area a mile wide. Only two data points were successfully uploaded to the server after being taken; the other two were uploaded later (during 'backfilling'). This was put down to issues with GSM signal strength.

The fact that only four readings were taken is odd. The report says that the tracker was set to take readings 'as often as it could', which would in normal conditions equate to one reading every approximately 5 minutes (completing a cycle of taking a reading and uploading it to the server). This would mean that in two hours 24 readings should have been taken — six times more than was the case. This may have been in part due to the fact that the tracker was being used abroad (the SIM was roaming), but may also be due to the long time taken to upload data in bad conditions (with repeated attempts to upload each point).

For the second test the tracker was placed into a rucksack and carried for six hours on a walk. During this time sixteen points were recorded, of which only three were uploaded in real time. One point did not have a valid GPS position. Fewer than half of the intervals between readings (7/15) were around 5 minutes — four were around 10 minutes, and the remainder were around 30–40 minutes. This was put down to potentially being due to the trackers 'backfilling', which took a considerable time. It is notable that three of the half hour intervals are immediately after each of the successful real time uploads, which is when the tracker would be attempting to upload any old points.

Durham For this trial, the tracker was put in a fixed position 'for extended periods of time over a week', recording 263 readings in just over 10 days (10 days, 4:05:58 hours), of which 59 were invalid. Very few of the points — approximately one fifth — were uploaded to the server in real time. The points are all within an area approx. $350m^2$, with the majority of points clustered into the middle of this area (approx.

$200m^2$).

5.4.3.2 Subsequent trials — Failures

Two trials were run with the improved trackers that unfortunately did not give useful results.

Postal service The first involved sending the tracker through the postal system, while switched on and taking readings. This continued reporting from while being packaged onwards, and the results appeared at first very favourable — the website showed it passing through the University's postal system (including a period indoors where it could not receive GPS), being delivered to Royal Mail, travelling to an unknown building in the centre of Durham, and then to the local sorting office. Unfortunately, only a few hours after leaving the department, the tracker stopped responding while at the sorting office. When it arrived at its destination, it appeared that the inductor had shaken loose from the PCB and stopped the unit from functioning. This may have been the root cause of the failure; another alternative explanation is that it was placed into a larger van which blocked its ability to either take GPS readings or connect via GPRS.

Australia The second trial was to take a Durham tracker across Australia, from Darwin to Adelaide, driving along the Stuart Highway. This was a journey of over 3,000km, and most of the trip was a long way from any form of civilisation, let alone mobile phone reception. The results from the Durham tracker could be compared against the tracker supplied as part of the World Solar Challenge (the Durham tracker would be carried by the Durham Solar Car team, DUSC). This would have been an excellent test for whether the Durham Trackers could cope with extremely long periods without being able to upload data, as there is no mobile phone reception for much of the route. Unfortunately, due to a problem with getting the SIM provided by a local mobile company to work with the tracker, this test had to be cancelled at the last minute. The tracker did, however, survive the rigours of the journey without any hardware problems.

5.4.3.3 Subsequent trial — Scotland, June 2010

A tracker was taken to Scotland, being carried both in a car and in a rucksack. This included, most notably, a walk up the Merrick in Galloway — the highest mountain in the Southern Uplands — in relatively bad weather conditions.

The tracker was set to take readings as often as possible for over a week. Readings were taken every 5–15 seconds, with the majority of readings at the lower end. Data was uploaded to the server (assuming signal was available) every 20 minutes.

It should be noted that the tracker used in this trial had not all of the modifications made as described above; the most notable difference being that it did not have a box.

Over the course of the week the tracker failed on 6 occasions — that is, stopped working until it received human intervention. Of these, two were believed to be from batteries running low (running out before the user had already replaced them); two from an error caused by the balance check (see Section 5.6.1); one from a hardware problem (one aerial cable coming loose); and the last from an unknown cause.

When climbing the Merrick, much of the early part of the route was through woodland with little view of the sky. The weather conditions were poor, with fog and rain in places, and mobile phone reception was entirely unavailable for the whole of the journey. The tracker was placed loose inside a rucksack, along with other items, and was subjected to a lot of vibration along the route. Despite these difficulties positions were taken at very regular intervals, and proved to be accurate to the path taken. The data was uploaded after leaving the mountain, when coming back into mobile phone range.

This compares very favourably to the trial made at Val Thorens (see Section 5.4.3.1).

The tracker was used for several tests in the run up to being taken to Scotland, and after returning it continued to be carried for some while after, until finally running out of credit. Over the course of 22 days it had used £10 of credit and had transmitted 110,106 data points. This means that it had averaged a reading every 17 seconds, despite spending some large parts of that time being switched off or uploading to the server.

5.5 Comparison with commercial trackers

A comparison was made between the Durham tracker and other commercially available options. Note that this is not a complete comparison against all possible alternatives; nor is it a true comparison, in that the options compared against have marginally different use cases. The most notable difference between the Durham tracker and most other tracker options mentioned is that the Durham trackers are modular; they have on board sensors (shock and temperature), and can be easily extended to give other sensor inputs. They can also communicate with other electronics, such as an on board control system for a UAV.

The list below therefore gives an indication of the types of trackers available commercially, allowing for the capabilities of the Durham Tracker to be put in context.

5.5.1 Globalsat TR-102

The TR-102[156] tracker from Globalsat is a personal tracker which can be used as a communications device, an emergency locator, and a remote tracker. It can report positions either by SMS text message or over GPRS to a computer.

The user interface is fairly simple. It consists of a number of buttons:

On/Off turns on and off the device

SOS transmits a text message to up to three preset phone numbers, containing the latitude and longitude of the device. This is designed as an emergency button

Answer/Hang up starts or ends a call. It is also used to put the device into a state from where it can be connected to a computer via USB.

Make phone call / Volume control are three buttons which each have two functions. The first function is to initiate a phone call with one of three corresponding preset numbers. The second functions are volume up, volume down and mute — these secondary functions are active during phone calls.

It also has a switch for locking the buttons, three status LEDs (GPS, GPRS and power), speaker, microphone, in built antenna and sockets for power and USB.

The device is small and portable, measuring 115 x 45 x 22.5mm. It takes a rechargeable 3.7v Li-Ion battery. Most of the settings for the tracker can seemingly only be set by connecting it to a computer via USB and using the proprietary software provided. This software runs on Windows; other operating systems do not appear to be catered for.

The tracker will respond to a telephone call[16] or to a text message by reporting its position. This occurs without requiring user intervention (or consent), which means that it can be used for remote asset tracking. It can be be set to either return its position via SMS, to the number from which it was contacted, or via GPRS. For GPRS it will connect to a preset server and send a string containing the data. The server can then optionally respond by asking the tracker to continue giving periodic results.

The major downside to this method appears to be that there is no way to initiate the tracker communicating its position via GPRS from the device itself.

The TR-102 is able to report at a range of periods — from 5–86400 seconds while keeping GPS and GPRS active between readings, or from 30–86400 seconds while sleeping between readings. This latter saves battery life, but increases the time for both taking GPS readings (increased time to first fix from a cold start) and transmitting over GPRS (requiring time to re-initiate the connection to the server).

Software is available from Globalsat for the TR-102 which is designed to run on a user's computer as a server for the tracker to connect to. The 'TR Management Center' will keep a record of the readings given, and shows a map of positions for up to five trackers. The program appears to work reasonably well, and setup — once the difficulties of getting a port opened on a Windows machine to the outside world were passed — fairly painless. The software does have the disadvantage that the data is only accessible from one computer — there is no way to see it from another machine, from a mobile phone, or remotely over the internet, for example. But for many use cases this is unimportant, and as the trackers are able to report positions via SMS this is less of an issue.

The TR-102 is also supported by a number of online tracking servers (see Section

[16] Depending on settings. On receiving a call it will either respond with a location or it will ring and can be used as a mobile telephone.

5.5.11), and comes with an instruction set so that a custom server can be written for it. Using this, a front end was written for the Durham tracker server to allow data from both to be plotted on the same system.

The TR-102 is compared against the Durham Tracker in a range of experiments in Section 5.5.12. While both are GPS trackers which report their position over GPRS, their use case is slightly different. The TR-102 does not have sensor inputs, and does not have outputs suitable for connecting to a UAV's control system.

5.5.2 Smartphone GPS tracker software

There are a great many smartphones on the market, and it is unusual now to find any which do not have both a GPS receiver and a mobile data connection. There are many options for using a smartphone as a GPS tracker, a small selection of which are listed below. It should be noted that this list is incomplete, and that it focuses on software which is available on the Android operating system. The list only contains software which allows for a trace to be recorded remotely — software that records a trace on the phone, or that only reports the current phone position without a historical trace, are not included.

One notable aspect of smartphone trackers is that as they are capable of multi-tasking they are easily able to upload data while taking GPS readings. They also come in all shapes, sizes and costs — if a small tracker is required, phones can be as small as (for example) the Sony Ericsson X10 Mini, measuring 83.0 x 50.0 x 16.0 mm and weighing 88g.

5.5.2.1 Google Latitude

Google Latitude is a tracking system designed for sending positions to friends and family. Your position will either be calculated from your smartphone using pure GPS, or from Google's cell ID and wifi location databases (this can take positions either from a phone or from a web browser on a computer). This position is recorded on their servers, and reported to specified friends. The location accuracy has two settings — either it will give your exact position to the best of its ability, or it will list the city you're in. Friends can see your last reported position, but cannot see

your history. This data can also be made publicly available and used in conjunction with other software, using provided KML, ATOM or JSON feeds.

There is, however, an option for showing your history on Google's mapping website. This is only available to the account which the smartphone is tied to. The data can be either viewed online, or exported for viewing elsewhere. Google attempts to sanitise the data somewhat; removing points which it believes to be erroneous. It also attempts to give statistics based on your habits and other information about you — how far you travel, where you spend your time, how long you spend at home per week, where you live and work, etc. These are not particularly reliable.

Although Latitude is not specifically designed for tracking, using the history means that it can be used for this. The main difficulty with doing so is the infrequency of the readings. It will only take readings from time to time, to reduce battery drain (GPS receivers use a comparatively high amount of power).

5.5.2.2 OsmAnd

OsmAnd is an Android mapping client for Open Streetmap (OSM; see Section 6.3 for more). It is designed primarily for displaying the OSM maps, but also contains functions for recording GPS traces and for uploading them to a server, mainly to allow for future editing of maps. The positions can be uploaded at a variety of intervals, ranging from 1 second to 5 minutes, and can be set to upload to any arbitrary URL.

Although this is not specifically designed for tracking, the ability to upload points at fast intervals to a remote server allows it to be used as such.

5.5.2.3 Greenalp Real Time GPS Tracker

This application is specifically designed for GPS tracking. It uploads data points to their remote server, and has a user control system which allows users to select who is able to see the current and past positions of the smartphone (publicly available; specified users; personal only). The positions are shown on a map on their website. The application description gives several use cases — including sending your location to friends, and using a smartphone for pet tracking.

There are a number of settings that can be used to configure how the tracker responds. These include a threshold for quality of GPS reading, a threshold for minimum distance between readings before uploading, and thresholds for how long to wait between readings if GPS is unavailable. There is a setting for how often to update the GPS position. The tracker also supports AGPS, and can be controlled via SMS. While being used to show positions to friends it has a messaging system between the application and the website.

5.5.3 SPOT Personal Tracker

The SPOT tracker, one of the Satellite Messenger devices produced by SPOT (a subsidiary of Globalsat), is a GPS tracker that transmits its position via the Globalsat low Earth orbit satellite communications network rather than via GPRS over the mobile phone network. This means that it does not have the difficulty of being unable to transmit a position when there is no reception — which is likely to be the case in situations where the user is in trouble or danger. It can be used in several ways — as an emergency location device, sending its position to the local emergency services when the user presses a button; to check in the user's position, sending it via email or SMS to friends or family; or as a tracker that reports positions to a website. It is designed to be a highly ruggedised device, for taking on outward-bound type activities, and claims a 14 day battery life while in tracker mode. The tracker measures 111 x 69 x 44mm, and weighs 209g.

5.5.4 Globalstar SmartONE

The SmartONE[157] is a tracker designed for both asset and vehicle tracking, and — like the SPOT tracker — transmits positions via satellite, using the Globalstar network. It sends information either by text or by email. It runs off a power supply or four AA batteries, giving up to 3 years of battery life with infrequent position readings. It contains motion sensors and supports other, custom configured sensors, with a serial connection to interface with sensors and deliver messages from the user. The tracker measures 3.25x6.5x1", and weighs 385g.

5.5.5 MeiTrack MVT-600

MeiTrack[158] make a large number of trackers, including vehicle, asset and personal. One example is the MVT-600 vehicle tracker, which includes inputs (mainly as triggers for sending positions, for example when connected to a car door switch, but also analogue sensor inputs whose values are reported when uploading positions). It also has a built in camera and remote audio feed. The tracker weighs 220g and measures 103x98x32mm.

5.5.6 MeiTrack VT-400

The VT-400[159] is another tracker from MeiTrack. It is a ruggedised tracker designed for heavy machinery, and it includes outputs as well as similar inputs to the MVT-600. These outputs are triggered by a remote request from the user, and are designed for things such as cutting out the engine. It measures 123x83x37mm and weighs 350g.

5.5.7 CelloTrack Solar

Part of the CelloTrack family, from Pointer Telocation. These are highly rugged asset trackers designed to be strapped to items such as shipping containers. They include movement sensors, and will report the position of a moving item more often than a stationary one, although they are designed to give extremely infrequent readings (once per day, although this is configurable to a small extent). The CelloTrack Solar[160] is notable because it is powered by a solar panel, charging an internal battery. This means that it can be attached to the outside of something like a shipping container or a train and left there indefinitely. As this tracker requires a solar panel and is designed to be attached to a large item it is larger than some of the other trackers mentioned here, measuring 585x130x30mm.

5.5.8 ATrack AY5i

This vehicle tracker[161] is unusual, in that instead of using GPRS to transmit data it uses 802.11b/g wifi. This means that it must be within range of an available wireless

network in order to be able to report; however, as it contains a 8MB internal memory it is also able to log positions while out of range. It supports a 1-Wire interface, which is a similar protocol to the I^2C supported by the Durham Tracker, and has a built in accelerometer for crash detection as well as analogue sensor inputs and digital trigger inputs. It measures 100x65x26mm, and weighs 161g.

5.5.9 UAV trackers

There are several trackers designed for attaching to model aircraft. These tend to be small and lightweight, but with a lower range method of transmitting and often no logging.

One example is the TinyTelemetry GPS tracker from Immersion RC.[162] This sends position data and sensor inputs back to the user using the audio stream of a video transmitter, but requires the user to be in radio range. It is extremely small at 55x15x10mm, and weighs 12g.

A similar device is the EagleEyes FPV Station, from EagleTree Systems.[163] This is a system for providing a camera for a model aircraft, with an on screen display. Position data and inputs from a range of sensors both overlay a video feed and are transmitted to a proprietary receiver attached to a computer to allow for graphing of data and plotting routes.

The TinyTrak SMT[164] from Byonics is slightly different. It's a surface mount version of the TinyTrak APRS GPS position encoder. APRS (Automatic Packet Reporting System) is a worldwide amateur radio protocol for transmitting positions. Each transmitter periodically transmits a packet containing its ID, GPS position, and the local timestamp. This is then propagated by other units, either by repeating the packet's transmission or via the internet. The data can then be viewed to give current or historical positions. The TinyTrak SMT is a particularly small APRS encoder, measuring only 1" x 0.925" x 0.16" (a separate power supply, radio and serial based GPS receiver are also required). The APRS protocol allows for a 'comments' field in the packet, which could be used for including any sensor data. It should be noted that using APRS can cause potential difficulties when away from urban areas as there is no guarantee that the transmitted signal is received

and forwarded; however, a series of UAVs would be able to repeat signals between themselves to form a chain back to a longer range repeater, and the UK has a number of automated wide area VHF repeaters which give a wide coverage. Another potential issue is that there is no aspect of privacy — any information transmitted by this means becomes public and is distributed worldwide.[17]

Complete autopilot systems will also usually include a GPS receiver for calculating positions, and several have capabilities for reporting back flight data to the user, usually by radio.

5.5.10 Animal trackers

There are many questions that biologists have about the behaviour of wild animals, such as foxes, which are difficult to answer due to lack of unbiased data. Wild animals are almost by definition elusive and difficult to accurately observe. One means of obtaining data is to remotely monitor the animals using animal attached transmitters. These can be GPS based, or can be radio transmitters which are then triangulated by teams of observers.

Most wildlife GPS receivers are prone to error due to their use case (they will often be transmitting either under heavy undergrowth, in underground dens, or in urban canyons, and will need to have an extremely long battery life). There has been interest in developing the Durham tracker for tracking foxes in Bristol,[144] due to its GPS capability and ability to transmit positions over GPRS.

Animal trackers share a number of similarities to UAV trackers — they need to be small, light, low powered devices with a long battery life, capable of relatively high speed tracking and tracking in difficult situations. Despite the problems associated with animal trackers there are a number of devices commercially available.

Vectronic's GPS Plus Telemetry Collars[165] take regular GPS readings as well as other sensor inputs (temperature, movement, etc.), log them locally, and can then report them via several different means — by satellite (ARGOS, IRIDIUM or GLOBALSTAR), by GSM, by radio (UHF or VHF), or by detaching from the animal for later retrieval. While many of their collars are designed for larger, heavier

[17] Including via satellite; the International Space Station, amongst others, will relay APRS data signals.

animals, their smallest is designed for tracking birds. It weighs from 100g and offers VHF or GSM for data transmission.

Another example is the Quantum 4000 Advanced GPS Collars from Telemetry Solutions. These include temperature and mortality sensors and can store 60,000 readings on internal memory. Readings can be taken as often as every 5 seconds, and transferred over UHF. They offer a customised collar for any animal specification, with their smallest weighing from 30g. This smallest uses UHF for data transfer with a two way connection — or alternatively can be set to fall off the animal after a set period, with a VHF beacon for locating the collar and recovering the stored data. Battery life is quoted as being up to 270 days with infrequent (twice daily) readings.

5.5.11 Online tracker servers

There are several online servers — either downloadable for a user's own server, or where users can register for accounts — which are not tied directly to one tracker, and often not to one specific manufacturer's range of trackers. Instead they offer a generic system which can be connected to by any supported tracker; taking the data (often sent in a proprietary format over GPRS) and decoding it.

One example of these is provided by GPSGate.[166] This company provides both servers and clients, and either licenses the servers to be installed on a user's own machine or charges for use of one of their dedicated servers. The clients connect to a server for accessing the data. This system offers a number of functions, and they support a wide range of commercially available GPS trackers (personal trackers, asset trackers, and vehicle trackers). It is aimed both at people who wish to track large numbers of trackers (such as fleet managers), and at companies supplying trackers to end users (the software can be rebranded, allowing it to be resold as part of a complete tracking package).

EZFinder[167] is another online tracker server, owned by Globalsat (the manufacturer of the TR-102 described in Section 5.5.1). It similarly supports multiple trackers, but a considerably smaller range, and unsurprisingly all are manufactured by Globalsat. It is free to use the basic system, but some features are only avail-

able in a 'premium' version. Globalsat also has a subsidiary, TraqLogic, who offer a tracker server called WebtraQ.[168] This is a paid for service, and although the only trackers which are listed as being supported are manufactured by Globalsat, it says that it will integrate with other trackers. The software can either be run on their own servers or on customers' servers.

A third — TrackingMate — is provided by MeiTrack,[169] the company who makes the MVT-600 and VT400 (described in sections 5.5.5 and 5.5.6). This is another web based tracking server which can be rebranded to suit an end user's application, and which is designed to work well with mobile phones. They also offer a tracker application which can be run from a standard computer.

5.5.12 Comparisons with Durham tracker

It is difficult to exactly compare the Durham tracker with others on an entirely even footing, as there is a large spectrum of trackers aimed at many different niches in the market. Our system is modular and takes in a variety of I^2C sensors — it's highly configurable while remaining a pre-set system that could be used off the shelf as a personal, vehicle or asset tracker. The trackers listed above may not all be exactly comparable, but are all potential alternatives for some applications. Most are not directly suitable for use with a UAV, for a number of reasons; either they have no sensor or communications IO for interfacing with the UAV controller, or they are too heavy for a UAV to lift, or they are designed to give positional updates too infrequently. However, that being said, the comparison made between the Durham tracker and other alternatives does allow for some degree of benchmarking — allowing us to see whether the tracker behaves notably worse or better than the alternatives in situations where they are on a relatively equal footing.

Two alternative tracker options were extensively compared against the Durham Tracker, in a variety of use cases:

- Stationary outside
- Stationary in a car
- Stationary inside

- Out of range of GPS
- Out of range of GPRS
- Walking in an urban environment
- Driving in a car

The two alternatives used were the Globalsat TR-102 (see Section 5.5.1) and a smartphone[18] running OsmAnd (see Section 5.5.2.2). Further trackers not tested are also included below, using data obtained mainly from the manufacturers' specifications.

The trackers were compared on a number of categories:

- Frequency of samples
- Frequency of uploads
- How well it copes when out of range of network
- How well it copes when out of range of GPS
- How well it copes when stationary
- Size
- Cost
- Weight
- Map quality

While some of these are more qualitative than quantitative, an attempt was made in all cases to put the tracker options in order of preference.

5.5.12.1 Experiments

Here follows a more in depth description of the use cases tested.

[18] An HTC Sensation, with the Android operating system.

Stationary outside The three trackers were placed on a table in a garden, 5.25m from any obstacles, with a clear view of the sky. They remained there for over three hours.

Stationary in a car The three trackers were placed on the dashboard of a car, parked on a suburban street on the outskirts of Durham. They remained there overnight.

Stationary inside The three trackers were placed on a surface one metre away from a window, remaining there for over three hours.

Out of GPS range The trackers were moved to a position from where they had no GPS signal but could still access GPRS, remaining there for a short period to ascertain their behaviour.

Out of GPRS range The trackers had their GPRS capabilities removed, and were placed indoors by a window where a GPS signal was available. They remained there for a short period to ascertain their behaviour, and then had their GPRS capabilities restored to them.

Walking in an urban environment The three trackers were placed into a small rucksack and carried across Durham. The walk included travelling through relatively built up areas, although Durham does not have any concentrations of high rise buildings. It took 40 minutes, and was 3km long.

Driving in a car The three trackers were placed inside a car and taken on a four hour drive, a journey of approximately 200 miles.

5.5.12.2 Frequency of samples

If tracking a fast-moving object such as a UAV, it is important to be able to take rapid readings. It is also important to be able to upload those readings in such a way as to reduce any down time where readings cannot be taken.

	Durham Tracker			TR-102		OsmAnd	
Experiment	No.	Avg	Total avg	No.	Avg	No.	Avg
Stationary outside	1122	8.92	9.63	259	41.79	1893	5.69
Stationary in a car	1423	9.31	10.12	-	-	2642	5.45
Stationary inside	1100	8.89	9.82	58 774	187.56 12.71	1870	5.78
Walking in an urban environment	229	9.42	10.11	32	75.29	391	6.14
Driving in a car	1301	9.70	10.61	24	60.26	2175	6.39
Totals	5175	9.24	10.07	1147	30.86	8971	5.83

Table 5.3: Number of readings, average time between readings, and (in the case of the Durham tracker) average time between readings if upload time is not taken into account. All times are in seconds.

The Durham tracker was set to take readings as often as possible, with uploads to the server once every half hour. The TR-102 was set to take readings at its minimum time of 5s, and upload every point when taken. OsmAnd, on the smartphone, was set to its minimum time of 5s, and also uploads every point when taken — although it is able to do so while simultaneously continuing to track, which the other two trackers are unable to do.

Looking at the results from all experiments (except those where GPS and GPRS were artificially made unavailable), and discounting time spent bulk uploading for the Durham tracker, the three trackers had average reading times of 9.24s (Durham Tracker), 30.86s (TR-102), and 5.83s (OsmAnd) (see Table 5.3). Including the bulk upload time changes the Durham tracker's average upload time to 10.07s. Interestingly the TR-102 reports every 5 seconds even when it has not achieved a new GPS reading, giving multiple reports for each timestamp. These have been discarded for calculating the above statistics.

Table 5.3 also demonstrates the reliability of the three trackers. The Durham Tracker and the smartphone both show a high level of reliability in the tests taken; the TR-102, however, has regularly stopped responding. For two of the stationary experiments it only reported for limited periods, and for the third it failed to report a valid position at all. In the driving experiment the TR-102 only reported 24 positions before cutting out, a very short way into the four hour journey.

Experiment	Number of uploads	Average time per upload
Stationary outside	5	167.80
Stationary in a car	7	174.43
Stationary inside	5	214.60
Walking in an urban environment	1	167.00
Driving in a car	7	179.00
Totals	25	182.12

Table 5.4: Number of uploads and average time taken per upload for the Durham tracker. All times are in seconds.

5.5.12.3 Frequency of uploads

The remote user will want to be able to know where the tracker is. In many cases it is not worthwhile tracking something if you can only find out where it was a long time ago, rather than where it is at the moment. It is therefore important for the tracker to upload its position regularly and often.

This is one category in which the Durham tracker is somewhat behind, in that it is set to bulk upload readings every half hour. The other two trackers tested upload instantly.

The main advantage of bulk uploading is that it can continue to take and upload readings even when no mobile phone reception is available — storing them up for upload later. It also greatly speeds up the time to take readings, although against the smartphone particularly this improvement is of a lesser importance.

The disadvantage is that, from the user's perspective, it is invisible for the length of time between bulk uploads. This can be improved by increasing the upload frequency.

Of the three trackers tested only the Durham tracker has a bulk upload. Over the course of the experiments, with it set to upload half hourly, the average time taken to bulk upload (during which time no measurements could be taken) was a little over 3 minutes (see table 5.4).

5.5.12.4 Out of network range

In many cases — particularly in situations where there is an element of risk or danger such as when searching for a missing person in a remote part of the country,

but also in everyday use such as when inside a large building — the GPRS network will not be always available. The tracker therefore needs to be able to cope with situations where it is unable to upload to the server immediately. In particular, it needs to be able to deal with situations where GPRS is unavailable but GPS (or other sensor data) is.

The TR-102 tracker, when out of GPRS range, was unable to store readings for later upload — if it does not upload the data immediately it is lost when the next reading is taken. OsmAnd, similarly, does not backfill readings — although it does have the capability of storing all data into the phone's memory, which could be used for later retrieval. Backfilling would be easy to implement, but is not included in this software. The Durham tracker stores all data that has not been successfully uploaded, aiming for perfect retention. As such, when out of GPRS range, it continues to take readings and uploads them when next back in range.

5.5.12.5 Out of GPS range

Similarly to the section above, it is not an uncommon occurrence to be out of GPS range. This is often the case when indoors, or in an urban canyon, or under heavy tree cover. The GPS tracker must be able to respond appropriately when unable to track via GPS.

When inside and out of GPS range (but still in GPRS range), the Durham tracker reported positions with no latitude or longitude, and reported that the positions were invalid. It still reported other information such as sensor readings, however.

The TR-102 continued to report positions every 5s, but with the timestamp given being the time of the last reading it had been able to take. The readings were reported as invalid.

The smartphone stopped reporting positions entirely. The software used has no means for reporting that a position is invalid, and this being the case it seems the best solution — although it does leave the user unknowing as to whether the tracker is out of GPS range, out of mobile phone reception, or turned off entirely.

5.5.12.6 Stationary

GPS trackers tend to give more accurate positions when moving than when stationary. When the Durham tracker is still, for example, it will often report positions over a large area; as soon as it is carried away it will give readings which are very close to the actual positions of the tracker (other effects such as multipath aside).

The reason for this is that the SiRFstarIII chip used in the Telit module takes the Doppler effect into account when the unit is moving. The GPS satellites are all moving at a high speed in their orbits, and there is a discernable Doppler shift in the signals being received from them. This shift is caused by the movement of the satellite (known), the rotation of the Earth (known), and the movement of the tracker. The Doppler shift can be measured with a far greater accuracy than the tracker position, as it will not be susceptible to things like multipath effects. This means that the positions of GPS readings — which would normally give a much higher error — can be smoothed by taking into account the more accurately known velocity when the tracker is moving at higher speeds.

The SiRFstarIII chip contains two further, optional filters that are not used by the Durham tracker but which may be used by other trackers. The first — 'Track Smoothing' — extrapolates future positions from previous ones, which greatly reduces the error when travelling in a relatively straight line (such as in a car), but causes the recorded position to continue moving for a short while after the tracker has come to a halt. The second — 'Static Navigation' — will freeze the tracker's position when the measured velocity drops below 1m/s. The recorded position will remain static until either the velocity rises approx. 10% above this, or the position reports an approx. 50m change. Both of these filters work better in vehicles such as cars than in situations with slower movement or movement in often changing directions.

Three experiments involved the trackers being stationary — outside, in a car, and inside near a window.

Outside For the outside experiment all three trackers had a good view of the sky, although they did not have a view to the horizon. The smartphone software does

not report the number of satellites used to calculate its positions; of the other two, the Durham tracker reported 6–9 satellites, mainly 8, and the TR-102 reported 3–5, mainly 5. All three trackers gave a range of points which were centred about the actual location. The Durham tracker and the smartphone both behaved similarly; their readings occurred regularly throughout the three hour period, and the positions obtained were both within a similar area (see Fig 5.12). The TR-102 was less reliable, however. It didn't respond consistently (spending long periods where it was unable to get a valid GPS reading), and the positions it did obtain covered a far greater area.

Ignoring the first few minutes of the experiment, where the errors were unusually high, the TR-102 showed errors of up to 270m. It also did not report consistently; there is a large period in the middle of this experiment where only a handful of readings were taken. The errors on this tracker are extremely high; for the latter part of the experiment the majority of errors were under 20m, but there were still a great number which were above this.

Both the smartphone and the Durham tracker have a much more consistent position. The error — while not static throughout — remains relatively constant for long periods. Oddly, for both these trackers, the error changes at about the same point. At around 1 hour 45 minutes the error increases from around 1.5m (Durham tracker) / 1.5–3m (OsmAnd) to approx. 5.5m.

Inside For the inside experiment the trackers had a much more limited view of the sky. The TR-102 reported 3–4 satellites, while the Durham tracker reported 3–9, averaging around 7. As above, the points measured by the trackers were centred around the correct position, but again the positions from the TR-102 covered a much larger area. The errors can be seen in Figure 5.13.

Again ignoring the first few minutes, the TR-102 gives much larger errors than the other two trackers. The errors go up to almost 50m, and for the most part are reasonably evenly spaced between 10 and 40m. It is also notable that the TR-102 reports extremely infrequently, with large gaps in reporting. Its time to first fix was also extremely high, and has been for all experiments run.

The smartphone has, to begin with, an almost constant position. This may be

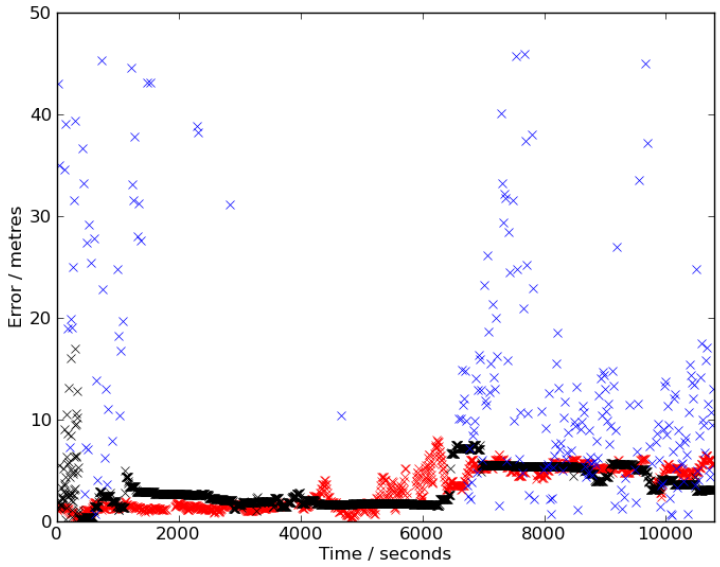


Figure 5.12: Positional error from three trackers lying stationary outdoors. The graph is cropped to errors below 50m. Errors are based on a central position taken from Google Maps, which may introduce a small error of a couple of metres. The three trackers are the Durham Tracker (red), the TR-102 (blue), and the smartphone running OsmAnd (black).

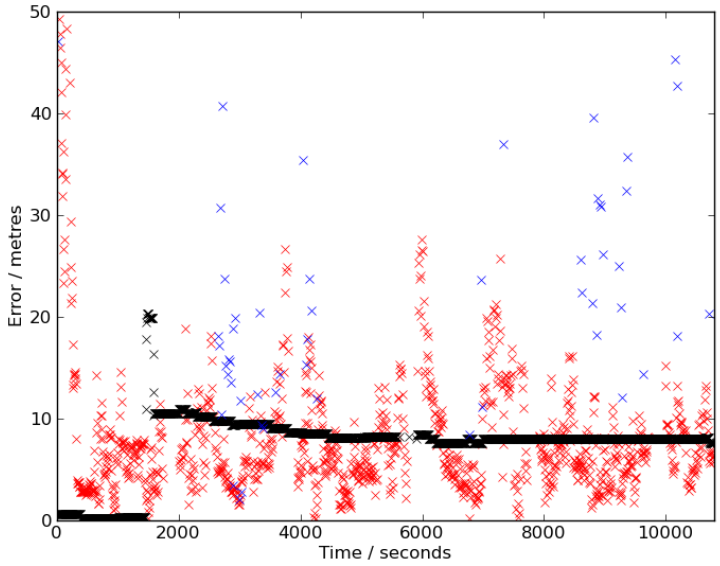


Figure 5.13: Positional error from three stationary trackers indoors, lying 1m away from a window. The graph is cropped to errors below 50m. Errors are based on a central position taken from Google Maps, which may introduce a small error of a couple of metres. The three trackers are the Durham Tracker (red), the TR-102 (blue), and the smartphone running OsmAnd (black).

due to it locking into a position and continuing to report this position while speed is low. However, after a short period while the error was extremely low, the reported position error rises to 10m.

The error on the positions reported by the Durham Tracker varies between 0 and 30m, with the majority of points between 0 and 10m.

To ascertain whether the poor behaviour of the TR-102 was a one off problem this experiment was rerun on a second occasion, with just the TR-102. On this occasion it gave a considerably improved response for the first hour of operation — reporting positions regularly and accurately, with an average time between readings of 12.71s. However, after this first hour it stopped responding entirely for the remainder of the three hour period.

This shows that the TR-102 has the capability to perform to a similar standard as the other two trackers — but on this and many of the other experiments it has failed to do so. It appears to be somewhat unreliable.

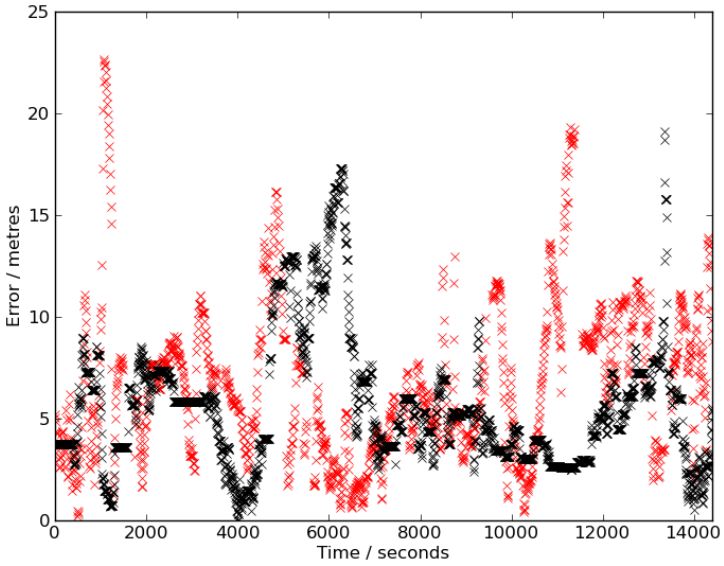


Figure 5.14: Positional error from two trackers in a stationary car. Errors are based on a central position taken from Google Maps, which may introduce a small error of a couple of metres. The two trackers are the Durham Tracker (red) and the smartphone running OsmAnd (black). The TR-102 failed to report a valid position during this experiment.

In a car The three trackers were left in a car overnight. Unfortunately the TR-102 failed to report any valid positions during this experiment, so the only data obtained is for the Durham Tracker and the smartphone running OsmAnd. The experiment lasted four hours, after which time the smartphone ran out of battery. The results can be seen in Figure 5.14.

The results for this experiment are very similar to the two previous stationary experiments. The errors for both trackers range from 0–25m, with the majority of points under 10m. The errors for both drift over time, rather than jumping around. Of the two, the smartphone gives slightly better results. Both give a regular set of readings, without any breaks.

5.5.12.7 Size, Weight and Cost

A tracker that is too large or too heavy to use, or which is too costly to buy, will not be of practical use. It therefore needs to be sufficiently small and light to be — depending on the context — carried around by a person, attached to a UAV, attached to an item, or placed in a car.

Table 5.5 lists the size, weight and cost of the various trackers described above. The first three are the trackers being directly compared; the remainder were described in Section 5.5. The last two sections are model aircraft trackers and animal trackers respectively, and may not be directly comparable — see Sections 5.5.9 and 5.5.10. For several of the trackers listed some information was unavailable.

Tracker	Width / mm	Height / mm	Depth / mm	Volume / cm^3	Weight / g	Cost / £[19]
Durham Tracker (boxed)[20]	130	100	30	390	185[21]	-
Durham Tracker (unboxed)[22]	85	67	14	79.7	65	-
TR-102	115	45	22.5	116.4	102	167.99
OsmAnd	83	50	16	66.4	88[23]	10/mo[24]
OsmAnd — HTC Sensation	126.1	65.4	11.3	93.2	148[25]	-
SPOT Personal Tracker	111	69	44	337.0	209	>63.07[26]
Globalstar SmartONE	165	82.5	25.5	347.1	385	-
MeiTrack MVT-600	103	98	32	323.0	220	-
MeiTrack VT-400	123	83	37	377.7	350	-
CelloTrack Solar	585	130	30	2,281.5	-	-
ATrack AY5i	100	65	26	169.0	161	-
TinyTelemetry	55	15	10	8.25	12	85.12
EagleEyes	100	70	40	280	78	63.07
TinyTrak SMT	25.5	23.5	4.1	2.5[27]	-	30.27
Vectronic's GPS Plus Telemetry Collar	-	-	-	-	100	-
Quantum 4000 Advanced GPS Collars	-	-	-	-	30	-

Table 5.5: Comparison of sizes and weights for a variety of trackers. Not all data is available for all trackers.

[19] All currencies have been converted to GBP.
[20] The current box design could be reduced in size for a future version.
[21] 185g without 4xAA batteries; 341g with.
[22] This does not include battery or aerials.
[23] Smartphone size and weight based on Sony Ericsson X10 Mini.
[24] The app is free; smartphones can cost under £10/month, but prices vary enormously.
[25] The experiments were taken using an HTC Sensation.
[26] £63.07 plus subscription.
[27] A separate power supply, radio and serial based GPS receiver are also required, which will take additional space.

5.5.12.8 Map quality

The user will want to be able to see where the tracker is, and may want to take in other information (such as historical tracks). This is a comparison between the Durham Tracker website and some of the other options — either as supplied with one of the trackers tested, or a generic tracking website as described in Section 5.5.11.

The servers compared below are:

EZFinder is a web based tracking server produced by Globalsat (see Section 5.5.11)

GPSGate is a generic tracker server which supports a very large variety of tracker types and manufacturers (see Section 5.5.11)

Greenalp is a company which produces the Greenalp Real Time GPS Tracker (see Section 5.5.2.3) and the associated tracking website

TR-102 Call Centre and

TR Management Centre are both provided by Globalsat for their TR-102 tracker (see Section 5.5.1)

TrackingMate is a web based tracking server produced by Meitrack (see Section 5.5.11)

TrackingServices is the Durham Tracker Server (`http://trackingservices.co.uk/`), described in Section 5.4.2.9

WebtraQ is a web based tracking server produced by TraqLogic (see Section 5.5.11)

One of the most notable features of most of the pieces of tracker software is a prominent map, showing the current or historical positions of the tracker or trackers. In the case of the Durham TrackingServices website this map takes up the entirety of the page, with extra data shown as overlays; for all of the others tested (with the exception of the 'TR-102 Call Centre', the software that initially came with the TR-102, but not the newer 'TR Management Centre' that replaced it) the map appeared in a smaller window with data around it. Some (most notably Greenalp) have an option for a fullscreen map which removes the other controls. Interestingly

the tracking servers almost exclusively use the Google Maps API, with some offering Open Streetmap or Bing as alternatives.

The user is likely to not only want to know where the tracker is but also something about its situation — any additional information that it has either transmitted back along with its position (e.g. battery level) or that the server has been able to calculate itself (e.g. times when positions have been received). The TR-102 Call Centre gave a full listing of information obtained in a table, taking up most of the screen. As mentioned before, most (TR Management Centre, GPSGate, WebtraQ, EZFinder, TrackingMate) list the information in tables beneath or to the side of the map. The Durham server lists it as an overlay. The Greenalp website does not list additional information, mainly because the single tracker it is designed for does not send any information back save for its position. The one exception to this is messages — the Greenalp tracking software allows for short messages to be sent between the tracker and the website, and these appear both as bubbles above the current location and in an overlay to the map.

Many people are interested in tracking not just a single tracker but a range of them, potentially of different types. All of the servers tested with the exception of GPSGate, WebtraQ and TrackingServices were limited to a single manufacturer's trackers. GPSGate supports a very wide range of different trackers; TrackingServices supports Durham Trackers, TR-102, and OsmAnd, and can easily be extended to support others; WebtraQ supports trackers produced by Globalsat (TraqLogic is a subsidiary of Globalsat), and is able to integrate with other trackers on request.

Tracking server software can be run in one of three situations. The first is for the company to run the software on their server, with the users logging in to view their trackers' positions — either for free or paying a subscription. Greenalp, GPSGate, TrackingServices, EZFinder, WebtraQ and TrackingMate are all examples of this — it allows the user to have everything necessary for following a tracker set up instantly without any of the problems involved in running their own server. It does, however, lead to other difficulties. It is unusual for it to be free, as it uses up limited resources. There are also data protection issues — the information being stored on the remote servers is not under the user's control, and if the company were to stop

running the server the user would have to start again from scratch.[28] For these reasons several of the options can be downloaded to run on a user's own machine—either a personal computer or a server. Both servers for the TR-102 run exclusively on personal machines, while GPSGate[29] and WebtraQ offer versions of their software which users can download to run on their own servers.

Discounting TrackingServices, as it is maintained by the Durham Tracker Project, the majority of the tracker servers charge for their use or download. The exceptions are Greenalp and the servers from Globalsat — the TR Management Centre, TR-102 Call Centre and EZFinder website, all of which are tied to their manufacturer's trackers. GPSGate offers their downloadable version for free, licenced for up to five trackers.

In summary, the TrackingServices website run as part of the Durham Tracker project offers a variation on the user interfaces used by other tracking websites, with the same or greater level of functionality. As it is maintained by this project it is, for our uses, both entirely free and fully customisable; for the same reason there is also protection from the problems associated with having a server hosted on a different company's server.

5.5.13 Conclusion

The Durham tracker appears to work as well as the alternatives compared against (particularly OsmAnd). In some cases it is slightly better; in some slightly worse; but overall there is not a vast amount to differentiate. This is in a great many ways a success. The tracker uses a commercial GPS module, and as such should not show a significantly better positional quality; nor should it be hugely more robust. That being said, the Durham Tracker has shown itself to give considerably more accurate readings than the TR-102, and has proved itself robust in that it has reported consistently throughout all experiments while the TR-102 did not (the smartphone running OsmAnd also stopped responding for part of one experiment, but this is

[28] Globalsat previously ran a tracking server (GS-TRAQ) in collaboration with a third party, and had difficulties with their users' data when the third party ended this collaboration and the server hardware failed.

[29] Under their 'No Cloud in the GPSGate Sky' policy.

likely to be due to other causes than the tracking software). The battery life on the Durham Tracker has also proved itself to be much longer than the other trackers.

The main benefit of this work is to have created a platform for future positioning experiments in general and a tracking system for a UAV in particular. The Durham Tracker can now be easily interfaced with the UAV's controller to give positional updates, and can be used as a communications device to take status information from the controller and report it to a remote user — as well as reporting back any commands from that user, allowing for a multi-level control strategy as described in Section 4.1.1.

As a standalone device the Durham Tracker is now sufficiently robust to be used in other search and rescue based situations — either being carried by walkers[30] to allow friends and family to locate them if they go missing, or in a SAR context being carried by the searchers to allow their routes to be plotted. This would mean that the people managing the search would be able to tell exactly where had been covered, and where still needed searching, without having the risk of searchers reporting their routes incorrectly (e.g. in the dark not travelling as far as they had thought).

To summarise, the Durham Tracker project is now substantially improved from its situation when taken over for this PhD — both in terms of usability and reliability — and shows itself to be not only as good as comparable commercially available trackers but also a suitable platform for use with a UAV.

GPS is commonly used for calculating positions in terms of longitude and latitude. It can also be used to measure altitude — although the errors for this tend to be considerably larger. It is important to be able to tell what sort of errors there are in positional measurements, particularly if precision is important (such as when formation flying), but also in general — if a UAVs altitude measurement is sufficiently bad, for example, then it will fly into the ground. The next chapter will discuss some of the errors inherent with GPS in general and with GPS altitude measurements in particular.

[30] See Section 5.4.3.3 for an example of the Durham Tracker being taken on a mountain walk.

5.6 Future changes

While the majority of problems with the trackers have now been fixed there are still some minor ones outstanding. There are also numerous additions that could be made to improve them further. Some of these are listed below.

5.6.1 Balance check

The Durham Tracker uses the Vodafone network, and the correct means to find the balance for a given SIM card is to send to the network the code '*#1345#'. The network then responds with a number, which is the amount of money left for that SIM.

The results from this are currently very erratic. In some situations it responds correctly, with the current balance. In others, it returns a generic 'OK' (signifying that the command has not produced an 'ERROR'), but no value. On some occasions the tracker crashes while checking for a balance. It is not obvious why this last occurs, but it is a serious flaw (and as such the balance check has been removed from the code).

The Telit has two options for sending codes to the network. The first is to send it as an unsolicited request — this command (AT#CUSD) is used for sending queries to the network. The second is to dial it as if it were a telephone number, using the ATD command. Oddly, some trackers work with one method but not the other, and vice versa.

5.6.2 Audio

The Telits have several functions which are not currently used. One of these is the ability to make and receive telephone calls. For the most part this is not of use for a tracker, but there are several situations where it might be advantageous to implement it.

A search and rescue UAV, when it finds a missing person, can report their location back to base — but that isn't going to give any information to the missing person, who may well be somewhat surprised if not scared by the UAV. If the trackers

were able to make telephone calls then they could automatically phone back to base when someone was found, allowing an operator to talk to the person, reassure them, and find out any useful information (such as any injuries they might have).

This could further be used as part of a search strategy; using a larger speaker, calling the name of the missing person and listening for a response.

If, alternatively, the trackers were being used to track vehicles then the telephone capabilities may form part of an anti-theft device. If the tracker believed that the car was being stolen (GPS position moves outside of normal working hours, say) then it could telephone its owner and allow them to listen to what was happening in the car.

5.6.3 On-tracker user interface

The trackers currently have five LEDs, which give information about the tracker status (see Section 5.4.2.6). These tell the user information such as when the tracker is taking readings or uploading, and to a small extent when the tracker is running out of battery. It is unable to give more detailed information, such as the current battery level, the number of satellites visible, how long until the next upload, whether the last upload was successful, etc. It is also unable to give an error status, such as that there is no balance, that one of the aerials isn't connected, that there's no SIM inserted, or that it's out of range for GPS or GSM.

One option for solving this would be to use the LEDs to output a binary code. The user could then look up the code in a table to find out what the tracker status is.

Another possibility would be to add an LCD display to the tracker, which would give a scrolling output of the tracker status. This could most simply be attached to the I^2C bus, which would mean that no PCB redesign would be necessary.

5.6.4 Memory card

The memory available on the Telits is very limited. While it can store data for relatively long periods between uploads, it has not the capacity for permanently storing historical points and cannot be out of mobile phone range for days without

ceasing to function. It is also necessary to limit the amount of debugging information stored on the tracker (see Section 5.4.2.3).

A more serious problem is that there have been occasional problems in the past with Telit modules that have stopped working entirely. They stop responding correctly to commands (occasionally responding, but more often returning 'ERROR'), and all data on them (including the program code) is wiped. This is extremely infrequent, and the root cause is uncertain, but it seems likely that it is related to the flash memory in the Telit modules having a limited number of write cycles. The Telit's internal memory appears to be primarily intended as a place to store program code, which in theory should be written only once (and read many times).[31]

One solution to both of these difficulties would be to add a removable memory card to the tracker. This would also have added advantages, such as being able to quickly extract debugging data and replacing it with a blank card. It would also enable keeping data that the end user can access (configuration files, data files, etc.) separate from the program code stored on the module's internal memory.

5.6.5 Multi sockets

One problem with the trackers storing points and uploading in bulk is that the user is unable to see where the tracker is in the intervening time. For many applications this does not cause difficulties — if the user is only going to check the tracker's position daily, say, then having an instant position is unimportant. However, in many cases (including, potentially, tracking a moving UAV) a way of finding the current position would be useful. The changes made to allow for instructions to be sent to the tracker via SMS ordering it to report immediately solve this to some extent, but do not offer a constantly updating position.

One of the reasons why the tracker does not upload its position every time a reading is taken is that the overheads for connecting to the server and uploading positions is too high. It takes a long time to set the connection up, and during that time no GPS readings can be taken.

[31] The limited write cycles of the GM862-GPS are noted by others — one manual includes the warning: 'NVM [non volatile memory] has limited write cycles. Limit is about 100000 times. Telit do not recommend to use NVM for data storage.'[170]

The Telit supports use of multiple sockets, including suspending and reopening sockets. As the current server receives the data via an HTTP POST, it does not lend itself to keeping a socket open for future data to be sent. It would not be too difficult to write a custom server that listened on a port and received the data as it arrived. The time for connecting would still be high, but if the tracker remained in GPRS range then it would be not need repeating. The time for sending individual points would be relatively low; it would reduce the maximum speed of taking GPS readings, but not by a particularly large amount.

In order to test how viable this is it was decided to connect to an existing server and see how well the multiple socket functions work on the Telit. The server selected was an IRC server.

IRC is a protocol for sending messages across a network. Users connect to the server, transmit some basic information (the main piece of information being the name the user is using), and then can join 'channels' (groups of users which will all see messages sent to that channel).

The main advantage of using IRC in this instance is that the protocol is very simple. Once a connection to the server has been made, only two commands need to be sent in order to allow the tracker to start sending messages.

This was attempted, but met with only limited success. The tracker was able to connect to the server and send its position, but several of the multisocket commands did not function correctly with the version of the firmware being used on the trackers, and in particular the program had great difficulties in recognising the escape sequence for suspending the socket. There were also problems with Vodafone, the provider of the mobile network being used, which often blocked connections to ports other than Port 80 (used for HTTP). While Vodafone do not officially block any ports, all connections to other ports timed out, while other networks had no difficulty connecting.[32]

This would be a useful function to get working. It would mean that the map

[32]This limitation to port 80 does not entirely stop most internet services — including IRC — from being used, but does cause some difficulties when prototyping. It stops many existing servers (not, as standard, using port 80) from being connected to, and prevents multiple services from being hosted on the same server without binding an additional IP address for each service. More critical was the failure of some of the multisocket commands, such as the suspension escape sequence.

was updated in real time while the tracker had reception, and would otherwise store positions until it was back in range and upload in bulk.

If multiple sockets could be used then other functionality could be opened up. The trackers could communicate between themselves, for example — making a connection between two trackers and sending information when necessary. It could also be used to allow the trackers to transmit positions to multiple media, for example (as in this case) to both the website and to an IRC channel.

It requires more work, however, to reach this stage.

5.6.6 Real time commands

If the server kept track of the IP of the trackers connecting to it then it would be possible to open a connection back to the tracker on demand. The trackers could then listen for a connection on a set port, and if one was received could receive commands. These commands could range from instructing the tracker to upload its data immediately, to changing configuration settings such as the frequency of taking readings.

5.6.7 Split files for data storage

In some cases, if a tracker is out of range of GPRS for a particularly long period, the file containing the points to be uploaded to the server becomes so large that it takes a very long time to upload all the points. In extreme cases it takes so long to upload that system failure occurs — usually the capacitor running down while high power draw is still required, but in other cases the socket dying before all data has been uploaded. If the file was split into smaller files when the main file became larger than a certain size (in a similar way to the debug logs, described in Section 5.4.2.3) then each file could be uploaded individually, reducing the time taken for each. This wouldn't normally be necessary, but could be useful in certain cases.

5.6.8 File size

When uploading a file over HTTP POST it is necessary to tell the server the amount of data being transferred before transfer starts. This is currently achieved by reading

the data file into memory twice — once to measure the file size, and once to upload to the server. If a different way of obtaining the file size could be found[33] then it would significantly reduce the time taken uploading to the server (see Figure 5.6), reducing the time when no GPS readings could be taken.

5.6.9 Relative Positioning

If the trackers are to be used in a swarm of collaborative UAVs in a search and rescue context, it is entirely possible that for much of the time some or all of the trackers will be out of GPS range. If the trackers were able to work out relative positions between each other then the UAVs would still be able to function as a group even without knowing their exact GPS position. If some of the swarm was in GPS range the rest could calculate their global position from them. This is further explored in Chapter 7.

[33]Note that the limited OS of the Telit module does not have many commands available in other OSes, including things such as 'stat'

Chapter 6

GPS Altitude Errors

6.1 Background

There are several localisation systems which are based on a constellation of satellites; by far the most common of these is the American Navstar Global Positioning System (GPS). Other Global Navigation Satellite Systems (GNSS) include the European Galileo, the Russian GLONASS, and the Chinese Beidou (or Compass) systems, but none of these (with the recent exception of GLONASS) currently has full global coverage.[1] GPS, then, remains at the forefront.

GPS allows for two methods of localisation — one for civilian use (the Standard Positioning Service or SPS), and one for US military use (the Precise Positioning Service or PPS). For the remainder of this thesis, use of the SPS will be assumed unless the PPS is explicitly stated.

GPS positions are susceptible to errors from a range of sources. Originally, errors were systematically added to the SPS signal by the US military; this was switched off in mid-2000, improving the accuracy of civilian GPS dramatically. The systematic errors were called Selective Availability (SA), and were specified to degrade accuracy to 100m horizontally and 156m vertically.[171]

Other sources of error include ionospheric effects, ephemeris (satellite position)

[1] Galileo is due to reach full operational capability in 2014 and the Beidou system is due to reach global coverage by 2020; GLONASS regained its full constellation in October 2011, but as yet has little commercial use.

and satellite clock errors, multipath distortion, tropospheric effects, relativistic effects,[2] and interference caused by natural or artificial means (such as jamming). These combine to a root sum square User Equivalent Range Error of ∼19.2m.[171]

To mitigate these errors it is possible to augment GPS. There are several methods for doing this; common ones include DGPS (Differential GPS, also known as a Wide Area Augmentation System or WAAS, where fixed bases transmit the difference between their known position and their received position, the theory being that many error sources will be constant in a small area), A-GPS (Assisted GPS, which downloads ephemeris and almanac data from an external source), and RTK (Real Time Kinematic, which correlates the carrier phase as received by a base station against that by a mobile unit as well as the GPS signal to give a highly accurate relative position).

The GPS system is designed primarily to give an accurate latitude and longitude, and it does this to a reasonable accuracy. Measuring altitude by GPS, however, is in general considerably less accurate — usually by a factor of two or more.[172] There are many reasons for this. The geometry of the satellite constellations is — by necessity — suboptimal for calculating vertical positions. The perfect condition would be to calculate a position from four satellites which were arranged in a tetrahedron; this is impossible to achieve in practice, however, as the Earth's surface blocks out any satellites currently below the horizon. GPS units will also avoid using satellites near to the horizon, as the atmospheric effects will be considerably greater. Working with only the satellites that are visible has little effect on the horizontal accuracy, but reduces the vertical accuracy.

6.2 Introduction to the problem

The US government provides a GPS SPS Performance Standard[173] which specifies the error which can be expected from a GPS receiver based on various criteria. The data included allows for the calculation of an average and a worst case error.

[2]The relativistic effects — effective clock changes between the satellite and receiver caused by the high speed of the GPS satellites, by the change in gravitational field between the satellite and receiver, and by the Sagnac effect — are mostly compensated for, but may cause minor errors depending on situation.

However, the errors taken into account are those which are caused by the GPS system itself — it does not include 'any error source that is not under direct control of the Space Segment or Control Segment', meaning that errors detected by a receiver are liable to be greater than those calculated.

A better source for governmental statistics on GPS accuracy is the Federal Aviation Authority (FAA), which publishes a quarterly report into the accuracy of the Standard Positioning Service.[174] This includes a graph of vertical position error from a network of fixed, high quality receivers making up a Wide Area Augmentation System (WAAS). A WAAS works in the same way as DGPS, so the receivers are designed specifically to calculate the error in the received position.

While the FAA WAAS data gives an indication of how GPS receivers are liable to behave in real situations, the receivers that they use are going to have a higher chance of producing an accurate fix than most handheld GPS receivers. This is because they are designed to only have errors based on conditions that would effect any receiver in the local area, while a handheld receiver would have additional problems caused (for example) by multipath errors or reduced signal strength.

It would be useful to be able to calculate the probability of vertical error for a standard, commercially available GPS receiver in a range of normal working conditions. This would allow the user to know how trustworthy GPS altitude actually is.

6.3 OpenStreetMap

OpenStreetMap (OSM) is an online collaborative mapping website.[3] The basic premise is to produce a map which is free from copyright. The predominant method of doing this is for people to upload GPS traces from routes they have travelled, which are then post processed by volunteers to produce the layout on the map.

The data used comes from a variety of sources. Some has been obtained from copyright free mapping sources, but the vast majority is taken from GPS traces from walkers, bikers, drivers, etc. The raw data — longitude and latitude — is shown, and volunteers then convert these raw points into roads, buildings, and other features.

[3] `http://www.openstreetmap.org/`

OSM only bases their map on the longitude and latitude; all altitude data is stripped. To calculate elevation the website uses the freely available SRTM dataset (NASA's Shuttle Radar Topography Mission, discussed below in section 6.4).

Most of the raw GPS traces are publicly available in the common GPX format, and these raw traces still contain the elevations. There are four alternatives for elevation in the traces. The first is that the GPS unit did not take elevation data. The second and third are that the elevation was calculated from the GPS signal. This may then be adjusted to allow for a local geoid (see Appendix G) or may be relative to the WGS84 ellipsoid. The fourth — in more advanced GPS units — is that the altitude was calculated not by GPS, but with a separate altimeter; a barometric one, for example.

Only one of these can be used directly in the comparison of GPS altitudes; the case where GPS is used relative to the WGS84 ellipsoid. Of the others, if no altitude is given then less data is available but no incorrect data is obtained. If the altitude is calculated from a separate altimeter then incorrect data will be being used, which will dirty the dataset as it will contain data that has not been obtained from GPS. However, it is likely that these will be in the minority, as for the most part only high end GPS receivers contain external altimeters. If the GPS altitude has been adjusted to allow for a local geoid then the GPX file will contain a '<GEOIDHEIGHT>' value which gives the offset from the WGS84 ellipsoid. It is a failing of this experiment that this is not taken into account; however, it is believed that the proportion of data which is affected is low, and future experiments can allow for this (see Section 6.5.5 for more).

The raw GPS traces give a large pool of GPS altitudes to compare. As of 14th July 2011 there were 2,413,056,358 points included, and this number constantly increasing.[4] Not all of these will produce an altitude, as not all GPS units report the altitude, and not all traces are publicly available.[5]

In order to analyse this altitude data there needs to be something against which to compare it; in this case, the Shuttle Radar Topography Mission.

[4] `http://www.openstreetmap.org/stats/data_stats.html`

[5] A rough estimate based on collected data gives approx. 50% altitude availability

6.4 Shuttle Radar Topography Mission

NASA's Shuttle Radar Topography Mission (SRTM) launched in February 2000 and over the course of its 11 day flight collected height data from the entire globe using radar interferometry.[177] This data was then made freely and publicly available. The original data covered the globe at a resolution of 1" latitude by 1" longitude with the exception of areas above 50 degrees latitude where it is reduced to 1" latitude by 2" longitude (as the lines of longitude come together).[178] This 1" data is available for the US; for the rest of the globe only 3" data is provided, which corresponds to approximately $90m^2$ at the equator.

Using the SRTM data for comparison has several advantages. It gives consistent worldwide coverage, which means that any GPS elevation taken from OSM can be compared against the same data. It also is based on the WGS84 ellipsoid (projections, ellipsoids and geoids are discussed more fully in Appendix G). As GPS uses the same ellipsoid as its basis, this allows for a direct comparison without first having to convert positions. It is possible for GPS receivers to compensate for the global WGS84 ellipsoid being a poor choice over a smaller area by converting data into a local projection or a globally calculated geoid such as EGM96 (the Earth Geodetic Model) — this would introduce errors into the data, but the majority of receivers are unlikely to have the ability to make this correction.

6.5 Additional sources of error

Although the SRTM data is being used as a base against which to compare the elevations obtained from OpenStreetMap, it is not itself entirely without error. The data was obtained using radar interferometry from a shuttle at an altitude of 233km. The surface that it detected beneath was not necessarily bare earth; heavy treecover, for example, would appear to the radar as a higher surface. Further errors were introduced based on the angle at which it passed over features — overestimating the height of slopes angled in one direction and underestimating that of those in the other.[179; 180] The data also contains some voids, predominantly in areas of high

relief. These, however, make up less than 0.2% of the total dataset.[6]

Despite this potential for error, SRTM data is specified to be accurate to within 16m,[177] and there are many papers which compare the SRTM data to other sources,[182; 183; 179; 180; 184; 185; 186; 187] several of which (although not all) have found it to be more accurate than this.

While this experiment is designed to calculate errors arising from GPS positioning, there are other potential sources of error in the GPS readings from OpenStreetMap which may affect the results of the experiment:

6.5.1 Altimeters

Some GPS receivers — predominantly ones which are at the upper end of the market — do not calculate their altitude from GPS, but instead rely on an internal altimeter. While this gives a more accurate reading for the user, it means that the data is useless for the purposes of this experiment. Some traces, therefore, may not be of use in this.

Barometric altimeters will tend to give a better altitude reading for the user, but are susceptible to external influences such as the weather (see Section 6.6.1).

6.5.2 Lat/Long errors

To compare the GPS calculated altitude against the SRTM data the assumption is made that both points share the same latitude and longitude. As, however, the latitude and longitude derived by the GPS receiver are themselves susceptible to error, the values being compared may vary. This is unlikely to cause major problems since the altitude is liable to remain relatively constant within any given small area.

6.5.3 Additional height

The GPS readings will not have been taken at ground level; instead they are liable to have been taken at the height of a car's dashboard, or a bicycle's handlebars, or a walker's rucksack. This will give an additional altitude to the GPS readings, which is liable to be constant during one trace but which may vary between traces. The

[6] Voids make up some 836,000km^2, which equates to 0.164% — Reuter et al. 181

error from this is unlikely to be more than a metre, however, which is the vertical resolution of the SRTM data.

It is assumed that as the data is being used to plot the route of land-based features (roads, buildings, etc.) it is unlikely for any readings to have been taken in conditions which are far from the ground (in aircraft, etc.).

6.5.4 Insufficient resolution

NASA SRTM data has a horizontal resolution of 90m.[7] All data in between is calculated through interpolation from nearby points — meaning that an area with sharply changing altitudes would give elevation readings that were inaccurate.

6.5.5 Geoid adjustment

As mentioned above, the experiments described in this paper do not take into account the Geoidheight field in GPX files, which means that if the GPS altitude has been adjusted to allow for a local geoid then there will be an unknown offset from the WGS84 ellipsoid.[8] It is believed, however, that this will only affect a small proportion of the data; in a smaller scale test only 6.0% of GPX files contained a Geoidheight field.

6.5.6 Dilution of Precision

GPS receivers are capable of reporting data on how precise they believe their data to be. While this is not technically a source of error, it would make for an interesting future experiment to gather data on the number of satellites used for each fix, and the Vertical Dilution of Precision (VDOP) — the GPS receiver's own metric for quantifying perceived vertical error, based on the range of possible altitudes from using more than the minimum number of satellites for a fix. This could then be compared against the OSM/SRTM difference.

[7] Vertical resolution is 1m.

[8] In future experiments this can be taken into account by subtracting the Geoidheight field from any data which has been adjusted.

6.6 Results

The altitudes taken from 20,140,623 GPS datapoints were compared against the altitudes given in the SRTM dataset. As the SRTM data is given to the nearest metre, the GPS data was likewise rounded to the nearest metre. The OSM GPS readings ranged from $-2,343$ to $20,000$m, and the SRTM readings ranged from -393 to $5,710$m.[9] Fig 6.1 shows the two datasets plotted against each other; as can be seen, most of the data follows the line of $x = y$ (where x is SRTM and y is OSM recorded altitudes). There are some interesting points to note. There is a line of points at 0m GPS reading ($y = 0$), presumably either from receivers which just output zero for any altitude, or receivers which have insufficient numbers of satellites to obtain a full 3D fix. As these don't occur for any points much over 1,000m the latter seems more probable. There is another set of readings along the line $x = -y$, presumably from receivers giving an unintentionally negative reading. There are many readings from low SRTM altitudes (mainly between 100–200m, with a smaller number between 350–600m) that report GPS altitudes of anything from 0 to 12,000m. There are several lines running parallel to $x = y$; these are potentially from units that have a fixed offset based on a different ellipsoid. There are also several areas some distance from $x = y$ which contain a surprisingly high number of readings, with notable clusters around (SRTM, GPS) = (100–200m, 9,900m), (60–600m, 10,900m) and (2,000m, 200m). It is possible — although unlikely (see Section 6.5.3) — that the first two of these clusters are from receivers in aircraft, some distance from the ground.

For the remainder, along the line of $x = y$, there is a considerably higher error rate at lower altitudes. There are several possible explanations for this. There is a greater chance of errors having an effect at lower altitudes; things like the multipath effects of high buildings, for example, are only going to be noticeable when at low altitudes and between high buildings, and at higher altitudes a receiver is more likely to have a clear view of the sky, right to the horizons. GPS receivers used when travelling at higher altitudes are more likely to be of a higher quality, which

[9] All max/min values are after removing extreme outliers (all points below $-3,000$m, which excludes any SRTM 'no data' values).

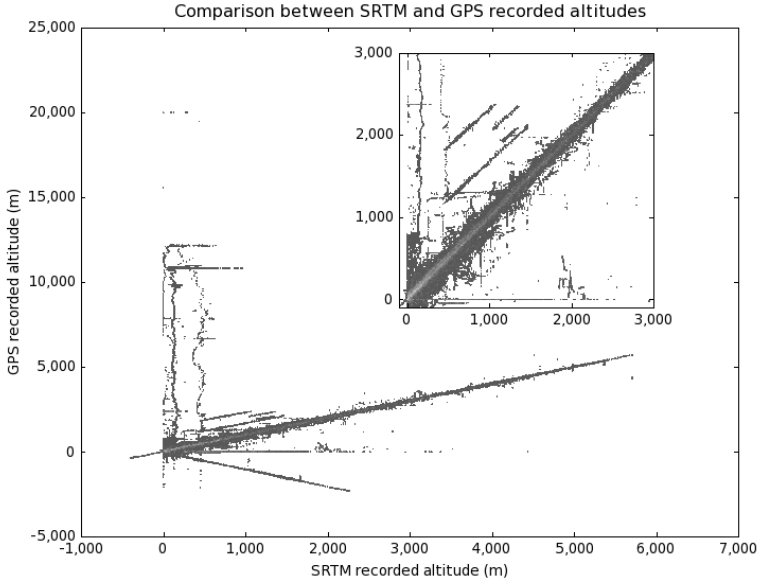


Figure 6.1: Comparison between SRTM and GPS recorded altitudes. Lighter grey areas represent areas of higher concentration, where there were greater numbers of GPS readings for this SRTM reported altitude. The inset graph shows the section between -100 and 3,000m

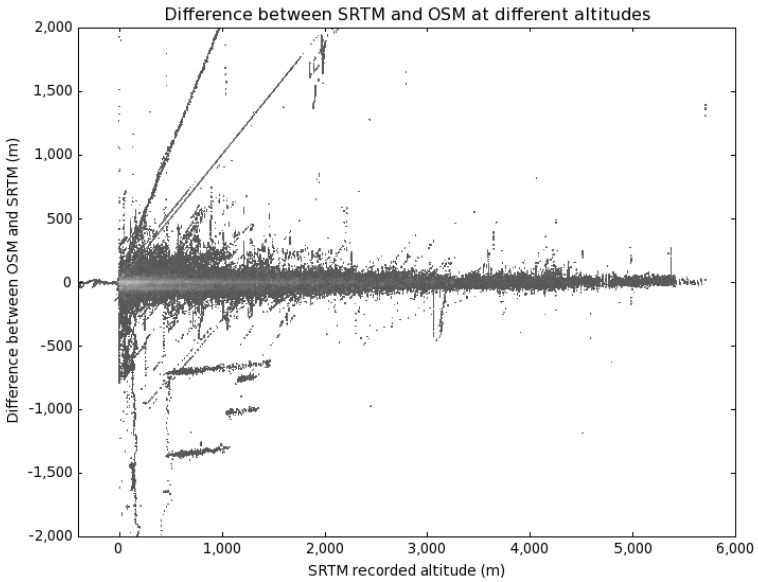


Figure 6.2: Difference between SRTM and OSM readings at different altitudes. Lighter grey areas represent areas of higher concentration, where there were greater numbers of GPS readings for this SRTM reported altitude. Graph cropped to differences between ±2,000m.

might give a better vertical accuracy.

Fig 6.2 shows SRTM altitudes against the SRTM/GPS altitude difference, cropped to differences between ±2,000m. Similarly to Fig 6.1 this shows that most of the errors occur at low SRTM altitudes, and that while the majority of differences are relatively low at all altitudes (as can be seen by the lighter line representing a high number of readings along $y = 0$) a not insignificant number of errors continue to a very high level — while the graph is cropped to ±2,000m, this continues all the way to differences of 12,000m.

Fig 6.3 (zoomed in to errors from ±100m) shows the GPS error curve. As can be seen, the majority of points have a zero error, and 73.26% of GPS readings lie within 20m of the SRTM data. Statistically, the curve has a mean of -7m, standard deviation of 273.05m, and median of -3m. The RMS error is surprisingly high, at 273.14m; if 2SD outliers are removed, however, this drops to 31.67m.

There are a few noteworthy aspects to this graph. The first is that there is a

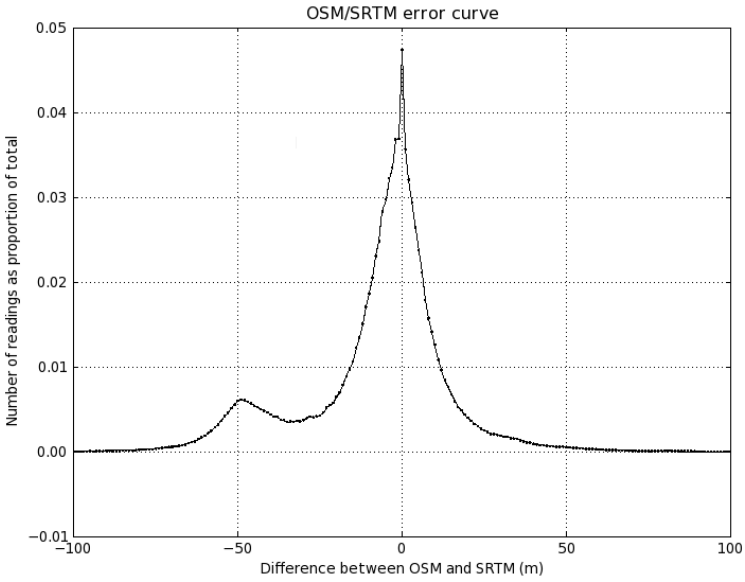


Figure 6.3: OSM / SRTM error curve, showing the proportionate number of readings for each error value. Graph cropped to errors between ±100m. 0.34% of readings lie below and 0.70% of readings lie above this range.

peak at -50m. This is also visible in the raw data in Figs 6.1 and 6.2, but is not as obvious when seen amongst the other erroneous data. The reason for this is unclear; certainly there must be a number of receivers which have a fixed error. One potential cause would be an intentional correction to allow for a local projection (see Appendix G); this would require a very large number of receivers to all have the same internal bias, however.

Another point to note is that as the errors are more pronounced on the negative side the GPS values have a tendency to be higher than the SRTM.

Looking at all the data, there is a very long tail; 7.95% of the data has a greater than 50m error, dropping to 1.05% with an error in excess of 100m. The large number of outliers explains why the mean is noticeably higher than the median, with such a high standard deviation.

6.6.1 Comparison with barometric data

A practical use case for a GPS altimeter would be hill climbing. If only altitudes between 0 and 1,000m are considered then it can be seen that the average absolute error is 24.2m. If this is compared to a barometric altimeter, where errors are caused by pressure changes, these errors would equate to a change in pressure of 173.4 Pa average absolute error.

A barometric altimeter calculates height based on the difference in pressure from sea level. However, as pressure is based not only on height but also weather conditions and temperature, barometric altimeters will tend to drift over time. Taking pressure data[10] from a weather station in Fife[188] it is possible to analyse the average pressure change over different time periods (see Fig. 6.4). Looking at the pressure change over a 5 hour period (5 hours being a not unreasonable length of time to be walking) there is an average pressure change of 173.5 Pa — this average is highly comparable to the average GPS error in the 0–1,000m range. As the length of time increases the average change in pressure also increases; at 24 hours the average error for a barometric altimeter would be 607.2 Pa, or 84.9m — considerably worse than GPS.

As the sources of error for barometric and GPS altimeters are different, each will behave better in different conditions. The accuracy of a barometric altimeter will be dependent on the weather; in a storm where pressure fluctuations are likely it is liable to give an inaccurate reading. The accuracy of a GPS altimeter will be more dependent on factors such as terrain and visibility of the sky.

Research exists on methods to fuse barometric and GPS altitude data,[189; 190; 191; 192] but a discussion of this falls outside the scope of this work.

6.6.2 Comparison with FAA data

As mentioned in section 6.2, the FAA produces a quarterly report on the accuracy of GPS. Part of this includes a graph of vertical error. Fig 6.5 shows this data alongside the errors between OSM GPS traces and SRTM. The FAA data is taken from April-June 2011, which corresponds well to the OSM data, taken from June-

[10] Data taken every minute from March 2006 to December 2011.

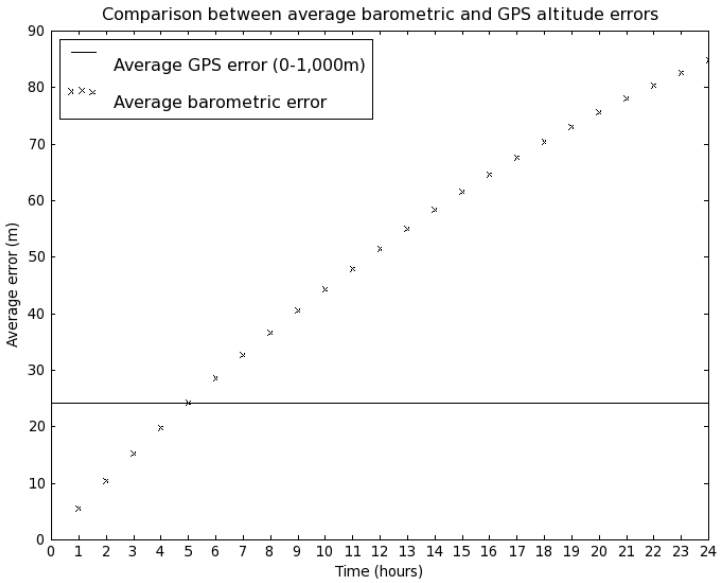


Figure 6.4: Barometric drift over time, converted into an estimated altitude error in metres. The horizontal line shows the overall average error for GPS readings between 0 and 1,000m, and does not take into account the length of time the GPS unit was running.

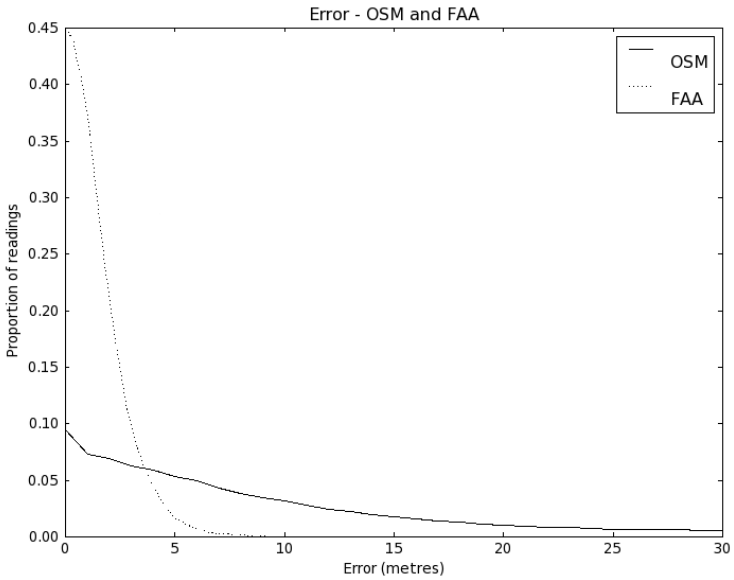


Figure 6.5: Error curves for OSM/SRTM and FAA. Both are given as a proportion of the total readings, to allow for comparison. The errors shown are the absolute difference between the GPS reading and the comparison data. Graph cropped from 0–30m.

July 2011. As the FAA graph shows absolute differences (i.e. no negative values), the OSM data in Fig 6.5 has been changed from Fig 6.3 to make all errors positive. As can be seen, the errors seen by a range of receivers in a range of conditions is far worse than those seen by well calibrated receivers in optimised conditions.

Part of this increase in error may be due to the errors inherent in the SRTM data; this is specified as being no more than 16m, however, and the difference is far greater than that. Fig 6.6 shows the two data sets — the FAA data and the OSM/SRTM difference — as cumulative percentages. Each point represents the percentage of readings which have an error of that value or better. As can be seen, the two graphs are very different — at the extreme, the 95% confidence level (that is, 95% of all values fall within this upper bound) for the FAA data is 4m while for the OSM/SRTM data it is 55m (just off the edge of the graph).

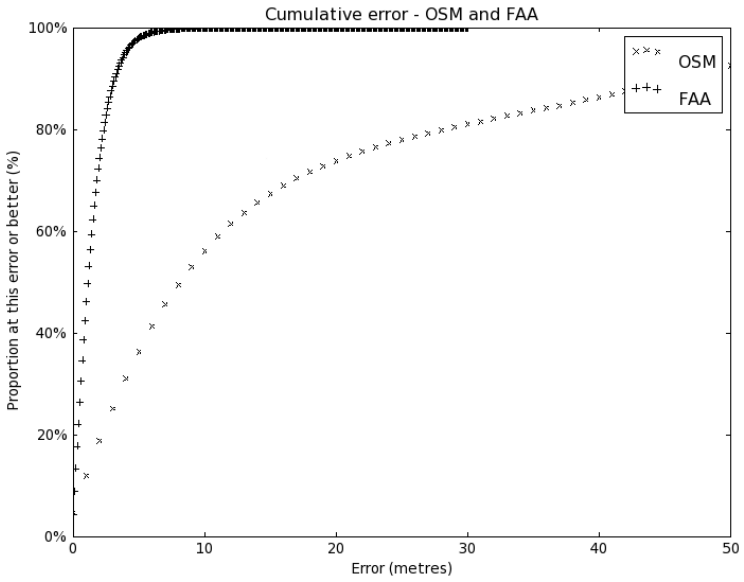


Figure 6.6: Cumulative error curves for OSM/SRTM and FAA. Values shown represent the percentage of readings which have that error or better. Graph cropped from 0–50m.

6.7 Conclusions

The US government publishes data on the accuracy of GPS altitude. However, for a generic commercially available receiver as might be used by the average person, the actual errors produced in any given situation are liable to be considerably worse than those specified. In order to gain an accurate altitude it is therefore necessary to use either a separate (non-GPS) altimeter or a high accuracy GPS receiver designed for surveying, in a carefully controlled environment.

Average GPS altitude errors are comparable to average barometric altitude errors after five hours' drift,[11] and may be more reliable in situations such as storms where pressure fluctuations are likely.

The raw traces from OpenStreetMap produce a useful resource of raw data that would make for interesting further research. In particular, a follow-up study comparing traces in a limited area against higher accuracy digital elevation models (DEMs)

[11] After five hours, GPS data is more likely to give a correct altitude

would allow for any errors imposed by use of SRTM to be eliminated.

The errors found are, for the most part, greater than the 16m errors specified by the SRTM dataset, but the SRTM error may yet account for a large proportion of the measured GPS inaccuracies.

Using the data from OpenStreetMap allows for a very large dataset. However, without details on the exact hardware used in readings (predominantly to remove those readings where the altitude data provided is not the raw GPS readings but instead uses either a separate altimeter or post processing to change the geoid), some of the measured error may be caused by influences other than those simply connected with GPS.

To conclude, this chapter has shown that the errors inherent in a GPS altitude measurement from a standard commercial GPS receiver in a range of standard cases are liable to be far in excess of the errors reported by the FAA using high quality receivers making up a WAAS, confirming that a large part of the altitude error is situational. Further research is required to fully quantify this in a more controlled environment.

GPS measurements — particularly altitudes — have associated errors, and are in some situations entirely unavailable. It would therefore be useful for a swarm of UAVs working in harsh environments (including in disaster areas, bad weather conditions, under heavy tree cover, in urban canyons, indoors, etc.) to have a secondary method of working out their locations — not necessarily positioning relative to the world, but positioning relative to each other. This is discussed in the next chapter.

Chapter 7

Relative Positioning

7.1 Introduction

In order to know where you are, you need to know what frame of reference you are using. In many contexts it is necessary to know a position relative to as large an area as possible — knowing where on the planet; where in the country; or where in a building or complex.

A group of GPS enabled devices, out in the open, will all be fully aware of their locations. These locations may contain errors (see Chapter 6), for example from multipath effects — but to some level of accuracy they will know where they are. They will know their positions relative to the GPS constellation, which in turn is tied to the WGS80 ellipsoid to give a location on the planet.

If these same devices were taken indoors, out of GPS range, this would no longer be the case. Out of range of the GPS satellites they would no longer be able to calculate their global position. However, while there are numerous situations where it is anything from useful to vital for a device to know where it is, this does not necessarily have to be on quite as large a scale as given by GPS. For patients in a hospital or stock in a warehouse it is only useful to know their position relative to the building they are in. A group of people or devices working collaboratively may not need to know their location relative to the outside world; only where they are relative to the others in their group.[193]

Similarly, the accuracy of localisation can be context specific. In some cases it

is vital to know a position with mm accuracy, while in others the only information required is whether an object is present in a given area (such as a room or building) or not.

This chapter will be focusing on relative positioning — that is, positioning relative to a group of movable devices, many or all of which will not have a known position.

Many of the techniques used in relative positioning — those used to calculate the distances between two points — are equally applicable to positioning relative to a fixed network, and have been used in numerous methods for calculating local positions in situations where larger scale systems such as GPS are unavailable.

This section will briefly outline in a wider scale some of the various methods for localisation,[194] with advantages and disadvantages for a number of contexts, with emphasis on relative positioning between collaborative UAVs where applicable. As localisation in a global context (GPS, LORAN, etc.) has been covered in Section 5.2, it will not be covered again in detail.

Many of the techniques described assume that all units are stationary — or at least, extremely slow-moving. This is usually a safe assumption, as long as the rate at which measurements are made is sufficiently high. There are, however, several reasons why you would want to transmit less often, and therefore reduce your measurement rate. These include the increase in power consumption while transmitting; the increase in congestion if a large number of units are trying to transmit at a high rate (which may also affect other, unconnected systems); and the risk in a noisy environment of having a high rate of packet loss (i.e. attempting to measure at a high rate, but only being able to measure at a low rate). Later in this section is described an experiment to ascertain the effect of speed and measurement interval.

Here, then, is an overview of some of the methods that have been developed. They are split roughly into groups, based on the technology they use.

7.1.1 Connectivity

The simplest method for working out where something is is by seeing whether it can be detected at all. If the range over which a sensor is able to detect something is known then simply by checking whether an object is detected or not will tell whether or not it is within that range.[195; 196] If a large number of sensors are used in a small area then the ranges of the sensors that cannot detect the object subtracted from the overlap of the ranges of the sensors that can will yield an area in which the object must be.

As with many of these methods, the roles of transmitter and receiver can be reversed with the method remaining the same.

This method is used by Nagpal et al.,[197] where units in a network calculate their position relative to anchors by calculating how many 'hops' away they are. If a unit is in range of the anchor it has a value of 1; the values then propagate outwards, such that a unit will have a value one higher than the minimum of its neighbours' values. If there are sufficient units in an area, and assuming a uniform range, this will equate to a distance at a resolution equal to the range of the units. Multiple anchors then allow for trilateration. The two main downsides to this idea are first that a large number of units are required for the distances to become accurate, and second that the range of the units must be constant.

Niculescu and Nath's Ad Hoc Positioning System[198] works (at its simplest) on a similar principle, but with the range of the units being calculated on the fly by dividing the number of hops between anchors by the known distance between them (as anchor positions are known, inter-anchor distances must also be known). The paper later discusses signal strength as an alternative to connectivity through hop counting, amongst others.

In some cases, the range of the sensors is large but easily blocked — ultrasound or infrared, for example, are unable to pass through walls. This allows for systems which have one receiver per room — if in range of that receiver, the detected object must be inside that room. Some examples of this are the Active Badge system,[199] using badges that transmit intermittent pulse width modulated infrared signals, and the WALRUS system,[200] which uses ultrasound pulses that are received by

standard computers (one per room).

While it allows for a relatively cheap and simple method of calculating positions, the main downside to a connectivity based method is that it requires a large number of units to be able to pinpoint to any degree of accuracy. With only two units (one transmitting, one receiving), for example, the most that can be discerned is whether or not one is within a certain distance of the other. Another problem is that the detected signals can be blocked or reflected by other objects, which can skew the calculated positioning.

7.1.2 Signal strength

A close method to connectivity is measured signal strength. A signal will reduce in strength as it propagates out from a source, with its strength at any given point being proportional to the distance as governed by the inverse square law. This means that in perfect conditions it is possible to calculate the distance to a transmitter simply from the strength of the received signal.

The downside to this is that it only works effectively in perfect conditions. While signal strength is a function of distance it is also a function of other factors, such as air conditions, the presence of objects (which might reflect, block or reduce signals), and any other transmitters. Even simple situations such as having a person walk between transmitter and receiver will drastically affect the perceived distance by reducing the received signal strength.

Nevertheless, despite these shortcomings there are many systems[201; 202; 198; 203; 204; 205] which are designed to use this method. They include the method described by Chintalapudi et al.[206], which uses received signal strength in a relative positioning context to calculate positions of fixed transmitters from a large number of mobile receivers without prior positioning data, using genetic algorithms.

7.1.2.1 Signal strength with pre-mapping (or 'fingerprinting')

One way in which to avoid some of the problems with the previous method is to carefully map signal strengths across an area. This bypasses the problems of (stationary) objects affecting the signal strength by measuring the signal strength at all

points. The signal strength from various fixed transmitters is then checked against a lookup table to give an exact position. Further improvements can also be made to partially allow for signal strength fluctuations.

There are numerous examples of this,[207; 208; 209; 210; 211; 212; 213] showing a marked improvement to using signal strength alone, but some of the problems with the pure signal strength method still remain. Mapping the signals from fixed transmitters will take into account fixed objects, but anything movable — anything from a filing cabinet being moved to the other side of a desk to a human walking through the room — will still cause the signal strengths to change. It also requires, unlike most methods, the whole area to be extensively surveyed before use. Not only must an infrastructure of transmitters be installed, but a computer must store the entire map of signal strengths.

This method has also been used on a far larger scale to obtain global positioning from a range of signals,[214] most notably using WiFi hotspots in a WiFi Positioning System (WPS),[215; 216] often as an implementation of the W3C Geolocation specification.[217] Some examples of companies offering WPS data are Skyhook Wireless, Navizon, Ekahau and Google.

A similar method is used by US Wireless's[1] RadioCamera system, where base stations listen for both transmitted signals from mobile units and the multipath signals caused by their reflection, and fingerprint the location of the unit from this.

7.1.3 Interference

Waves will interfere with each other. The received signal at a given point is based on the sum of the signals at that point — if two signals are received, one of which is positive and one of which is negative, they will cancel one another out (known as an 'antinode'). If they are both positive they will add (to produce a 'node'). As the knowledge of the transmitter positions, their signals and the receiver position will give the received signal, measuring the received signal will — if the transmitter positions and their signals are also known — give the position of the receiver. As the signals are cyclical this position will be regularly repeated, meaning that a

[1] Proprietary system; company filed for bankruptcy in 2001

globally unique position cannot be found, but systems do exist which use multiple transmitters and receivers to overcome this limitation.

One such system is the Radio Interferometric Positioning System (RIPS),[218] which uses the interference between two simultaneously transmitting units at slightly varying frequencies to calculate a sum of distances. Multiple measurements between a large number of units allows for point to point distances to be calculated, using an optimisation method based on genetic algorithms. Because the method uses interferometry instead of TDOA their processing time is far higher (the paper makes reference to 80 minutes of post processing, and real time positioning is impossible), but their average error is an unusually low (for RF based systems) 4cm over a maximum range of 170m. They also use COTS (Commercial Off The Shelf) hardware, solving the problems of requiring high precision time synchronisation and high sampling rates by respectively using the relative phase offset at two receivers (a function of the relative positions of the units; this still requires time synchronisation, but at a far lower precision), and by using close frequencies at the two transmitters (meaning the envelope on the interference pattern is at a low frequency).

Barber[219] describes an interferometry based system for keeping two ships a set distance apart when refueling at sea in storm conditions, designed in partnership with HMS Collingwood. A pair of transmitters on one ship produce an interference pattern. The two ships are positioned manually, and three orthogonally positioned receivers on the second ship then each detect when they cross a boundary between a node to an antinode (or vice versa), giving relative movement between the two ships. This shows how far the ships have drifted and allows for compensation to be applied to bring them back into alignment.

7.1.4 Angle of Arrival (AOA)

An alternative to measuring distances and using trilateration to calculate positions is to instead measure angles and then use triangulation.[220; 198] One method for measuring angles is to correlate the inputs from multiple receivers — a signal arriving at 90 degrees to the line between two receivers will arrive at both at the same time; as the angle changes one way or the other the signal will start to arrive sooner at

the corresponding receiver.

Systems which use this include the Acoustic ENSBox,[220] which uses four microphones to obtain angles in either 2D or 3D (not to triangulate between units, but instead to calculate the positions of noises to track other things) with average 2D position errors of 5cm.

Sen et al.[221] suggests a variation on this, where a user rotates themselves in order to calculate angles to fixed transmitters (wifi access points). As they rotate, their body will shield signals when pointing away from the access point. By analysing the signal strength to a range of access points at each angle during the rotation, a position can be triangulated between them.

Some systems exist[222; 223] which use both Angle of Arrival and Time Difference of Arrival (see below).

Another option which is somewhat different is to use lasers, which have a high transmission speed and travel in perfectly straight lines. The use cases for this are limited, as perfectly direct line of sight is required, but some systems use this. One example is the Metris[2] 'Indoor GPS',[224] which uses rotating infrared lasers in a fixed position (similar to a lighthouse) that send out two fan shaped planes at an angle to one another. The time between detection of the two planes gives the angle from the horizontal, and the time between an intermittent reference signal (transmitted at a known point in the rotation) and the detection of the rotating beam gives an angle from the vertical.

7.1.5 Time Difference of Arrival (TDOA)

As has been mentioned above, signals travel in air at an accurately known speed. Therefore if you transmit two signals simultaneously from different points — or transmit one signal and receive it at two different points — the difference between the two arrival times will tell you the difference in distances. This can then be used to calculate the exact distance between points, if enough measurements are taken.

There are two main types of signal which are commonly used — radio (RF) and ultrasound. Both have advantages and disadvantages. The big disadvantage with

[2]Now owned by Nikon Metrology.

radio is the timing accuracy required for calculating distances within a reasonable range. If you measure a time difference of arrival for two ultrasound signals, a timing resolution of 10 microseconds equates to a distance error of ∼3mm. For two RF signals this same timing resolution will translate to ∼3km, meaning that ultrasound allows for the use of far less precise equipment — there is a greater margin for error.

Ultrasound is not without its own disadvantages, however. It is unable to penetrate through solid objects, making it both difficult to use in environments without a clear field of view (such as indoors) as well as being more susceptible to multipath effects from reflecting off objects. Depending on the frequency selected ultrasound can be heard by a number of animals. It also tends to have a far greater problem with the transmitters being directional — that is, unable to transmit in all directions equally.

Examples of this include Cambridge's Active Bat system,[225; 226] which uses ultrasound transmitters on portable tags with a network of fixed receivers to obtain a 3D accuracy of around 3cm; the UbiSense Smart Space Platform,[222] which uses a mixture of TDOA and AOA with ultra-wideband (UWB) RF transmissions to obtain ∼15cm accuracies; and the Maxelbot Experimental Sonic Range Finder system,[227; 223] which uses RF to initiate with TDOA for three ultrasound transceivers on each unit, combining TDOA and AOA.

TDOA requires synchronisation between clocks, in order to give a basis for transmission or receiving times. Clock synchronisation is discussed in Section 7.1.6.

7.1.5.1 RF/Ultrasound mix

In the case where both RF and ultrasound are used it is possible to calculate a distance purely from the time difference of arrival from one transmitter to one receiver. As RF and ultrasound travel at different speeds they will cross the same distance in different times. If both are transmitted simultaneously at one point then by measuring the time difference of arrival between the two signals at another point will give the distance between them.

The most notable system to use this is the Cricket Location System[228] from MIT, which uses a network of fixed transmitters around an office environment with

mobile receivers. The receivers listen for a radio signal, and once received listen for an ultrasound signal. They have an accuracy of 1–3cm.

Other systems[229; 230] also use this method.

7.1.6 Time of Arrival (TOA)

For Time of Arrival (TOA) it is necessary to have a synchronised clock between the units — the first unit will transmit a pulse which the second will receive, along with the time at which the pulse was transmitted; the second can compare the time that the signal arrived against the time that it was transmitted to obtain a Time of Flight (TOF) and from there a distance.

Perfect clock synchronisation is in many situations impossible, although some systems do make use of this. GPS is the most notable example — by using expensive and highly accurate atomic clocks the satellites will all have perfectly synchronised clocks, and by comparing against more satellites than would otherwise be necessary it is possible for a GPS receiver to obtain an accurate time from the satellites without needing an accurate clock of its own. Other methods involve synchronising units using a remote clock — for example GPS (although this is not available in many situations), the Rugby signal (limited to the UK), or an NTP (Network Time Protocol) server — or by using an often less accurate synchronisation method based on passing times between nodes in a network such as Reference Broadcast Synchronization (RBS),[231; 220] or others.[232]

In the same manner as GPS, it is possible to have some units synchronised and others to obtain a time from them. Systems which use TOA in a similar way to GPS using a fixed local network are commonly referred to as Pseudolites.

One way in which the requirement of clock synchronisation can be avoided is to measure round trip time. The first unit will transmit a pulse, which the second will receive. The second will do likewise. By subtracting the time between receiving a pulse and sending a pulse at the second unit from the time between sending a pulse and receiving a pulse at the first the round trip time is calculated. This is discussed further in section 7.3.

Examples of this are the RF based Pinpoint system[233] (described in more

detail in section 7.2.3) and the audio based BeepBeep[234] (similarly described in section 7.2.2), and this method is also the basis for the TimeLoc[235] protocol used by Locata.[236]

Winkler et al.[237] describe a variation on this method which uses a number of fixed dumb receivers. One 'Master Base Station' transmits a signal, and on receiving this the mobile unit transmits another signal in reply. As the locations of the fixed receivers are known, the TOF of the first signal to them can be calculated. The transmission time for the mobile unit can be worked out as described above, and the TDOA between the two signals at the fixed units — along with the known TOF between them and the Master Base Station — gives the TOF between the fixed units and the mobile unit.

It should be noted that an instantaneous response from the second unit would remove the need for measuring a second pair of time stamps[238] — but as processing time is non-negligible this would lead to larger errors, and as each pair needs to communicate individually this scales badly ($o(n^2)$ instead of $o(n)$). Giustiniano and Mangold[239] describes such a system, which uses the time of the ACK response in the 802.11 wifi protocol to calculate distances (measuring the MAC idle time, which is proportional to TOF and ACK signal strength).

7.1.7 UAV localisation decision

For localisation between a swarm of UAVs the best solution would be to use radio based TOA. Radio will work over long distances, and is far less susceptible to multipath effects. It also does not require line of sight, as a system based (for example) on light would. The main problem with radio is that even small timing errors will lead to very large distance errors due to the speed of light. The next best option would be to use a hybrid RF/ultrasound TDOA method such as the Cricket system, although this would give a reduced range, particularly in noisy environments, and would be susceptible to multipath errors.

One source of error for time based systems is when units are moving, and this is investigated in more detail in section7.6. As development of a radio based TOA system with sufficient accuracy is non-trivial, it was decided to investigate audio

based TOA — and the effect of motion upon it in particular — as a stepping stone towards it. Audio, with its lower speed, can be measured comparatively easily but retains many similarities to a radio TOA system.

For the remainder of this thesis, unless otherwise specified, an echo based TOA system (such as PinPoint or BeepBeep) is assumed.

7.2 Literature review

A selection of the above mentioned works which are particularly relevant are described here in more detail.

7.2.1 Overviews of relative positioning

Castillo-Effen[193] gives an overview of various localisation methods, and describes an interesting means for categorising them using four attributes: Definition of location (whether quantitative, giving an exact X,Y position, or qualitative, giving an abstract area such as a room), Research communities that have an interest in the localization problem (navigation, robotics, wireless networking, or localisation theory), Processing and infrastructure required by the technological solutions (whether cooperative, centralised, or using an infrastructure), and Sensors and measurements (inertial, or using one of the methods as described in Section 7.1).

Hightower and Borriello[240] similarly creates a taxonomy for categorising localisation methods and uses it to compare a variety of different options, including many which are not included in this paper (such as vision or contact based systems). The categories selected are similar but not identical to those described above, and consist of Physical Position vs Symbolic Location (exact X,Y position vs abstract area), Absolute vs Relative (whether the positioning is tied to a global frame of reference or locally to other units), Localized Location Computation (whether the positioning is calculated by the unit or by a central server), Accuracy and Precision, Scale, Recognition (a method for globally identifying individual units), Cost, and Limitations.

Mautz[241] also describes some of the options for indoor localisation, including a graph comparing the range and accuracy of the various methods.

7.2.2 BeepBeep

The BeepBeep Acoustic Ranging System[234] is designed as a method for calculating distances between pieces of inexpensive COTS[3] equipment such as 'cell phones, PDAs, MP3 players, etc.', without modification, meaning that it must be performed in software. The requirements of the equipment are a microphone, a speaker, and some form of device to device communication. It is unusual in that it is designed to use audible pulses instead of ultrasound.

The aspect of the paper which is most important is its method of timing. It is extremely difficult to tie a particular sample from a microphone on a computer to an exact time. There are several layers between the hardware of the ADC and the software of the program, and in most cases programs do not have exclusive use of the processor and sound card. To record a sound it is necessary to make a request for a certain number of samples. These samples will be recorded as soon as the computer is able to, and the results placed into a buffer which is returned to the program as soon as possible. If a program takes a timestamp and then records, the timestamp will be marginally before the actual time of the start of the recording. The same is true in reverse when instructing a sound card to make a noise. The difference may not be sufficient to cause problems in most contexts (it will be sufficiently fast that to a human it will appear as instantaneous), but it is high enough, and inconsistent enough, to cause large distance errors.

The BeepBeep system bypasses this in two ways. Firstly, it does not time transmitting as being different from receiving; instead it listens for its own transmission (the sound coming out of the speaker and immediately into the microphone), and calculates the transmit time as it would a receive time. Secondly it solves the problem of timestamps when listening by recording for a long period — sufficient to include all pulses required — and then working out the TDOA of the transmitted pulse and the received pulse. As the time difference is all that is required (actual

[3]Commercial Off The Shelf

times are unnecessary — see Section7.3), the error is reduced to being based purely on the time taken to recognise a pulse rather than also on the time taken to record it.

The BeepBeep protocol consists of three stages. To start with one unit sends a message containing a schedule to say when the other units should transmit. To prevent overlaps caused by clock skews, a 1s buffer is placed between all transmissions. When all of the units have received this schedule they will wait until the time allotted to them before transmitting, meanwhile recording all that occurs. When the schedule is complete (all units have transmitted), they each process the recorded data to obtain a series of TDOA, and broadcast this to all other units.

The signal used is a 50ms 2–6kHz linear chirp, with a 5ms 2kHz sine wave played before it to warm up the speaker cone to reduce distortion.

The idea of post processing a series of samples to gain timestamps based on those samples is a clever one, however it does not entirely get around the problems associated with timing. The system is described as having 'no notion of local clock or timestamp at all', as samples are counted between received signals rather than times being taken for each one, but this ignores the fact that samples are a method of counting time in exactly the same way as seconds. If one unit has a sampling rate that is not exactly at the frequency expected or is not in sync with another then errors from these will accumulate in exactly the same way as with taking timestamps, just without the delays associated with recording.

The system is tested in a variety of situations, both indoor and outdoor, at different distances, and it reports a maximum average error outdoors of 3.8cm over an operational range of 12m, with a confidence of 98%. Over shorter distances indoors they managed a maximum average error of 1.4cm over 4.5m with 100% confidence.

7.2.3 PinPoint

PinPoint[233] is very similar to BeepBeep in a great many ways. The most notable difference is that it uses RF rather than audio for TOA. It uses custom hardware with a high speed FPGA, and — as with BeepBeep — calculates the distance be-

tween the units by recording the timestamps for sending pulses and receiving them from other units. Its measurements assume clock drift, and take it into account by causing all units to transmit two pulses instead of one — the difference between the two measurements gives the difference between the clock speeds. This process is described in more detail in Section7.4.2.

As the PinPoint system uses custom hardware, the time delay for transmitting and receiving data will be very small and relatively constant (unlike sending data to a sound card on a computer, where the delay can be extremely variable). PinPoint therefore assumes that it is constant, and measures it by transmitting a pulse and listening for it at the same unit (similarly to BeepBeep, but only doing it intermittently). This means that it can take the timestamp for transmission from the software, rather than listening for itself in the hardware.

The pulse used by this system is rather different to the chirp used by BeepBeep, or to the sine-based pulse described later in this chapter. It consists of 20, 128 microsecond baseband cycles, followed by one marker baseband cycle which is high for 64 microseconds (as before), but the half cycle where it would normally be low is filled by 18 quick pulses (each approx. 3.2 microseconds long). This is followed by another 20 128 microsecond baseband cycles. The reason for this is to make it more robust — even if some of the baseband cycles have been missed, it will still be able to calculate a precise receive time (presumably as long as the middle marker pulse is detected).

At the start of a ranging process a master node instructs all other nodes in turn to transmit. They do so, twice (to obtain clock drift data), and then each send a further signal containing the timing data obtained so that all units can calculate their offset from each other.

For a radio based system, the results obtained by PinPoint are very good. They sample at 300MHz, which gives them a resolution equating to almost exactly 1m, meaning a theoretical timing error equating to up to 2m (as transmission timings are taken from software). They achieve average errors of 1.3m, with maximum errors of 4.3m. It should be noted that the paper does not describe the sampling process — whether the readings at the 7–17 locations tested were single readings (meaning the

average and maximum errors were from up to 17 readings), or whether the readings were each themselves averages of the results obtained while in that location.

The system is also tested in a minimal moving situation, where the mobile unit is moved at a rate of 0.762 m/s along an approx. 50m track. The results for this are extremely good; this is liable to be due to the very high sampling rate.

The PinPoint method would seem a perfect system for UAVs. It is RF based, and has accuracies that are suitably small for all but close formation flying (the errors described would require a 10m+ exclusion zone between UAVs).

However, it transmits at a very high rate (see Section 7.1 for a brief description of the problems with this, further elaborated later in the chapter), and has not undergone experimentation at high speed.

7.3 Time of Arrival measurements

As described in section 7.1.6, one way in which the requirement of clock synchronisation can be avoided is to measure round trip time. This commonly relies on the fact that the distance between two units is constant, and so the length of time spent between pulses being sent and received is unimportant; the only distance inaccuracies are based on measurement errors. (This is only true if all units can be assumed stationary (this is a common assumption of most methods, and is true in most cases) — if the units are moving then there will be additional errors, which are covered in section 7.6.)

Take two units — A and B — which are stationary and a fixed distance d apart (see Figure 7.1). The two units can transmit and receive pulses, which travel at a speed V_S.

Unit A transmits a pulse at time t_1, which is received by Unit B at time t_2. After a delay, Unit B then transmits a pulse at time t_3, which is received by Unit A at time t_4. No assumption is made of clock synchronisation; therefore times recorded by Unit A cannot be directly compared to times recorded at Unit B.

As the exact point at which the timing is started at each unit is immaterial (everything can be easily returned to an arbitrary starting point), let $t_1 = 0$. There are two periods which are timed: $t_A = t_4 - t_1$ by Unit A, and $t_B = t_3 - t_2$ by Unit

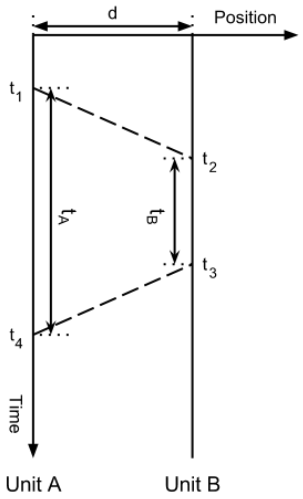


Figure 7.1: Two units, distance d apart, transmitting (at t_1 andt_3 respectively) and receiving (at t_2 and t_4 respectively)

B.

As the distance is constant, subtracting the time difference at Unit B from the time difference at Unit A gives a round trip time, and so a distance:

$$\begin{aligned} d &= \frac{(t_4 - t_1) - (t_3 - t_2)}{2} V_S \\ &= \frac{t_A - t_B}{2} V_S \end{aligned} \tag{7.1}$$

In order to measure the time between received pulses, we need to be able to detect the pulses. There are many ways in which a signal can be detected. These depend on several factors, such as the specification of the signal. Three of these which we will now cover are using a Fast Chirp Transform (FCT), using a Fourier transform, or using cross correlation, although others (such as the Fan Chirp Transform (FChT) are available.

To compare these a pulse signal was created which could be detected by any of the three methods, combining both a sine wave and a quadratic chirp[4].

[4]In this chapter, where a chirp is referred to as being 'quadratic' it is a chirp of the form $cos(Ax^2 + Bx + C)$. This is referred to in some places as a quadratic chirp due to the quadratic equation inside the cosine[242], and in some places as a linear chirp due to the linear change in

7.3.1 Pulse description

A pulse was selected which could be detected using a Fourier transform, cross correlation, or a Fast Chirp Transform:

1. 0.1 second 2kHz sine wave
2. 0.05 second linear chirp, from 2kHz to 4kHz
3. 0.1 second sine wave at a random frequency (specific to that pulse in the short term)

This can be seen in Figure 7.2. The 2kHz sine wave can be detected by a Fourier transform; the linear chirp can be detected by a Fast Chirp Transform or by cross correlation; and the closing frequency allows the different pulses to be easily differentiated, which means that time difference of arrival can be compared without a complex timing protocol.

7.3.2 Fast Chirp Transform

The Fast Chirp Transform (FCT)[242; 243] was developed by Jenet and Prince, initially to detect gravitational waves from black holes, but it can be used to detect chirps in a wide range of applications.

It works by taking a 2d fourier transform over a specially prepared sparsely populated array, which has the signal placed through it. The signal is run along a line where the X coordinate matches the position in the signal, and the Y coordinate follows the equation[5]:

$$X = S(t)\%N_X Y = N_Y \left(\frac{S(t)}{N_X}\right)^2 \tag{7.2}$$

where (X, Y) is the coordinates on the sparsely populated array, $S(t)$ is the signal at time t, and (N_X, N_Y) is the size of the array.

frequency over time

[5]The derivation for this can be found in [242], where $\phi = x^2$, and this is scaled to the size of the array.

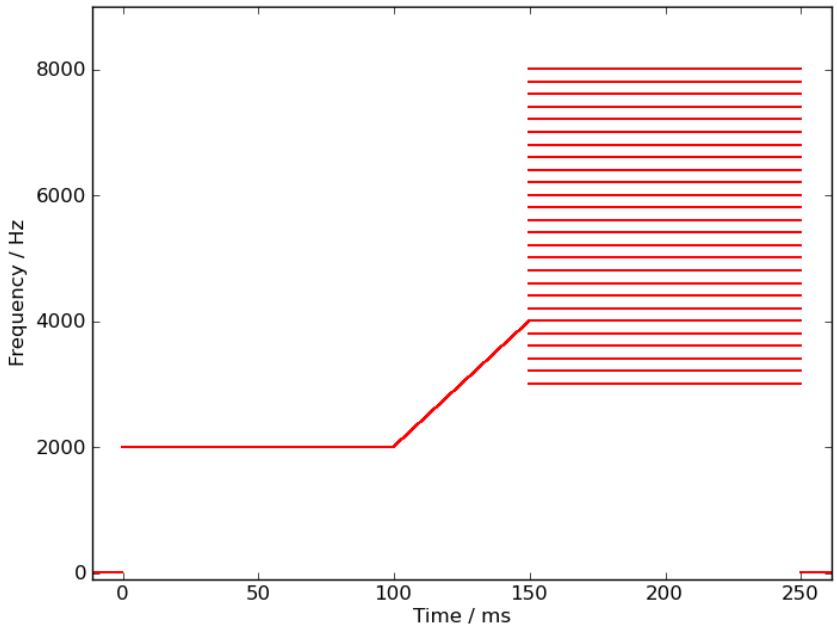


Figure 7.2: Frequency diagram of the pulse used in the experiment

This can be seen in Figure 7.3, which shows the sparsely populated array with the signal passing through it. The large blue sections of the image show areas which contain no data.

When a Fourier transform is taken of this array the resultant output will give a high point corresponding to a detected chirp. Figure 7.4 shows the Fourier transform of Figure 7.3, showing a large peak in the top left hand corner.

The position of this peak is dependent on the specification of the chirp. Given the equation of a chirp:

$$cos(Ax^2 + Bx + C) \tag{7.3}$$

(where x is the time from the start of the chirp). If this chirp is put through a FCT the output will show a peak at position (A, B), assuming that the chirp started at $t = 0$.

If the sample which is being analysed instead contains a later section of the same

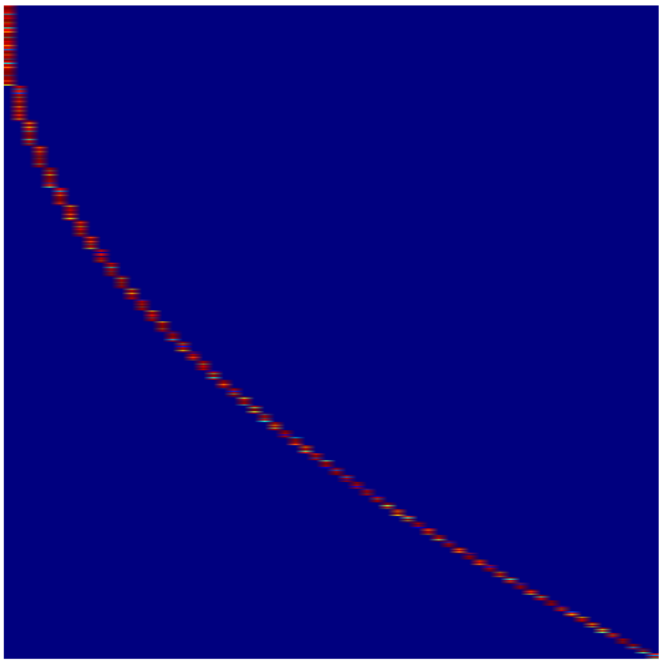


Figure 7.3: A sparsely populated array containing the signal, prepared for the FFT, as part of the FCT

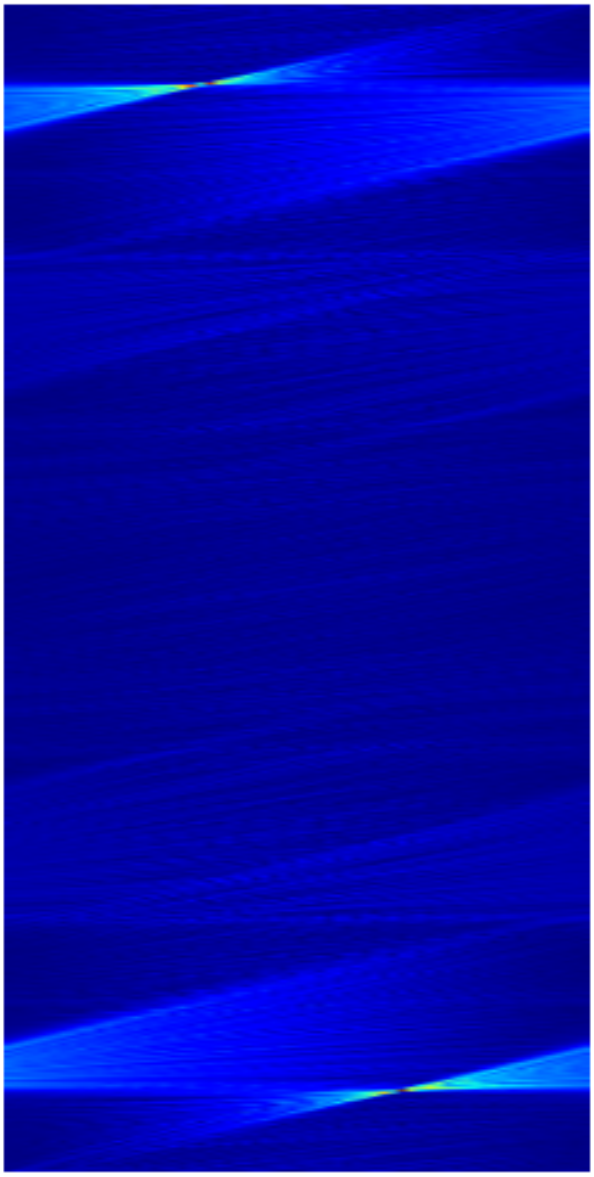


Figure 7.4: The output from the Fast Chirp Transform, showing a peak in the top left corner corresponding to a detected chirp. The second peak (bottom right corner) is a mirror image of the first, and is not a second chirp.

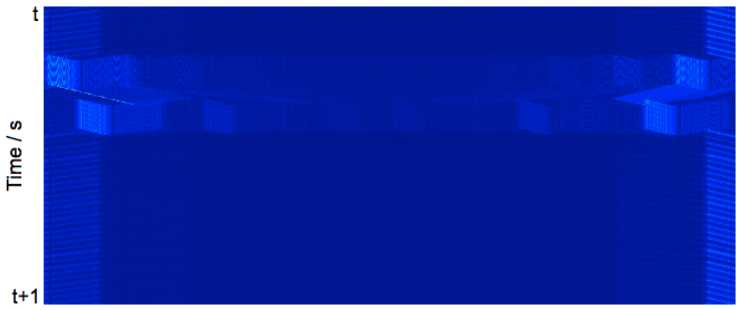


Figure 7.5: Layered strips from FCT outputs, showing values corresponding to all chirps with a set gradient of frequency change over a 1s period. For each row, the position of the peak will give the time of the start of the chirp. A strong chirp is visible as a diagonal line down the left side.

chirp, starting at at $t = n$ such that $t = x + n$, the chirp itself will be the same but the visible portion (ie. the portion which is passed to the FCT) will be effectively different:

$$cos(A(x+n)^2 + B(x+n) + C) \tag{7.4}$$

Splitting this out and recombining gives:

$$cos(Ax^2 + (2An + B)x + (An^2 + Bn + C)) \tag{7.5}$$

This would therefore give a peak at position $(A,\ 2An + B)$. This means that depending on the point in the sample which contains the chirp it will produce a peak in a different position, and these will follow a straight line based on that starting point. It is therefore possible to track the line backwards until it intercepts with B, giving the start time of the chirp with great accuracy regardless of the time at which the readings were taken.

As A will remain constant, a strip of values can be taken from each reading and layered together to show how $2An + B$ changes over time. Figure 7.5 shows a period of one second, with a line of a chirp visible along the left side.

The difficulty comes with a quiet signal in a noisy environment. The line formed by the chirp is still there, but are almost invisible to the eye (see Figure 7.6). In order to detect the line we must also detect many points which are not part of this

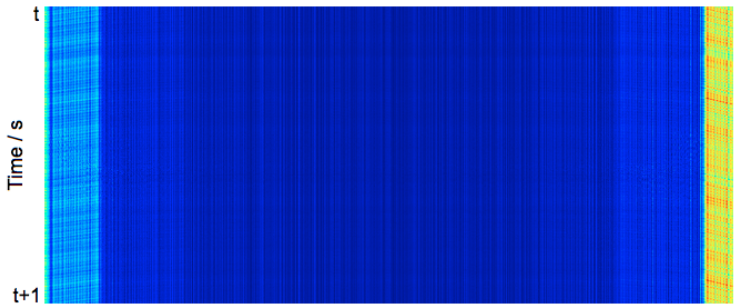


Figure 7.6: Layered strips from FCT outputs, showing values corresponding to all chirps with a set gradient of frequency change over a 1s period. For each row, the position of the peak will give the time of the start of the chirp. A weak chirp exists along the left side, but is very difficult to see by eye.

line, and so therefore need a means of detecting which points are relevant and which are not. We can do this using a Hough transform.

The Hough transform is used for detecting straight lines in images[6]. It works by taking a series of points which may be parts of lines, and for each plotting a sine wave which represents the possible lines it could intercept on an (r, θ) graph.

For a given infinite line (r, θ), which is perpendicular to a line at θ degrees and distance r from the origin, every point which is tested that would form part of it will produce sine waves on the (r, θ) graph which intersect at the same point. This means that if the specification for the line being searched for is known it is possible to quantify how likely it is that it exists.

In this case, the angle of the line is known but its intercept is not. A sufficiently large peak along the correct angle (θ) will both show that the line exists in this image and also give the point at which it intercepts (r). Figure 7.7 shows the Hough output from Figure 7.5.

In order to use the Hough transform, the input image must be binary — this allows for each point to either produce a sine wave in the output or to be ignored. This conversion to binary does, however, cause some problems. A threshold must be taken, but the position of this threshold may not be correct for all signals.

Figure 7.8 shows the strong signal from Figure 7.5 with both a high and a low

[6]It can also be used for detecting arbitrary shapes, but in its original form it is used for straight lines

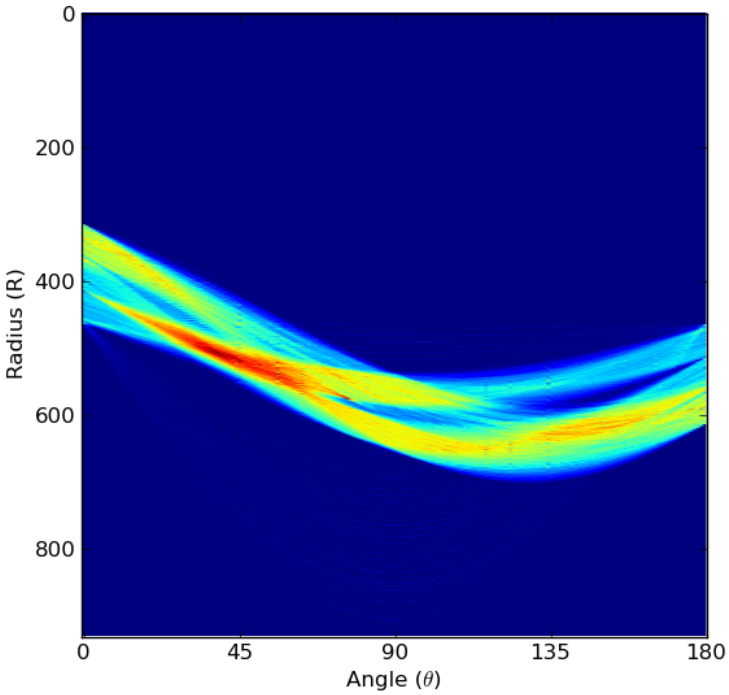


Figure 7.7: The output from the Hough transform, used to detect the line of the chirp. This is represented as a peak.

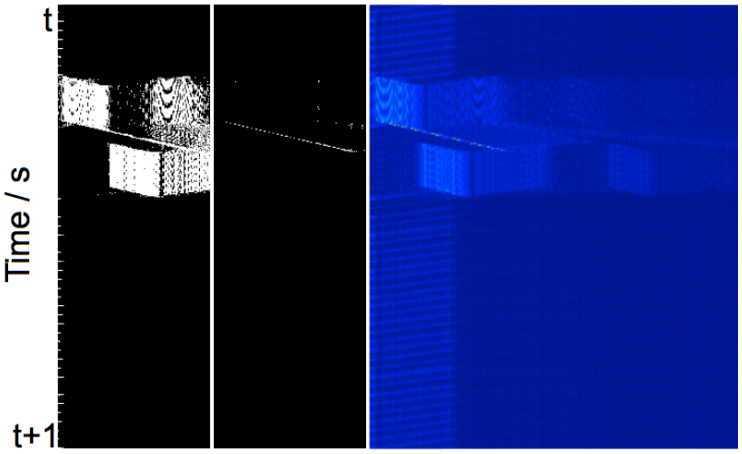


Figure 7.8: Binary images for a strong chirp signal, at two thresholds

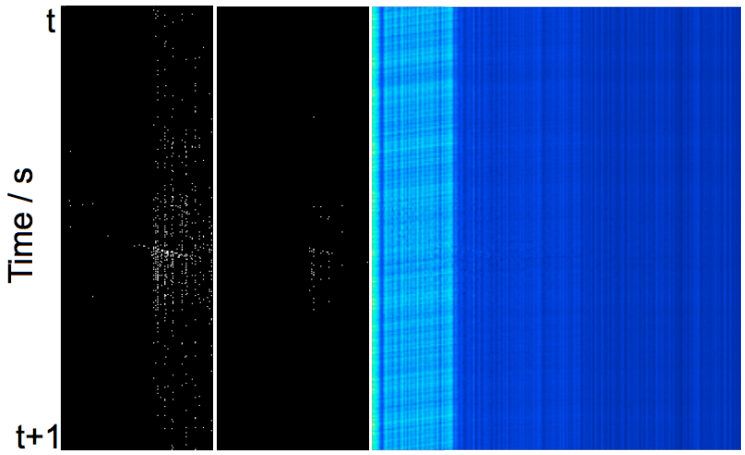


Figure 7.9: Binary images for a weak chirp signal, at two thresholds

threshold (1 and 5 standard deviations above the mean). Figure 7.9 shows the same thresholds for the weak signal from Figure 7.6.

In the case of the strong signal, the low threshold allows through too much — the Hough transform would produce many possible lines which are not actually as strong as the correct line. In the case of the weak signal, the high threshold does not allow through enough — the signal is so weak that the Hough transform would not have enough data points to produce any reasonable lines.

In order to fix this, it is possible to take several Hough transforms for each image, each using a different threshold for the binary conversion. These can then be summed so that the peak is visible for weak and strong signals.

7.3.3 Fourier Transform and Cross Correlation

7.3.3.1 Cross Correlation

It is possible to search for a chirp signal by cross correlating a known chirp against the signal[234]. A quadratic chirp can be easily detected through cross correlation, as no part of its signal is repeated. As a chirp can be easily converted into an impulse (and vice versa), they can provide an accurate timestamp while remaining sufficiently long that they can be detected over long distances.[244]

The chirp in the pulse is of a known length c. If we are continuously searching for an unknown but lengthy period of T samples for a signal, we can split the task

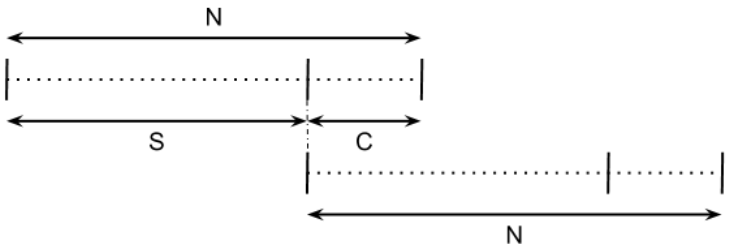


Figure 7.10: $N = S + C$, where N is the number of samples being tested for this iteration, C is the chirp length / overlap, and S is the number of samples searched per iteration / step length.

up into smaller sections of N samples each. (The reason for splitting the data up into sections is that the data will be arriving in real time for an unknown length of time; it is therefore necessary to process the data in chunks rather than waiting for it all to have arrived[7])

As the correlation requires a full overlap of all data (see Figure 7.10), $S = N - C$ samples will be completely searched per N samples tested, where $C = c - 1$ (this is to prevent the same sample being tested twice — once in this set of N samples, once in the next set).

The program will therefore analyse a length of N samples (comparing against S of them) then move forward S samples, meaning that to completely search T samples will take $\frac{T-C}{N-C}$ iterations.

Logically $C < N \leq T$, and for the following some values can be assumed. The chirp length in samples will be the multiple of the sample rate and the required chirp length in seconds. Assuming a 0.05s chirp and a sample rate of 44100, $C = 2205$. Similarly, assuming a 10s sampling period, $T = 441000$. The speed at which each iteration can be tested will vary depending on N.

7.3.3.2 Fourier Transform

A Fourier transform can be used to detect for pulses which are made up from sine waves of known frequency. The program can pass through the data a certain num-

[7]This is different from the situation described by Peng et al.[234], where the entire period is received at once.

ber of samples at a time and perform a fast Fourier transform to search for a set frequency. This could give a number of potential improvements — it could increase the speed at which the analysis could be performed, and it could allow for an easier system to differentiate between pulses without requiring a complicated timing protocol (as in Peng et al.) by using different frequencies[8].

For ease of comparison we will use the same variable names where possible as above. Let N be the number of samples used in the Fourier transform, and S be the step length (that is, the number of samples further on that the next set of samples to be tested are taken; see Figure 7.10).

It is important that $S < N$, and it should be noted that as $S \to N$ there is an increased risk of a signal being missed because it occurs at the boundary between tests. The greater the overlap (C) the slower the detection, but the less the chance of missing a signal (at least when C is small).

This method will analyse N samples then move forward S samples, meaning that to completely search T samples will take $\frac{T-(N-S)}{S}$ iterations.

N needs to be reasonably high so that the resolution in the frequency domain is sufficient to detect a set frequency. Let $N = 2^{13}$. As above, $T = 441000$. The speed of iterations will be constant for a constant value of N.

It should be noted that once the FFT method as described above has discovered a signal it only knows the position to a resolution of S samples. This can be solved by going through the section again at a higher resolution. This will increase the number of iterations to $\frac{T-(N-S)}{S} + \frac{N}{S_2}$, where S_2 is the new step length.

In theory, this would give a resolution of 22.7 microseconds (1 sample, for $S_2 = 1$ and a sampling rate of 44100), or sub cm when used for calculating distances as described in Section 7.3. In practice, when testing this experimentally, this gave errors in excess of 8m. The reason behind this may be that depending on background noise it required varying amounts of the sine wave to be included in the data before being detected by the Fourier transform.

[8]It is possible to use varying chirp specifications, but with the limitations of frequency range and a minimum length of pulse, the set of possible quadratic chirps is also limited

7.3.3.3 Combination

In order to gain both the speed of the FFT and the accuracy of the cross correlation, it is possible to combine the two. As the pulse consists of both a sine wave and a chirp, detection can be obtained by performing a quick pass at a low resolution (high S) with a Fourier transform, followed by a second pass over any sections which tested positive looking for a chirp. This would give — for a period T containing a single pulse — $\frac{T-(N-S)}{S}$ iterations for the FFT, and one iteration of length N for the cross correlation, where N in this case is fixed at 2^{13}.

From experimentation, using a Fourier transform gave a worse accuracy than cross correlation, but cross correlation had a longer execution time than the Fourier transform (as S tends to N). Using a combination of the two, as described above, gives the same accuracy as cross correlation but the time compares very favourably against a pure cross correlation time, being 3-4 times faster. For this reason, this combination is used for comparison against the Fast Chirp Transform.

7.4 Sources of error

There are several sources of error in time of arrival measurements. The main one is from measurement errors — a small error in the measured time of a pulse will translate to a long distance error, and as each value is used for two distance measurements (comparing against the pulse before and the pulse after) any incorrect value will skew two results.

The methods described above assume stationary units. If the objects are moving, this will cause additional errors — these are described more in Section 7.6.

Although any real life situation would require positioning in 2D or 3D, for this section we will assume measurements between two points.

7.4.1 Clock resolution

One of the most obvious sources of error is clock resolution — that is, the sampling rate, or time between samples.

For each of the four measurements required to find a distance (t_1, t_2, t_3 and t_4)

there will be an associated error from the clock resolution. Let t_{xM} be the measured time for the signal at actual time t_x:

$$t_x = t_{xM} + E_x \tag{7.6}$$

where E_x is the timing error, R is the clock resolution, and $E_x < R$. Assuming a stationary unit (as described in section 7.3):

$$\begin{aligned} t_A &= t_4 - t_1 \\ &= t_{4M} + E_4 - t_{1M} - E_1 && (7.7) \\ t_{AM} &= t_{4M} - t_{1M} && (7.8) \\ t_A &= t_{AM} + E_A && (7.9) \\ E_A &= E_4 - E_1 && (7.10) \end{aligned}$$

where E_A is the timing error, and $-R < E_A < R$. (Similarly, E_B is the timing error on t_B, and $-R < E_B < R$).

Let t_{RT} be the round trip time, such that $t_{RT} = t_A - t_B$:

$$t_{RT} = t_{RTM} + E_{RT} \tag{7.11}$$

where E_{RT} is the round trip timing error, and $-2R < E_{RT} < 2R$. Let d be the distance between units:

$$\begin{aligned} d &= V_S \frac{t_{RTM} + E_{RT}}{2} \\ &= d_M + E_d && (7.12) \end{aligned}$$

where E_d is the distance error, and $-V_S R < E_d < V_S R$.

Putting in some reasonable numbers, if the sample rate $\frac{1}{R}$ is 44100 (a standard rate for sound cards) and V_S is the speed of sound, $-7.7mm < E_d < 7.7mm$.

Hence — at that clock resolution — the measurement accuracy is around 1.5cm

using this method. Adding in additional pulses (for example when calculating clock drift, as described in section 7.4.2) will increase this error.

7.4.2 Clock Drift

The equations described above assume that the clocks are not synchronised. They also, however, assume that the clocks are in time with each other — that is, that one second measured by one will be the same length as one second as measured by the other.

If the two clocks are running at different frequencies (that is, one is running fast or slow with respect to the other) then the two times t_A and t_B will not be referring to the same length of seconds, and so the round trip time will be inaccurate.

The BeepBeep[234] paper attempts to solve this by saying that all units will be in phase because they are using the sampling rate of the sound cards, and these will be of a consistent length.[9] However, sound card sampling rates are only as accurate as their internal clocks, and as such they are liable to drift over time.[245; 246]

PinPoint[233] describes a method for allowing for clocks running at different frequencies. Instead of each unit transmitting a single pulse, the PinPoint ranging algorithm requires each to transmit twice. This means that all pairs of units have times for two signals travelling in the same direction. If both clocks were perfectly in time, and the distance between them was constant, then the time between the transmission of those two signals and the time between their being received would be the same. If one clock runs more slowly than the other then there will be a variation, proportional to the difference in times. PinPoint's method for taking this into account simply involves assuming that all clocks are liable to be out of time as well as out of sync, and to include variables for clock speed and offset in all calculations. The times for two pulses from each unit in a pair gives enough equations to solve for the unknowns.

[9]The paper does mention frequency drift, but describes it as being negligible if the sampling period is low; this does not solve the problem of frequencies being out of sync, however.

7.5 Stationary Experiment

To investigate relative positioning a test platform was created. This was based on the system described by Peng et al., with some modifications. It was written to run on a series of low cost and low specification laptops (primarily Asus EeePCs, but also others), all running Linux.[10] The problem was split into three tasks, which were each treated separately. These three tasks are Transmitting, Recording, and Processing. As with Peng et al.,[234] transmission times are recorded along with received times. For the experiment, processing was farmed out to a remote server with all data from the sound card being uploaded for processing in near real time. This could easily have been performed on the units if they had sufficient power, but outsourcing allowed for greater flexibility in development. In a final incarnation all of the processing could take place on the units themselves. However, allowing this as an option does open up several interesting possibilities, including allowing for units of varying processing power (the minimum requirement would be a microphone, speaker, and a means for streaming the sound), or for distributed processing (where data is farmed out across a swarm of UAVs).

The method described by PinPoint for mitigating clock drift was used, as the sound cards were found to have frequencies sufficiently different to those specified to cause errors in time difference comparison.

Two units were set to transmit pulses at a range of intervals. Both were also listening for pulses transmitted from both themselves and from the other unit. The first unit was placed in a fixed position with the other placed at varying distances from this and allowed to remain stationary until more than 50 distance measurements had been taken.

Each unit comprised of an EeePC, with external microphones. The base unit used an external, powered speaker; the moving unit used the EeePC's own internal speakers.

[10] The system as designed is sufficiently customisable to run on a much greater variety of hardware platforms, such as smartphones, and on a greater variety of operating systems, such as Windows or OS-X, with minimal software changes.

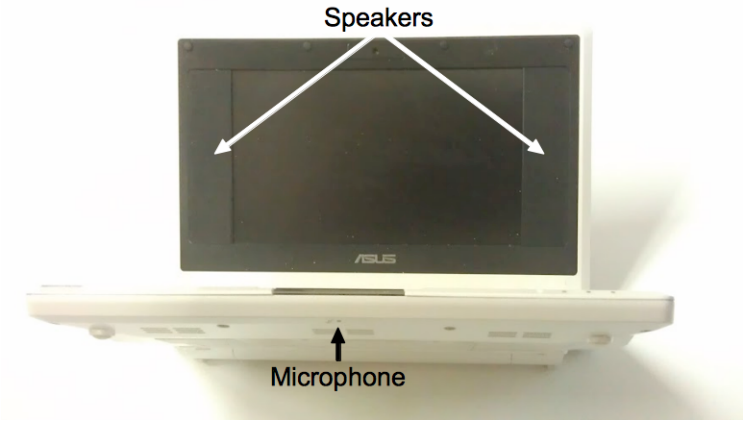


Figure 7.11: Asus EeePC, showing microphone and speakers, photographed from below to illustrate the position of the microphone

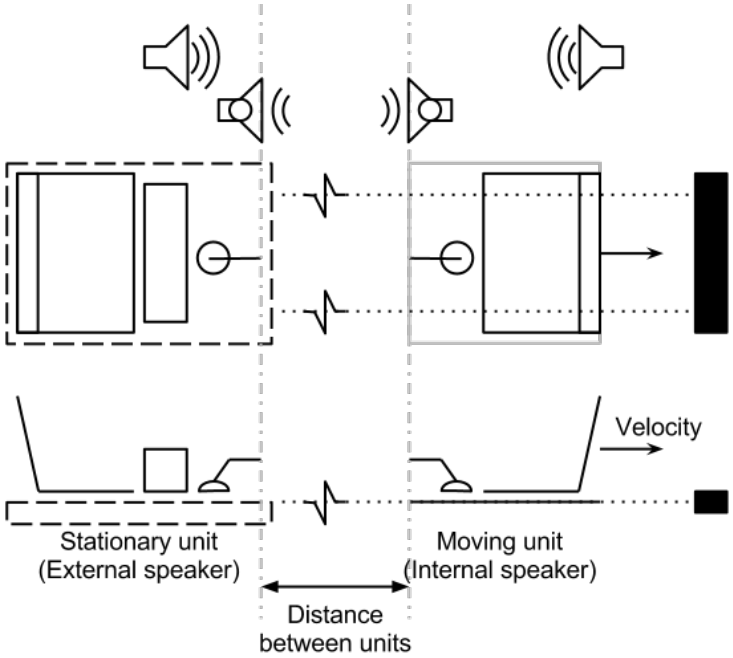


Figure 7.12: Experimental setup. Note that as the microphones and speakers are all in a direct line, the distance between the units as measured will be the distance between the microphones.

Pulse interval
1s
1.5s
2s
5s

Table 7.1: Pulse transmission intervals tested

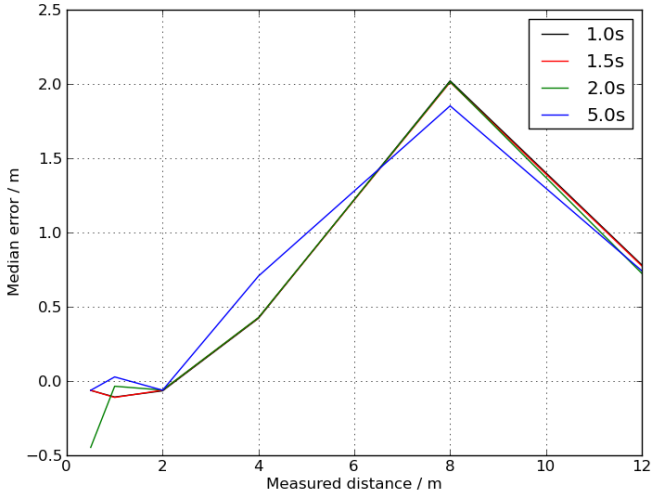


Figure 7.13: Median errors when stationary, at a range of distances, as measured using a Fourier transform and cross correlation. The error is the difference between distance as measured by the unit and the distance as measured by a tape measure. The four lines represent different pulse intervals; the number of seconds between one pulse and the next.

7.5.1 Stationary Results

The pulses were transmitted at a range of pulse intervals, as outlined in Table 7.1.

Figure 7.13 shows the distance calculated by the units using a Fourier transform and cross correlation, and Figure 7.14 shows the distance calculated using a Fast Chirp Transform, as compared to the distance measured with a tape measure.

Comparing the two against each other, the results from the FCT are notably worse than those obtained using cross correlation. The highest error in the first graph is 2m, at 8m; the errors in the second graph are over 2m for any distance over a metre, and appear to follow an almost linear correlation between error and

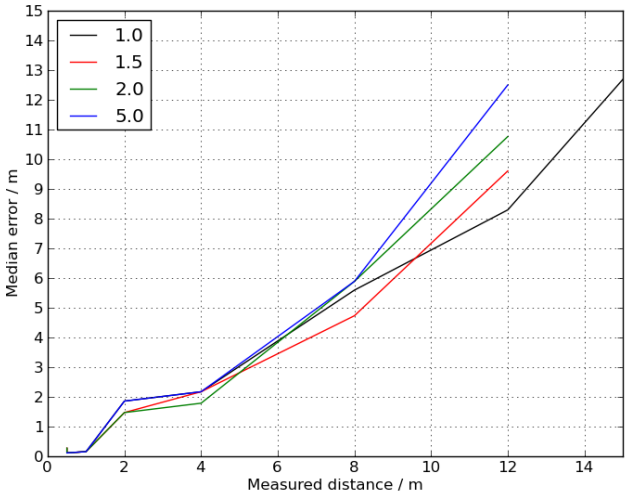


Figure 7.14: Median errors when stationary, at a range of distances, as measured using the Fast Chirp Transform. The error is the difference between distance as measured by the unit and the distance as measured by a tape measure. The four lines represent different pulse intervals; the number of seconds between one pulse and the next.

distance. The first (Fourier / cross correlation) graph does not appear to show much correlation against the measured distance, and is only able to detect up to distances of 12m, at which point only a small number of readings were available. For low distances the errors are by comparison very low, around 5–10cm. Neither graph compares well against those described by Peng et al., where the median values were within 1cm indoors (up to 4m) and 2cm outdoors (up to 15m).[234] However, this may in part be due to the location used (in our case, a corridor, which would have more opportunity for multipath errors).

It is notable that at low distances the median error for all intervals is approximately constant — that is, although the measurements are wrong they are consistently so. Figures 7.15 and 7.16 show the interquartile range (IQR), 90% range, and maximum/minimum difference (MM) for the Fourier/cross correlation and FCT methods respectively, to show the spread of the data. It should be noted that these are logarithmic graphs.

Again, the Fourier / cross correlation method appears to be working considerably

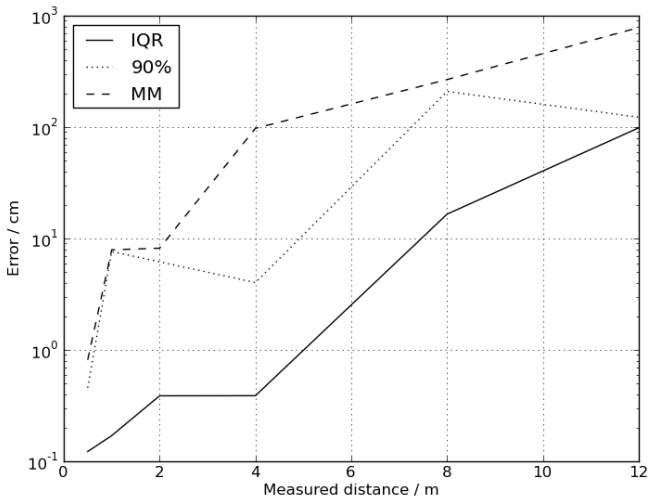


Figure 7.15: IQR, 90% and Max-Min (MM) ranges of errors when stationary, at a range of distances. Note that this graph is logarithmic, and (unlike others in this section) measured in cm.

better than the Fast Chirp Transform. In the first, the calculated distances are all within a small margin of error from the median — the IQR is always under a metre, and are well under 1cm for the first four metres. In the second, the range of readings is much wider — the IQR is around 5–15m for all distances from 2m and over. These error ranges are notably wider than the median errors.

The stationary errors will be to a small extent due to experimental error (i.e. the distance between the two devices as measured by a tape measure being incorrectly measured by a small amount, in the region of a couple of cm), but it is likely that the majority of the discrepancies — at least with the Fourier / cross correlation measurement — will be from multipath errors where the measurement is taken from a signal that has bounced off walls and ceiling instead of travelling directly. This is evidenced both by the consistency of the errors between measurements and by the relatively small IQR. In the case of the Fast Chirp Transform, however, the errors will be mainly timing errors from chirp detection — they cannot be based on the experiment, as the other method gives a better result for the same pulse data.

The units were able to measure distances up to a range of 12m using both methods. Readings at 15m were only taken with a 1s pulse interval, as they were

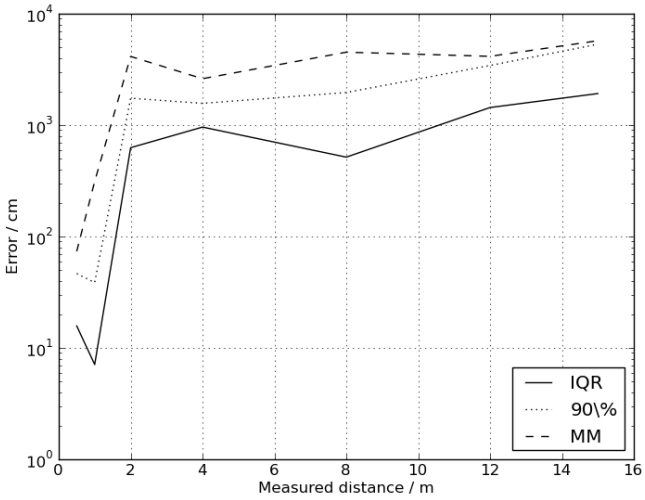


Figure 7.16: IQR, 90% and Max-Min (MM) ranges of errors when stationary, at a range of distances. Note that this graph is logarithmic, and (unlike others in this section) measured in cm.

not detectable using the Fourier transform / cross correlation method.

7.5.2 Conclusion

Of the two methods, the Fast Chirp Transform appears to be considerably the worse for accurately detecting chirp start times. This is surprising, as it is able to detect more pulses (particularly at further distances), and it should be able to calculate chirp start times with a high degree of accuracy by taking the line traced by the output peaks and mathematically calculating the intercept from the chirp equation (see Section 7.3.2).

Neither method is able to perform as well as systems described in the literature; this may in part be due to problems with the methods used for chirp detection, but a large factor will be from the scenario in which it is being used, which is prone to multipath errors from the walls.

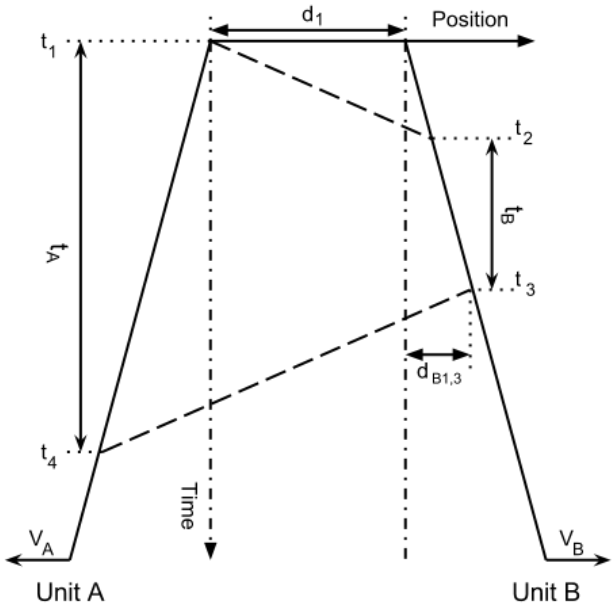


Figure 7.17: Two units, initially distance d_1 apart, moving at speeds V_A and V_B, transmitting (at t_1 and t_3 respectively) and receiving (at t_2 and t_4 respectively)

7.6 Movement errors

Take two units — A and B — travelling directly apart from each other at maximum velocities V_A and V_B respectively[11] (see Figure 7.17). At time t_n, they are distance d_n apart. The two units can transmit and receive pulses, which travel at a speed V_S.

As the exact point at which the timing is started at each unit is immaterial (everything can be easily returned to an arbitrary starting point), let $t_1 = 0$. There are two periods which are timed: $t_A = t_4 - t_1$ by Unit A, and $t_B = t_3 - t_2$ by Unit B.

Times t_2 and t_4, the times at which the pulse from one unit reaches the other unit, can be calculated from the transmission time, the distance travelled by the transmitting unit from the initial distance, the distance travelled by the receiving unit from the initial distance, and the speed of the pulse:

[11] Constant velocity is not guaranteed — the two units could be moving but stationary relative to one another, or moving towards one another, but the maximum velocities are known.

$$\begin{aligned} t_2 &= t_1 + \frac{d_1 + t_2 V_B}{V_S} \\ &= \frac{d_1}{V_S - V_B} \end{aligned} \tag{7.13}$$

$$\begin{aligned} t_4 &= t_3 + \frac{d_1 + t_3 V_B + t_4 V_A}{V_S} \\ &= \frac{d_1 + t_3(V_B + V_S)}{V_S - V_A} \end{aligned} \tag{7.14}$$

If the units were stationary — as described in Section 7.3 — it would be possible to find the distance between them from Equation 7.1. If this equation is used when the units are moving, however, it will create errors. Let d_M be the distance as measured by Equation 7.1.

Rearranging to include Equations 7.13 and 7.14, and noting that $t_A = t_4$:

$$d_M = \frac{t_4 - t_B}{2} V_S \tag{7.15}$$

$$\frac{2}{V_S} d_M = \frac{d_1 + t_3(V_B + V_S)}{V_S - V_A} - t_B \tag{7.16}$$

$$t_B = t_3 - t_2 \tag{7.17}$$

$$\begin{aligned} \frac{2}{V_S} d_M &= \frac{d_1 + t_B(V_B + V_S) + \frac{d_1}{V_S - V_B}(V_B + V_S)}{V_S - V_A} - t_B \\ &= \frac{d_1 + t_B(V_B + V_S) + \frac{d_1}{V_S - V_B}(V_B + V_S) - t_B(V_S - V_A)}{V_S - V_A} \\ &= \frac{2 d_1 V_S - t_B V_B^2 + t_B V_B V_S + t_B V_A V_S - t_B V_A V_B}{(V_S - V_A)(V_S - V_B)} \\ d_M &= \frac{2 d_1 V_S^2 + t_B V_S(-V_B^2 + V_B V_S + V_A V_S - V_A V_B)}{2(V_S - V_A)(V_S - V_B)} \\ &= d_1 \frac{V_S^2}{(V_S - V_A)(V_S - V_B)} + t_B \frac{V_S(V_A + V_B)}{2(V_S - V_A)} \end{aligned} \tag{7.18}$$

7.6.1 Negligible velocities

V_A and V_B can be considered negligible compared to V_S if they are sufficiently low, and t_B is sufficiently short. $V_S = 340.29 m/s$ if sound is used for the pulses, and 3×10^8 if radio is used; in at least the case of radio both V_A and V_B will almost

certainly be negligible in comparison. In this case $V_S - V_A$ and $V_S - V_B$ can both be approximated to V_S:

$$d_M \approx d_1 + \frac{V_A + V_B}{2} t_B \tag{7.19}$$

That is, the distance as measured will be the initial distance between the two units plus the average distance travelled by the units (average velocity times time) during the time spent at Unit B (assuming constant velocities).

$t_A - t_B$ is negligible in comparison to the velocities V_A and V_B. The distance calculated is therefore still accurate, but is out of date — by $\frac{t_B}{2}$ $\left(\approx \frac{t_A}{2}\right)$. This means that when the distances are used to trilaterate between units the resultant positions will be incorrect, as (best case, where velocities are constant) they will be all from different timestamps. Worst case, where velocities are changing, the distances measured may bear little relationship to the distance between the units at any point in time.

To take some example velocities, a walking human will move at 1.5m/s. A UAV helicopter might be moving at a top speed of 20m/s. A reasonable value for t_B might anything under 10s, depending on context.[12] Assume constant velocities.

If these moving units are assumed stationary during the ranging process, the errors for $t_B = 10$s for a walking human will be 15m, and for a UAV 200m. Those errors are before taking into account the fact that multiple units will need to exchange pulses before trilateration is possible, and in this time the units will be continuing to move.

7.6.2 Non-negligible velocities

If V_A and V_B are not considered negligible compared to V_S, we are left with Equation 7.18:

$$d_M = d_1 \frac{V_S^2}{(V_S - V_A)(V_S - V_B)} + t_B \frac{V_S(V_A + V_B)}{2(V_S - V_A)}$$

Using the example velocities above of 1.5m/s for a walking human and 20m/s for

[12]The Active Badge system[199]has a 15s transmit interval, to increase battery life and reduce the likelihood of collision

a UAV helicopter, with V_S being the speed of sound, comparing the equation above and Equation 7.19 (where V_A and V_B are considered negligible compared to V_S), we get a difference of $8.9 \times 10^{-3} d_1 + 6.6 \times 10^{-3} t_B$ for the human, and $0.129 d_1 + 1.25 t_B$ for the UAV. If $t_B = 10$s, we would get a 1m difference between the two equations for a value of d_1=105.20m for the human, and a difference of 12.5m for the UAV even if $d_1 = 0$.

It is therefore sensible to use this full equation in cases where not doing so would lead to unacceptable errors.

7.6.3 Distances

The above equations are calculated based on maximum velocities, as if the two units were travelling apart at a constant full speed. If they are instead calculated based on distances it can be seen that:

$$t_2 = t_1 + \frac{d_1 + d_{B1,2}}{V_S} \tag{7.20}$$

$$t_4 = t_3 + \frac{d_1 + d_{B1,3} + d_{A1,4}}{V_S} \tag{7.21}$$

$$\begin{aligned} d_M &= \frac{t_4 - t_B}{2} V_S \\ &= d_1 + \frac{d_{B1,2} + d_{B1,3} + d_{A1,4}}{2} \\ &= d_1 + \frac{d_{B2,3} + d_{A1,4}}{2} + d_{B1,2} \end{aligned} \tag{7.22}$$

where $d_{Na,b}$ is the distance travelled by unit N between times t_a and t_b, and $-V_N(t_b - t_a) \leq d_{Na,b} \leq V_N(t_b - t_a)$. This is the same as equation 7.19, with the additional term $d_{B1,2}$.

$d_{B1,2}$ cannot be measured directly, as it requires knowledge of time t_1 at Unit B, which is not known. It could, however, be considered negligible if V_B is sufficiently small and the threshold for error is sufficiently high. $d_{B1,2}$ is the distance travelled by Unit B between a pulse being transmitted from Unit A and being received by Unit B. If the pulse travels at a speed of V_S and Unit B has a maximum speed of V_B:

$$t_2 = \frac{d_1 + d_{B1,2}}{V_S} \tag{7.23}$$

$$d_{B1,2} \leq V_B t_2 \tag{7.24}$$

$$d_{B1,2} V_S \leq V_B d_1 + V_B d_{B1,2} \tag{7.25}$$

$$d_{B1,2} \leq d_1 \frac{V_B}{V_S - V_B} \tag{7.26}$$

Using the above example velocities of a walking human at 1.5m/s and a UAV at 20m/s, and V_S of the speed of sound, we get maximum values for $d_{B1,2}$ of $4.4 \times 10^{-3} d_1$ and $62.4 \times 10^{-3} d_1$ respectively. For an additional error of 1m, this would equate to values for d_1 of 226m and 16m respectively, neither of which are particularly far (depending on context).

7.6.4 Inertial

One potential method for reducing errors from motion would be to use sensor fusion between TOA and an inertial system, to give positions (albeit ones highly susceptible to drift over time) between timestamps by taking movement into account.

A detailed description of this is outside the scope of this paper.

As described in Section 7.6, if the two units are moving then they will not be able to measure an accurate distance, if t_B is sufficiently high.

Most methods assume stationary units, as for most purposes moving units can be assumed stationary if they transmit pulses at a high enough speed.

There are, however, several reasons why you would want to transmit less often. Transmitting a radio or audio pulse will use some power, so for systems where power consumption is very important it would be useful to transmit infrequently.[199] It also reduces congestion; if several units are all wanting to transmit within a short period then there will be a higher chance of overlap and the time between pulses must by necessity decrease. The spectrum might also be being used by other systems. A final reason why it is useful for units to be able to accept infrequent pulses is if they are working in a noisy environment. If only a fraction of the transmitted pulses arrive at their destination then the time between successful pulses will be high.

For these reasons it was decided to run an experiment to see what effect a range of values of t_B had on the accuracy of the relative positioning system at a range of speeds, comparing these experimental results to those obtained using Equation7.19.

As in the previous experiment, two units were set to transmit pulses at a range of intervals. Both were also listening for pulses transmitted from both themselves and from the other unit. One unit was placed in a fixed position, while the other was moved along a 19m long route travelling away from the fixed unit at a constant speed (see Figure 7.12).

Each unit comprised of an EeePC, with external microphones. The stationary unit used an external, powered speaker; the moving unit used the EeePC's own internal speakers.

Preliminary tests were run with other configurations — most notably using the EeePCs with their internal speakers and microphones only — and the results were similar to those described below, with the exception that using internal microphones and speakers varied the range. The range slightly reduced (particularly when using Fourier or Cross Correlation) for moving units and increased slightly for stationary ones. This was probably due to the microphones on EeePCs (see Figure 7.11) being positioned on the base of the computers, right next to any noise from motion, but nonetheless their being more sensitive than the external microphones being used.

To remove zeroing errors, the moving unit was allowed to sit for 10s at its initial position before moving.

7.6.5 Moving Results

7.6.5.1 Range

The moving unit travelled at a range of speeds, as outlined in Table 7.2, and at a range of pulse intervals, as outlined in Table 7.1.

Stationary distances were measured up to 12m (15m in the case of 1.0s pulse interval). Distance measurements when moving, as measured using the Fast Chirp Transform, went out of range at between 10 and 15m (see Figures 7.20 and 7.21). When using a Fourier transform and cross correlation, however, the unit went out of range in general far sooner than the stationary tests — in some cases at only a

Speed name	Speed (m/s)
Stationary	0 m/s
Very slow	0.17 m/s
Slow	0.25 m/s
Medium	0.33 m/s
Fast	0.48 m/s

Table 7.2: Unit speeds tested

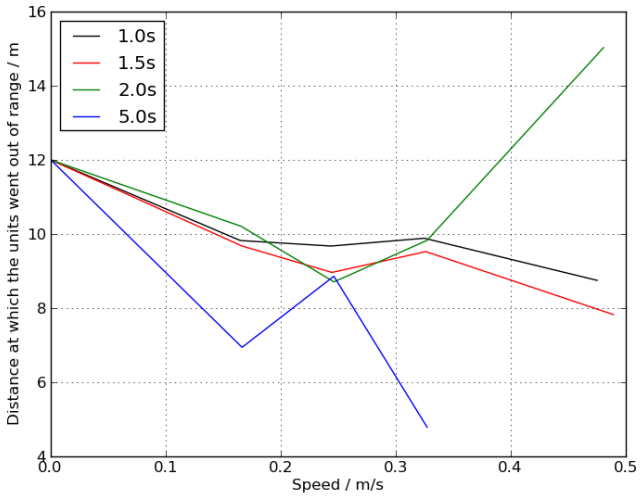


Figure 7.18: Average distance at which the moving unit went out of range, as measured using the Fourier transform and cross correlation at a range of speeds. The four lines represent four different pulse intervals. Stationary results were obtained by placing the units at a range of distances.

few metres (see Figures 7.18 and 7.19).

Although there is some variation as to when it went out of range between the speeds and the intervals, some of this will be due to the unit moving out of range between one pulse and the next (as the out of range distance can only be measured to the last received valid pulse) — if the unit's last transmission is five seconds before going out of range then it will disappear far sooner than if it was one second before going out of range.

Out of range was defined as being the first point at which the next five distance measurements all had an error of over 5m. In many cases, the unit which ceased to be able to hear first was the stationary unit, possibly because it has a considerably

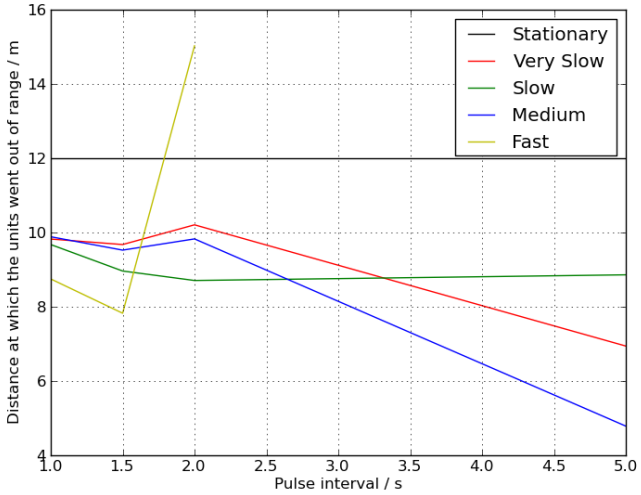


Figure 7.19: Average distance at which the moving unit went out of range, as measured using the Fourier transform and cross correlation at a range of pulse intervals. The five lines represent five different speeds. Stationary results were obtained by placing the units at a range of distances.

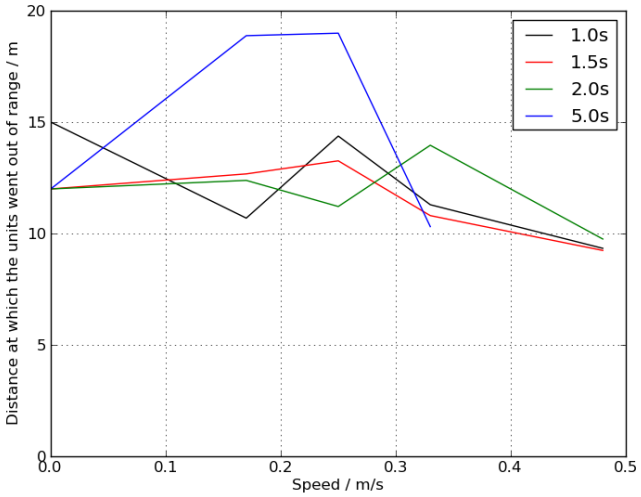


Figure 7.20: Average distance at which the moving unit went out of range, as measured using the Fast Chirp Transform at a range of speeds. The four lines represent four different pulse intervals. Stationary results were obtained by placing the units at a range of distances.

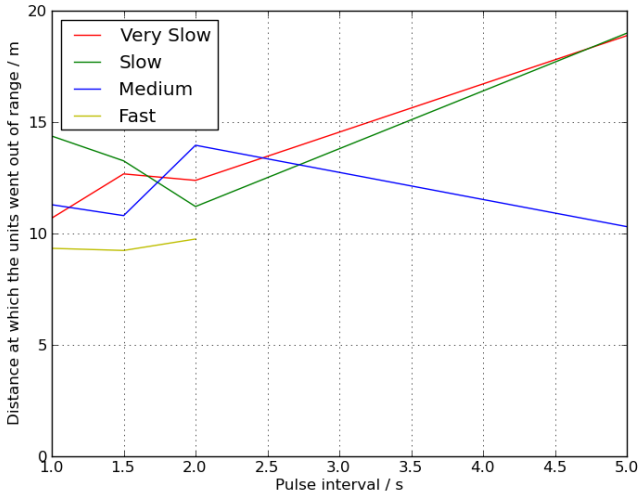


Figure 7.21: Average distance at which the moving unit went out of range, as measured using the Fast Chirp Transform at a range of pulse intervals. The five lines represent five different speeds. Stationary results were obtained by placing the units at a range of distances.

larger speaker than the mobile one.

Figures 7.22 and 7.23 show the average number of distance measurements made before going out of range for the Fourier transform and cross correlation, and Figures 7.24 and 7.25 show the same for the Fast Chirp Transform. As is to be expected, at larger intervals and at higher speeds the number of readings decreases.

The same audio recordings were analysed by both methods, so any variation in number of pulses measured is directly correlated to the method used. In general, the Fast Chirp Transfer detects around twice the number of pulses as Fourier/cross correlation, and goes out of range at about the same distance regardless of speed or pulse interval, with a couple of experiments not going out of range for the full length of the run.

From this, it would seem that the Fast Chirp Transform is a much better method to use in noisy environments as it is able to detect a considerably larger number of pulses and is able to detect them over a far larger distance.

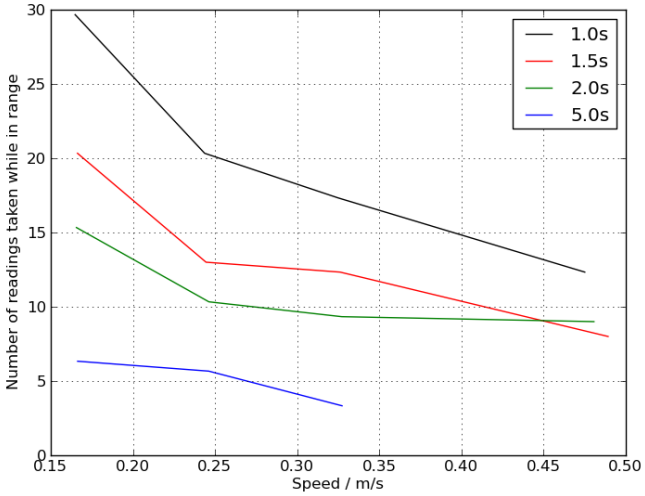


Figure 7.22: Average number of readings available (before going out of range), as measured using the Fourier transform and cross correlation at a range of speeds. The four lines represent four different pulse intervals. Stationary results are not included.

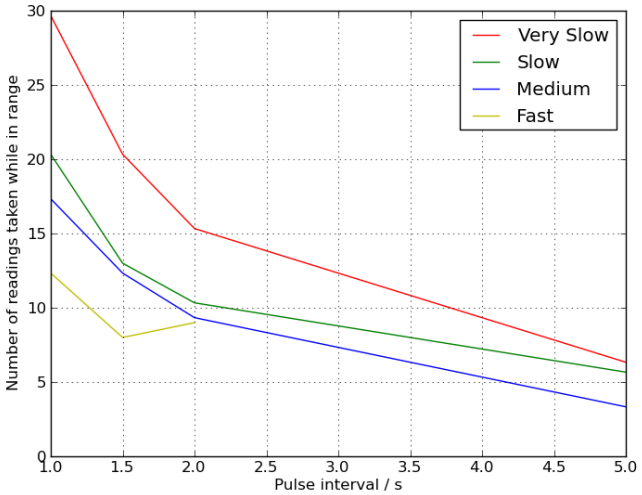


Figure 7.23: Average number of readings available (before going out of range), as measured using the Fourier transform and cross correlation at a range of pulse intervals. The four lines represent four different speeds.

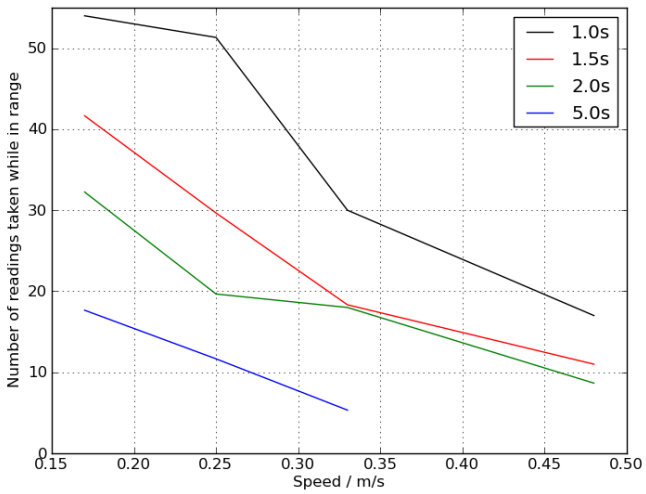


Figure 7.24: Average number of readings available (before going out of range), as measured using the Fast Chirp Transform at a range of speeds. The four lines represent four different pulse intervals. Stationary results are not included.

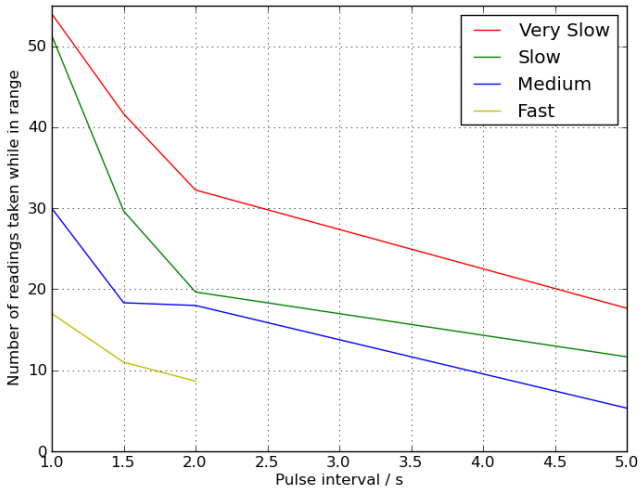


Figure 7.25: Average number of readings available (before going out of range), as measured using the Fast Chirp Transform at a range of pulse intervals. The four lines represent four different speeds.

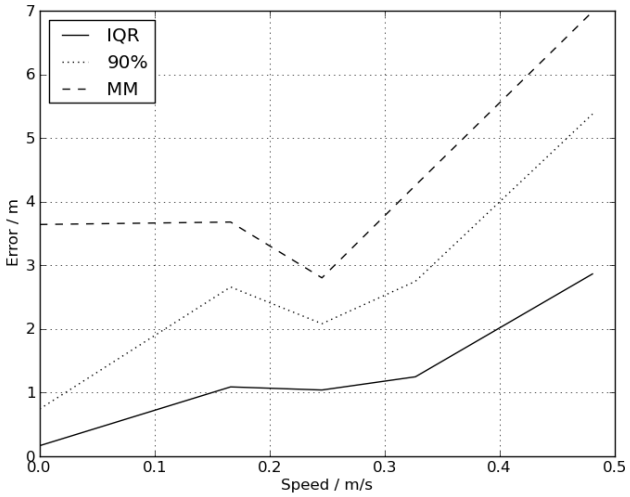


Figure 7.26: IQR, 90% and Max-Min (MM) ranges of errors, as measured using the Fourier transform and cross correlation at a range of speeds

7.6.5.2 Errors

Figures 7.26 and 7.27 show the range of the errors at different speeds and different intervals for the Fourier/cross correlation method, and Figures 7.28 and 7.29 show the same for the Fast Chirp Transform.

The error is defined as being the difference between the measured and calculated distances, where the calculated distance is obtained from the time at which the distance was measured and the speed as measured from the time taken to complete the run.

The spread of readings is considerably worse when measured using the Fast Chirp Transform, with some IQR values being higher than some Max-Min values for the other method. This is not a good sign; if the spread of data is tight then the ranging data is consistent, which means that any errors can be calibrated out. If they fall within a large spread then there is no way to mitigate the errors.

Looking at the median errors directly, the Fast Chirp Transform (Figures 7.32 and 7.33) continues to give worse results than a Fourier transform and cross correlation (Figures 7.30 and 7.31) — median errors are approximately double, ranging from 0.6–2.8m as opposed to 0.04–1.4m.

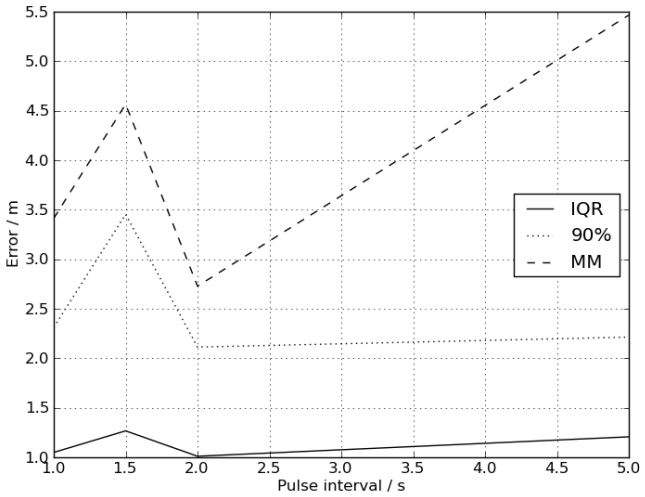


Figure 7.27: IQR, 90% and Max-Min (MM) ranges of errors, as measured using the Fourier transform and cross correlation at a range of pulse intervals

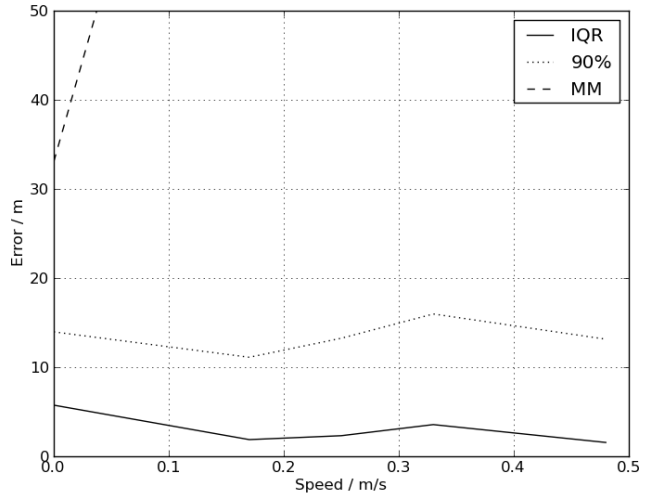


Figure 7.28: IQR, 90% and Max-Min (MM) ranges of errors, as measured using the Fast Chirp Transform at a range of speeds

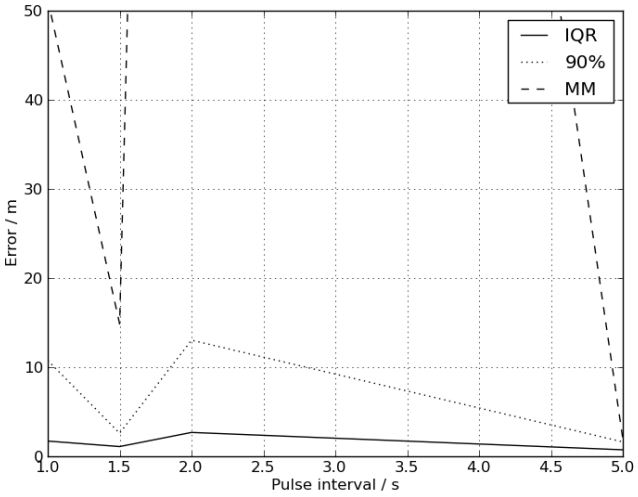


Figure 7.29: IQR, 90% and Max-Min (MM) ranges of errors, as measured using the Fast Chirp Transform at a range of pulse intervals

On all four graphs, the dotted lines represent the estimated error caused by the effect of motion on the distance measurements, as described in section 7.6. As can be seen, the errors as measured bear little relationship and show little or no correlation — this is possibly in part because the spread of errors is sufficiently high to mask any errors caused by motion.

From this, it would seem that the Fast Chirp Transform is a much worse method to use, as it consistently gives a higher error (and higher spread of errors) than the alternative method considered, despite being able to detect considerably more readings. This is due to inaccuracies in measuring the receive time of pulses, and this result is unexpected given the advantages of using the method as outlined earlier in this chapter.

7.6.5.3 Analysis

As this is an unexpected result, both the data and the processing techniques were further analysed to see if a cause could be found. There are two main places where an error could be occurring. Either the FCT is reporting chirp positions with insufficient accuracy, or the Hough transform is not detecting the lines that the FCT is

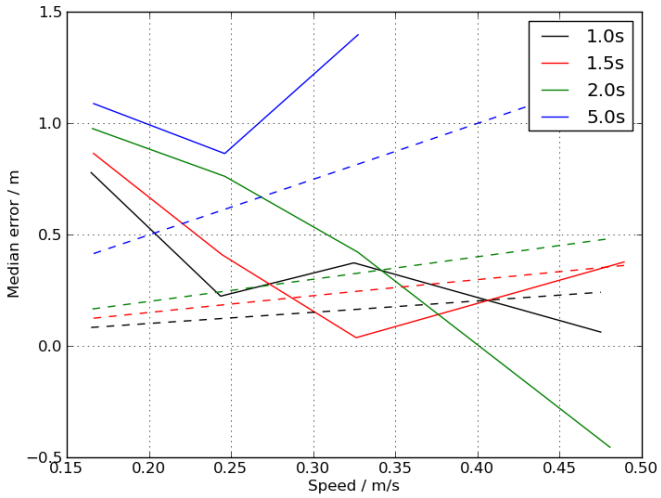


Figure 7.30: Median errors, as measured using the Fourier transform and cross correlation at a range of speeds. The four lines represent four different pulse intervals. Dashed lines show the corresponding expected errors. Stationary results are not included, as moving but not stationary values have had initial offsets taken into account.

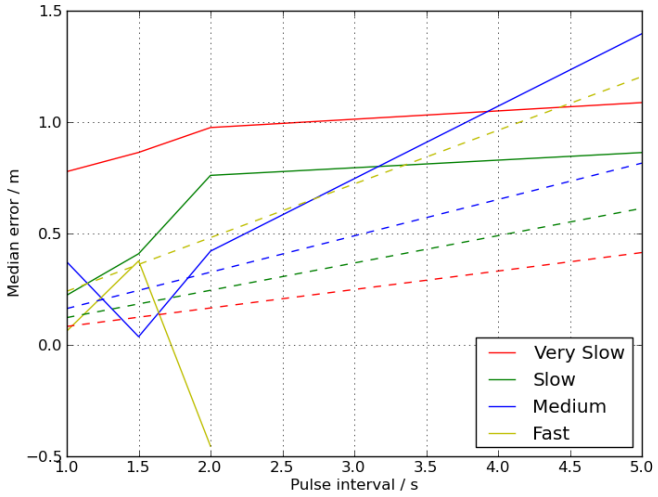


Figure 7.31: Median errors, as measured using the Fourier transform and cross correlation at a range of pulse intervals. The four lines represent four different speeds. Dashed lines show the corresponding expected errors. Stationary results are not included, as moving but not stationary values have had initial offsets taken into account.

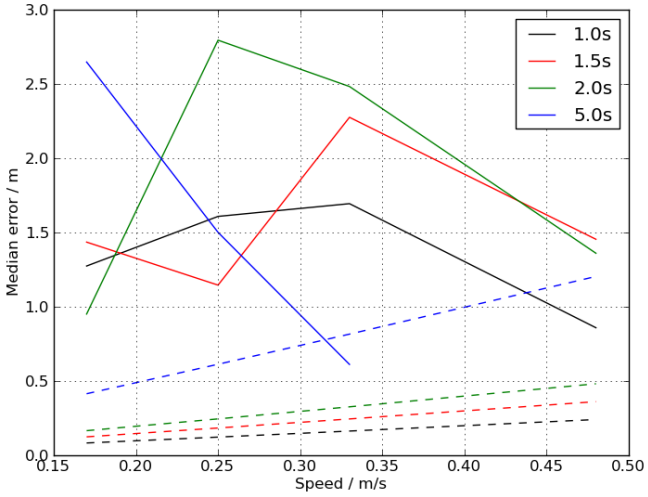


Figure 7.32: Median errors, as measured using the Fast Chirp Transform at a range of speeds. The four lines represent four different pulse intervals. Dashed lines show the corresponding expected errors. Stationary results are not included, as moving but not stationary values have had initial offsets taken into account.

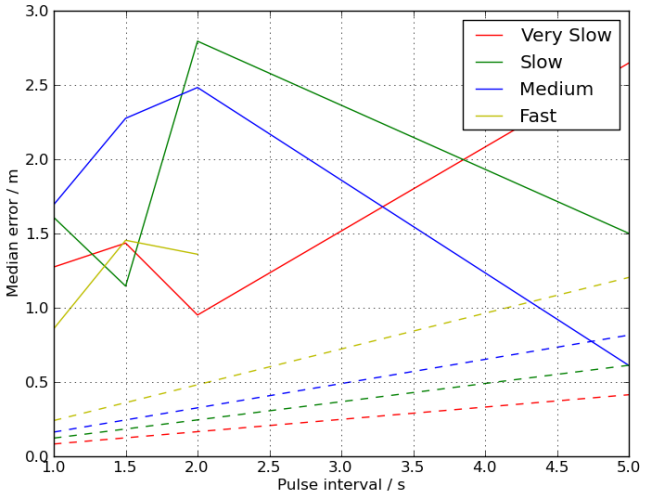


Figure 7.33: Median errors, as measured using the Fast Chirp Transform at a range of pulse intervals. The four lines represent four different speeds. Dashed lines show the corresponding expected errors. Stationary results are not included, as moving but not stationary values have had initial offsets taken into account.

outputting with sufficient accuracy to gain a correct timestamp.

Four adjacent chirps were chosen for analysis, from one of the runs at a medium speed with a 1.0s chirp interval. The distance errors calculated using these chirps are:

Time	Error
22.4240362812s	1.96637820972m
23.5714285714s	10.8473305894m
24.4829931973s	10.1606979357m
25.4965986395s	0.180799994121m

Each distance is measured using two chirps (one moving in each direction), and each is measured twice (once at each end). An error in any one chirp measurement will cause the distances using it to be incorrect. In the above table there are two errors which are over 10m; both are probably caused by the same single measurement error. The other errors around them are considerably lower.

Chirps are detected using a Hough transform, and to reduce the occurrences of the transform detecting lines which are not part of the chirp several detections are attempted for each. Chirps times are taken as being the time which is detected most often within a 0.4s period — for example, if four attempts are made to detect the line of a particular chirp and three return the same time for it, it can be safely assumed that the fourth time is incorrect. If the result is ambiguous — for example if half the results are for one time and half are for another — then the time is chosen based on the strength of detection, as reported by the Hough transform. (It should be noted that the Hough transform will be returning the strongest line visible, so it is most likely for it to detect the chirp over any other noise in most cases).

As the 0.4s window is relatively wide (in order to discount as many spurious readings as possible) there are likely to be a number of incorrect times listed for each chirp alongside the correct measurements — some from false detections, some from multipath, and some from artefacts caused by a strong signal.[13] Each chirp has four opportunities to be detected by the Hough transform, although that does not

[13]These can be seen in Figure 7.8, where the diagonal line appears to have two 'boxes' attached to either end.

mean that it will be detected by every measurement, and as 0.4s is a wide window more than four readings can be included for each.

The four chirps listed above were each detected several times, as shown in Fig. 7.34. Most had the same measurement made in almost all cases — for example Chirp 3 at unit A, which has three measurements at the same place and one spurious measurement elsewhere — implying that the measurement was correct. Some were more ambiguous — for example Chirp 2 at unit A, which has four close timestamps of which the highest has only one more reading than the others, or Chirp 1 at unit A, which has two close timestamps both with the same number of readings. The figure includes distances for each measurement, which correspond to the change in measured distance that it represents compared to the chosen measurement (the distance travelled at the speed of sound in the difference in times).

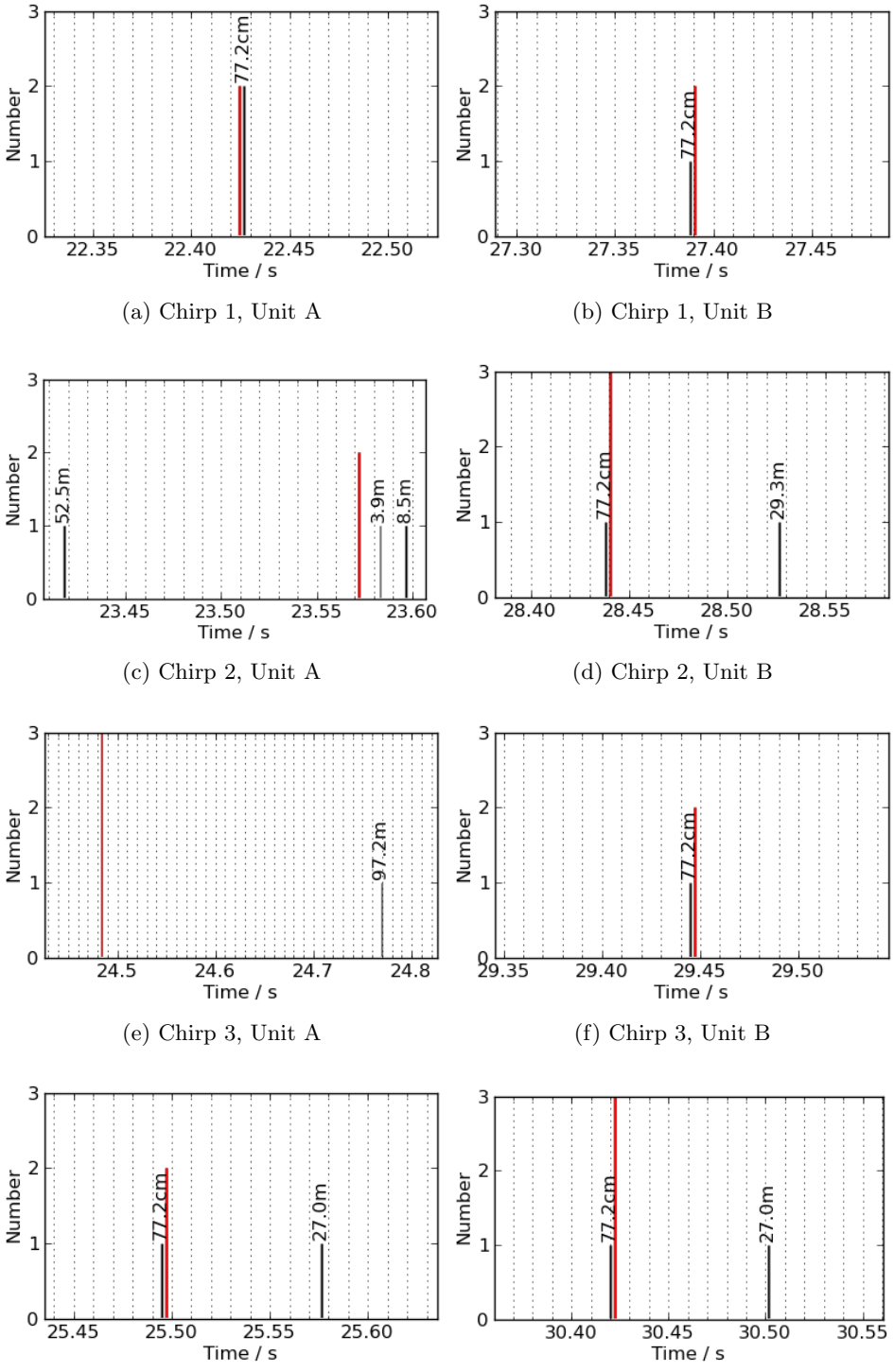
As can be seen, there is a large opportunity for error, and if (for example) the reading at 23.571s at unit A was instead taken as 23.596 — a change of under 25ms, to a time which had only one fewer detection — then that would account for much of the 10m error seen in the results.

This is a single case where a large error is being considered, but the exact same problem will be occurring elsewhere on a smaller scale — giving smaller inaccuracies, but still introducing errors. It seems probable that this is the reason why the FCT is giving a worse response in this situation than the other method tried. Future work should be undertaken to attempt to increase the accuracy of line detection — either through increased measurements (increasing from four Hough transforms per chirp will increase the time to analyse the data, but will should give an increase in correct timestamps), or by reducing the number of false positives (reducing the number of incorrect timestamps).

7.6.5.4 Further limitations

There are several other limitations related to the test setup as described above, which may also account for some of the errors seen.

As described above, the mobile unit was moved directly away from the stationary unit at a constant speed. It travelled strapped to a small trolley on castors, running



(a) Chirp 1, Unit A

(b) Chirp 1, Unit B

(c) Chirp 2, Unit A

(d) Chirp 2, Unit B

(e) Chirp 3, Unit A

(f) Chirp 3, Unit B

(g) Chirp 4, Unit A

(h) Chirp 4, Unit B

Figure 7.34: Four chirps (medium speed, 1.0s pulse interval) with their Hough detection times. The line in red shows the timestamp used. The other lines are marked with the approximate distance change if this was used instead.

on wires for the full length of the corridor. It was started at a set position, allowed to settle for 10s (to ensure that the starting position was calibrated), and then moved a set distance. The time taken to complete each individual run was timed.

The most notable source of error from this is that of timing. The start and end times were measured by a human with a stopwatch. From this was calculated the speed and, more importantly, the time at which the unit started moving. Although the times are relatively consistent, there is some scope for this causing errors. While the speeds are fairly tolerant to timing errors the start time is not, and the variation in start times for each speed is relatively high.

The speeds measured do not take into account acceleration, particularly acceleration from stationary at the start of the run. Although this takes very little time, it cannot be truly instantaneous. This will further add error to the measurements.

Figure 7.35 shows the distance as calculated by the Fast Chirp Transform for all pulses detected by both units, plotted against time. This shows four lines at different gradients representing the four speeds, as well as many readings which are away from these lines. Interestingly, at the start of of each of these lines they do not appear to be straight — they show a curve, which is possibly due to the acceleration of the moving unit from stationary.

The speeds at which the unit travelled for this test are notably at odds with the speeds as described previously. The maximum speed tested was 0.48m/s, while previous discussion used two example speeds — that of a walking human, at 1.5m/s, and that of a UAV at maximum velocity, at 20m/s. It is clear that using higher speeds would make the errors due to movement more noticeable, and would probably mean that they were larger than the experimental errors. The reason for not running at a higher speed is that the units go out of range very quickly; the number of readings available while in range would drop further if higher speeds were used.

Another potential source of error is from multipath. A signal transmitted by one unit will be received directly by the other, giving a time of flight of the distance between them; but the other unit will also receive the signal as reflected off any objects in between — walls, floors, ceilings, etc. This second signal will give a time of flight of the distance between the first unit and the wall, and the wall and the

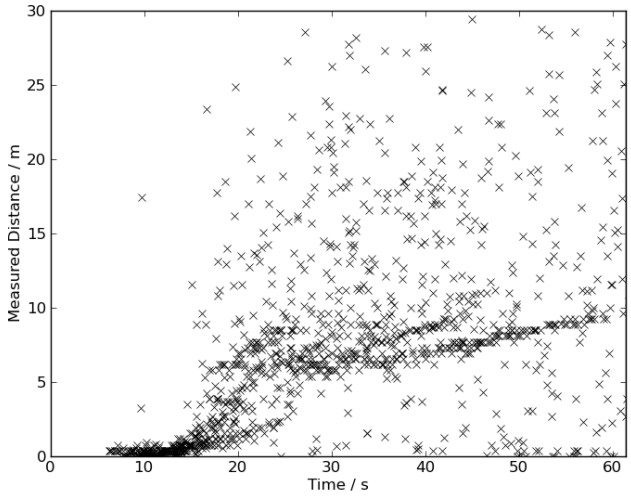


Figure 7.35: Distances as measured using the Fast Chirp Transform, plotted against time for all pulse intervals and all speeds

second unit. In order to mitigate the effects of multipath a relatively wide corridor was selected for the experiment, and the units were programmed to take timestamps from the strongest signals received. This is not always a sure means of removing multipath, as the reflected signal can sometimes be stronger than the direct signal, but as there was a direct line of sight between the two units in the experimental setup it is likely that the stronger signal is the correct one. However, despite these precautions the errors from the stationary experiment as shown in Figure 7.13 seem to imply that multipath had had an effect on the measurement accuracies. Errors of up to 2m from sources other than pulse interval or speed (the two variables being tested) would certainly be able to skew the results so as to mask the results being looked for.

7.7 Conclusion

An analysis has been made to compare two different methods of detecting pulses for a relative positioning system based on round trip time of flight measurements. While using the Fast Chirp Transform would appear at first to be a good choice, as it gives a much better detection rate than using a Fourier transform and cross

correlation, it appears to give worse results in terms of timing accuracies. This means that the distances it calculates are not very close to the distances as measured, and the errors are widely spread. This may largely be due to the use of a Hough transform for converting the FCT output into a timestamp, which appears to be giving measurements that vary in accuracy. Future work is required to attempt to improve this.

A discussion was covered regarding errors caused by movement giving a varying round trip time. The errors which this would cause are less than the errors as seen in the experiments run, so a future experiment would be required to give a conclusive answer to this.

Errors may be caused (as described above) by many sources, falling into two main groups — errors caused from timing measurement of pulses, and errors caused by the experimental setup.

Future experiments in better conditions, reducing other error sources, may show more accurately the errors involved. The use of laser gates to ensure accurate timings, automated speed controls or an alternative means for accurately measuring the position would remove the timing and speed measurement errors. Audio damping on all surfaces would reduce the possible effect of multipath. An improved experimental setup may also allow for a greater range between the two units, which would allow for longer runs and give more data for better results.

The results shown do imply that this form of audio based relative positioning would be of little use in many mobile situations, unless all units are liable to remain within 10m of one another. However, the reason for choosing this method was that it was simpler to build and test than an RF based system, and the same theory can be used for radio based relative positioning which would have a considerably higher range (albeit with a higher resolution based error).

Despite this, the results have shown that the pulse interval does not greatly affect the accuracy — particularly when travelling at lower speeds, where units take longer to go out of range. Spreading out the pulses brings many benefits. Fewer pulses means lower power consumption, which is useful in mobile devices where power is at a premium. It also reduces noise pollution — there is less risk of pulses overlapping

and being missed, and the units will have less effect on anything else using the same frequency (including humans, if — as in this experiment — the pulses are audible). The ability to cope with more intermittent pulses will also help in noisy environments, where many pulses may be undetectable and only a small percentage can be used for calculating distances.

The original requirement for a positioning system was for use with UAVs. The system described here would not be suitable for this, for the reasons given above. The PinPoint[233] system would be suitable in terms of range and accuracy, but the frequent pulses it uses could be improved upon. It also has not been tested at the high speeds possible with a UAV. Future experiments are required using an RF based system such as PinPoint but with intermittent pulses and high speeds.

Chapter 8

Conclusion

This thesis has introduced a number of novel research concepts which could be applied in the future to improve UAVs, particularly those used in land-based search and rescue. While the things developed can't — for the most part — be taken immediately out into the field and used to improve land-based SAR, some aspects (such as the work on diffusion) could be implemented with relatively little extra work.

Here follows a conclusion broken down by chapter, outlining the main contributions made.

Chapter 2 discussed the field of Search and Rescue, covering the creation of probability maps and methods for searching across them. It described the ways in which land-based search and rescue is usually performed at the moment, where search areas are often large with travelling time assumed negligible compared to search time. The method for searching these areas is often up to the people on the ground, although some methods for area searching are covered. This could be improved for UAVs by splitting the search area up into smaller cells — making the time taken to search an area negligible compared to the time taken to travel between them. This allows for faster changes in probability, and having a large number of small areas means that there is never a need to change area boundaries. Some UAVs use similar probability maps, but the algorithms used to cross them are poor, in that they can easily get caught on local maxima.

Chapter 3 described an experiment to determine a method for searching a map, such that areas with a higher probability are searched in the minimum time possible — without getting distracted by either local maxima which are close but globally low[1], or by global maxima which are at a great distance at the expense of closer near-maxima. While aimed primarily at covering probability maps for search and rescue this could equally be used in many other applications, such as a cleaning or painting robot.

The method under consideration used diffusion algorithms to inform the robot which was the best route to reach the high valued areas. It was compared against boustrophedonic sweep searching (a brute force method which did not take into account probability levels) to provide a base line, and multiple diffusion equations were tested against each other — comparing both the speed at which high probability areas were searched, and the processing time required.

Diffusion proved in general faster than a boustrophedonic search for high probability areas, but as a consequence — as was to be expected — a slower result for completely searching the entire map. Several different diffusion equations were compared, but the difference in the results were relatively small. A larger difference was that some methods were able to get stuck, while others would always complete. The equations which proved most effective were not the same as those which were least computationally intensive. When both criteria are important, the Median or Highest Neighbour algorithms are the best two options.

Future work in this includes introducing non-static base maps, allowing for obstacles, optimising program code to reduce processing time, and — most importantly — extending the system to allow for multiple UAVs which are able to take each other's plans into account.

Chapter 4 covered UAVs in general, and UAVs for disaster management and search and rescue in particular — describing the work of a number of research groups worldwide, and explaining how some systems are used in emergency situations such as search and rescue. This gave a review of the current state of research in UAVs, allowing the reader to see how realistic it would be to create a swarm of autonomous,

[1] As might be found by a greedy lookahead algorithm, as described in Section 2.3.3

collaborative UAVs for search and rescue.

Chapter 5 documented the creation of the Durham Tracker Project, covers a range of improvements made to it, and compares it experimentally against other commercially available GPS trackers. It proved itself to be as accurate if not better than the alternatives. The tracker created is a fully working platform for future positioning experiments, and it supports two way interfacing. This means that positional data and remote instructions could be sent to a UAV controller, with status information being sent back. The UAV would have a location for route planning, and the people back at base would not only have regular updates on the current situation and location but would also be able to send instructions to the UAV over the mobile phone network. This could be used as part of a multi-level control strategy, where the UAV displays varying levels of autonomy depending on the instructions available.

Chapter 6 discussed the accuracy of GPS positions, focussing on GPS altitude measurements. It aimed to quantify this by comparing GPS measurements against the SRTM dataset. The results showed that the errors produced were in the majority of cases considerably higher than those given by the FAA in controlled conditions (a result to be expected, as the majority of errors are situational). In most situations a separate altimeter is therefore to be recommended, although barometric altitude errors — being more susceptible to weather related pressure fluctuations — may be worse than GPS in some cases (over long periods, or during storms).

As the Open StreetMap data is not entirely clean,[2] the results obtained have scope for inaccuracies. A follow-up study over a more limited area using hardware under the control of the researchers, and comparing against higher accuracy digital elevation models, would give an indication as to the accuracy of the data used here — it would, however, give a considerably smaller set of data to use. The OSM elevation data makes for a useful and very large dataset which would be an interesting base for further experiments.

Chapter 7 contains details of the creation of a relative positioning system which uses infrequent pulses to save power and noise pollution and allow for interference.

[2]Data is not available to allow for the removal of cases where the GPS receiver used had a separate altimeter, nor those where the data was postprocessed to change the geoid used

There were two aims in this experiment. The first was simply to create a relative positioning system which could be used by a swarm of collaborative UAVs in harsh environments. The second was to investigate the errors caused by the assumption that all units are stationary during the ranging process, when the ranging process is long (infrequent pulses).

Unfortunately the experimental results obtained are somewhat inconclusive. They neither prove nor disprove the effects of motion, as the errors from this are negligible in comparison to other errors (mainly stemming from the experimental setup). Future experiments in better conditions where other error sources are reduced will therefore be necessary in order to investigate this further.

Although the system created proved unsuitable for UAVs, the theory behind it is one which warrants further investigation, and the experiments have shown that infrequent pulses can be used to make relatively accurate measurements. An eventual relative positioning design should aim to use the same equations but using an RF based system. This would give a far longer range, although the resolution based errors would increase.

8.1 Thesis

Land-based Search and Rescue can be substantially improved through increased use of technology, and through use of UAVs in particular.

To return to the original thesis and hypotheses, the overview of Search and Rescue in Chapter 2 shows that the field of land-based Search and Rescue is — from a scientific point of view — still in its infancy, and could be improved through increased use of technology. The overview of UAVs in Chapter 4 describes several research groups which are working on search and rescue or related areas. This shows that the subject — autonomous, collaborative UAVs for search and rescue — is one that is ripe for research.

1. Improving algorithms — Current search strategy is suboptimal for use with a swarm of collaborative UAVs, and can be

improved upon

Current methods for searching across a probability map are suboptimal for use with a swarm of collaborative UAVs. Chapter 2 describes some of these (for example methods which are designed for searching large areas of probability with teams of slow-moving humans, or greedy algorithms which don't take into account global maxima). Chapter 3 provides a method to improve upon these, giving a diffusion based algorithm which optimises a search route to visit the highest probability areas across the entire map in the shortest possible time — taking travelling time into account.

2. Improving hardware — Autonomous, collaborative UAVs for search and rescue could be developed which would improve the work of SAR groups

In terms of improving hardware, Chapter 5 covers the creation and improvement of the Durham Tracker Project. This could be used for many applications, and two of these are most applicable to this thesis. The first is as trackers for human searchers (to allow the central control to know exactly where the tracking party has been, and so the areas which have been searched, without having the difficulties inherent with first finding the exact areas to search and then reporting back an accurate summary of the coverage achieved). The second is as part of the control system for a SAR UAV — not only giving positioning information to the autopilot, but also reporting back to base regular information about the UAV's location with any other status data (both from sensors, including things like temperature and battery levels, and from the autopilot, including things like flight plans and decisions).

3. Improving localisation — UAVs require accurate positioning in order to successfully complete their tasks, and GPS can be improved upon in some cases

Chapter 6 describes some of the inaccuracies inherent with using GPS — in particular for altitude measurements. As accurate positioning is often a strong requirement for UAVs in order to successfully complete their tasks, but in many

cases only relative positioning is required, Chapter 7 describes experiments into a UAV relative positioning system which takes into account their speed of flight.

8.2 Closing remarks

This work set out to improve the efficiency of search and rescue teams by producing research into the creation of a swarm of autonomous, collaborative, unmanned aerial vehicles. This has been to a great extent successful.

Although much future work is required before the final vision of this project is realised — the creation of a swarm of collaborative UAVs that fly entirely autonomously and which can undertake SAR missions and save lives — the work documented in this thesis will, I hope, bring this vision closer to reality.

Appendix A

Radio Distress Beacon

A radio distress beacon is a device which transmits a signal to show that the person activating it is in trouble and requires assistance.

The most prevalent radio beacon system currently in use is the Cospas-Sarsat Distress Beacon.[247] This comes in three types — EPIRBs (Emergency Position Indicating Radio Beacons) for maritime use; ELTs (Emergency Locator Transmitters) for aircraft use; and PLBs (Personal Locator Beacons) for personal use.[13; 5]

The original standard for these transmitted a distress signal (constant tone) at 121.5 MHz (this frequency is still in use for emergencies, and is reserved for emergency communication — whether automated sirens or voice communications). This signal was then picked up by one of a series of satellites. The satellites used the Doppler shift to calculate a position, based on their own known speed and position. One pass gave two possible positions, and a second pass pinpointed the transmission to within 20km. This standard was discontinued from February 2009.

The new standard transmits a Hex encoded distress message at 406MHz, containing information such as a unique beacon identification number (to allow the search and rescue teams access to pre-registered details such as a phone number to check with the user for false alarms), country of origin, GPS location (where available — many new standard beacons contain GPS receivers as well as Cospas-Sarsat transmitters),[1] and whether the beacon contains a 121.5MHz transmitter (the 121.5MHz

[1] Interestingly, the staff at RAF Kinloss say that the doppler derived position is often considerably more accurate than the GPS position when it is available

transmitter is often retained as a homing beacon, as although it is no longer monitored by satellite it is still a designated emergency frequency). This gives several benefits over the older system. As a definite message is transmitted false alarms from unconnected transmissions are minimal. The unique identifier helps with pinpointing location. Position can now be pinpointed to within 5km with only one pass. Detection and response time are reduced to approx. 4–10 minutes, and the SAR teams now have an opportunity to check with the beacon's owner before departing. Newer models also transmit in bursts to save power, allowing them to transmit at 5W as opposed to the 0.1W of the older system.

Two constellations of satellites are used in the system. LEOSAR (Low Earth Orbit Search and Rescue), the earlier system, is composed of five low Earth orbit satellites. GEOSAR (Geostationary Earth Orbit Search and Rescue), the second constellation, is composed of a further five satellites in fixed positions. This is to be further complemented by receivers on GPS and Galileo satellites, which will greatly reduce detection time.

The message from the satellite is passed on to the Mission Control Centre (MCC) in charge of the relevant Search and Rescue Region. In the UK and surrounding waters (including a large area of ocean) this is the RAF's Aeronautical Rescue Coordination Centre (ARCC) at Kinloss.

Similar systems also exist, such as the SPOT trackers with emergency buttons which pass GPS positions via a GEOS International Emergency Response Centre to the relevant country's search and rescue organisation.[2]

[2]In the UK this is also RAF Kinloss

Appendix B

Missing person types

These are the categories of person as described in the International Search and Rescue Incident Database (ISRID):

- Abduction
- Aircraft
- Angler
- ATV
- Autistic
- Camper
- Caver
- Child
  - Child 1–3
  - Child 4–6
  - Child 7–9
  - Child 10–12
  - Child/Youth 13–15
- Climber
- Dementia
- Despondent
- Gatherer
- Hiker

- Horseback Rider
- Hunter
- Mental Illness
- Mental Retardation (Intellectual disability)
- Mountain Biker
- Other
  - Base Jumper
  - Extreme Sports
  - Motorcycle
- Runner
- Skier-Alpine
- Skier-Nordic
- Snowboarder
- Snowmobiler
- Snowshoer
- Substance Abuse
- Urban Entrapment
- Vehicle
  - Missing vehicle
  - Four-wheeled drive
  - Abandoned vehicle
- Water
  - Powered boat
  - Non-powered boat
  - Person in water
    - Flat water
    - Current
    - Flood stage
- Worker

For comparison, these are the categories of person as described in Mountain Rescue England and Wales' Missing Person Behaviour Study Report 2011:[19]

- Children aged 1 to 16[1]
- Climber
- Dementia
- Despondent
- Development problems
- Fell runner
- Health related
- Miscellaneous
- Mountain biker
- Organised party
- Other vulnerables
- Psychological illness
- Substance related
- Walker
- Water related

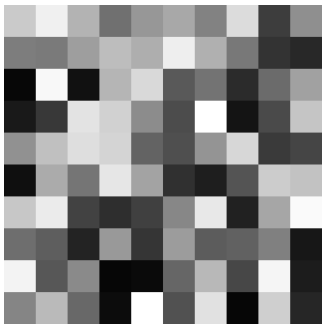
[1] In previous reports this was split into three: Children aged 1 to 6 years; Children aged 7 to 12 years; and Youths aged 13 to 16 years

Appendix C

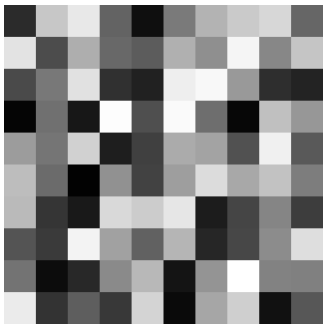
Base map images

There were two sets of Base Maps, as described in Section 3.3.

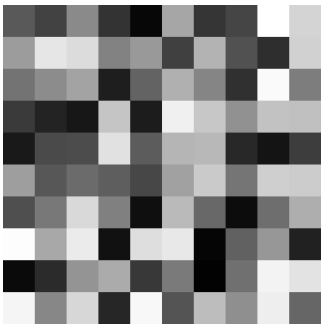
C.1 Set A — Randomly generated



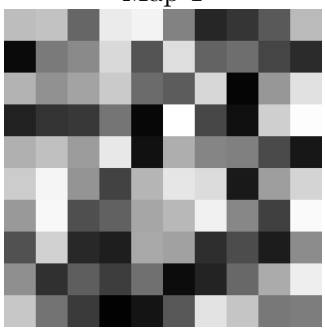
Map 1



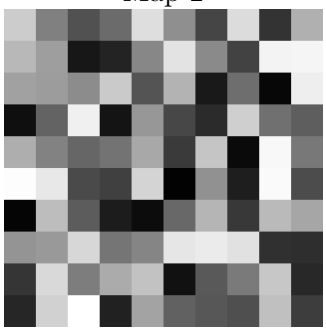
Map 2



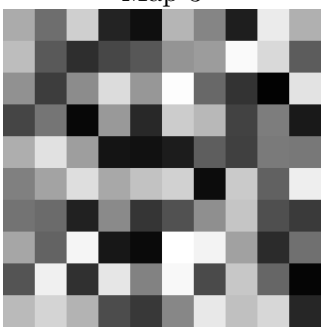
Map 3



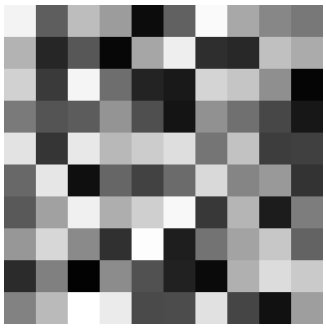
Map 4



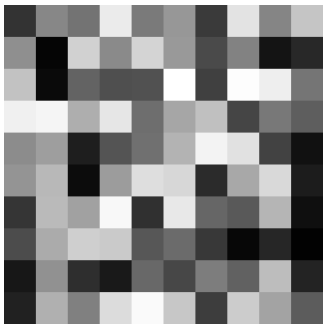
Map 5



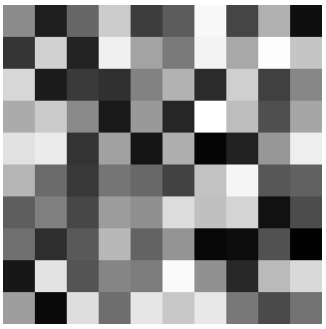
Map 6



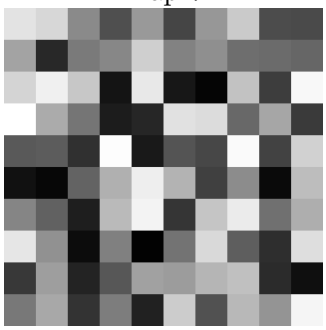
Map 7



Map 8



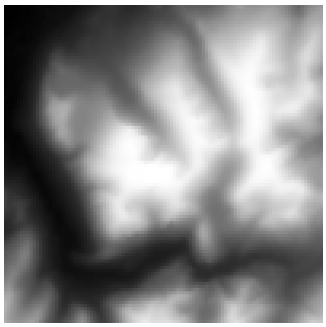
Map 9



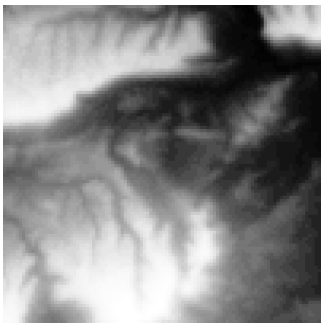
Map 10

C.2 Set B — Based on NASA SRTM data

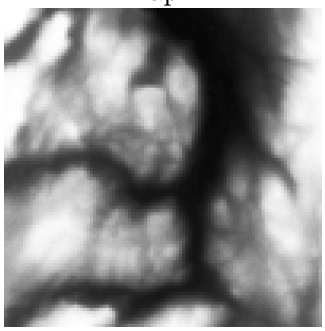
Map 1



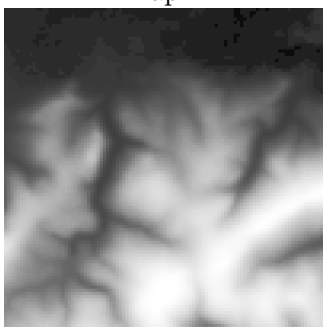
Map 2



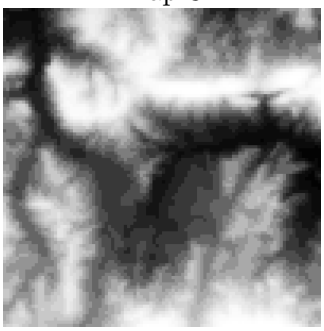
Map 3



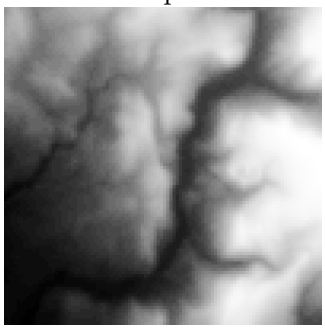
Map 4



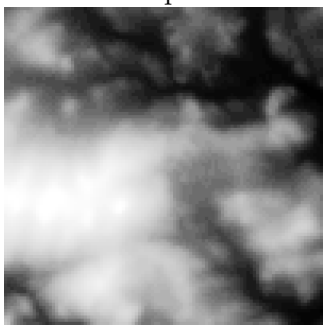
Map 5



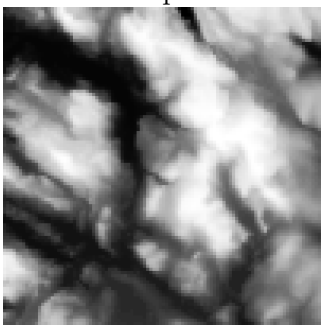
Map 6



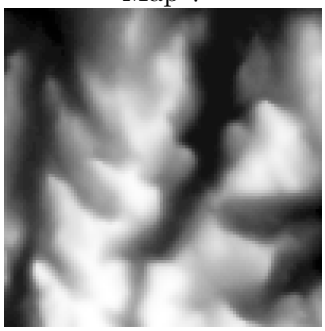
Map 7



Map 8



Map 9



Map 10

Appendix D

Diffusion Results

D.1 Highest Neighbour

Fig D.1 shows the results from the imperfect cleaning simulation for both map sets. The lines show the time at which each of the first 500 'people' on the map were found. As can be seen, all of the diffusion lines (D in equation 3.1) are relatively straight, implying that their ability to find people is at a relatively constant speed. It would be expected that the number of people found would drop over time, as the probability of success reduces across the map; the 500 person threshold is potentially too low to demonstrate this. The low diffusion rates show a consistently high time for finding people, with the best results over the 500 being shown by the high levels of diffusion (95% and 99% both performing well). Similar graphs are produced for both of the individual map sets.

The 10% curve closely matches the 50% curve. The reason for this is that in both cases the diffusion is comparatively low. Taking the case of a single cell with probability 100 in a map which is otherwise at 0, a diffusion of 10% would give (moving radially outwards) probabilities of 10, 1, 0.1, 0.01, etc. Depending on the threshold used this would cease to be discernible relatively close to the initial cell — the distance would be further reduced if, as is usually the case, its neighbours are of a closer probability. The 10% curve therefore represents the bare minimum in diffusion — little more than a hill climbing algorithm over the initial probability map. Given the same situation of a single cell with a probability of 100 in a map

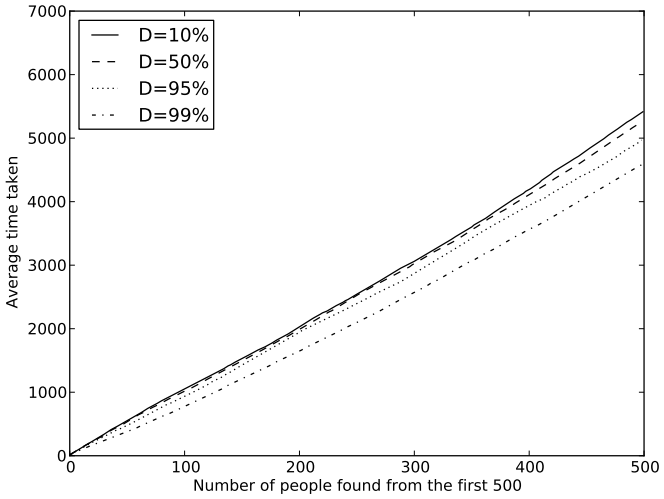


Figure D.1: The time taken to find 500 people, averaged over 20 base maps for four levels of diffusion (D from equation 3.1) for the Highest Neighbour method

otherwise at 0 it would take 4 cells for the probability to drop below 0.1 at a diffusion of 10%, 10 at 50%, 135 at 95%, and 688 at 99%. These would be greatly reduced if the neighbouring cells were not at so great a difference in probability.

Fig D.2 shows the box plots of the times to find 500 people using this method. Again it can be seen that the speed improves as the diffusion level increases. The ranges of the results remains relatively constant between variable levels.

The equation for the Highest Neighbour method (from equation 3.1), at the best recorded diffusion level, is:

$$l_t(i,j) = \max \begin{cases} 0.99 l_{t-1}(i^N, j^N) \\ L_{t-1}(i,j) \end{cases} \tag{D.1}$$

D.2 Average

Figure D.3 shows the same simulation, but run using equation 3.3 for the creation of the Navigation Map. The four reduction values tested have given near identical responses, with a marginal trend towards lower percentages giving faster times. Similar results are shown for each of the map sets individually. This shows that

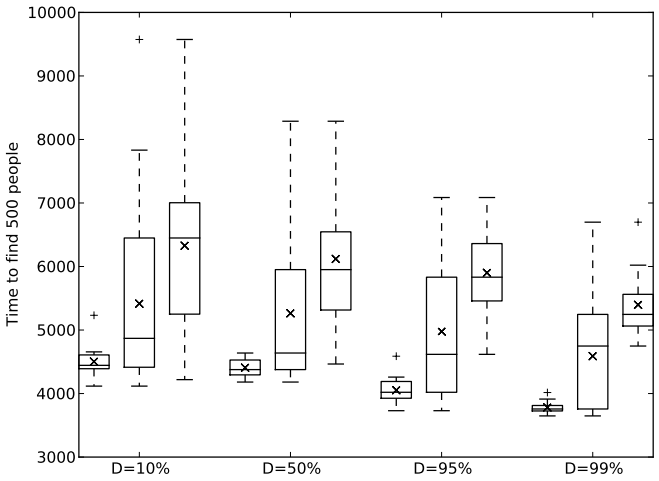


Figure D.2: The time taken to find 500 people using the Highest Neighbour method, averaged over 20 Base Maps for four levels of diffusion (D from equation 3.1). The three plots show results from Set A, all maps, and Set B respectively

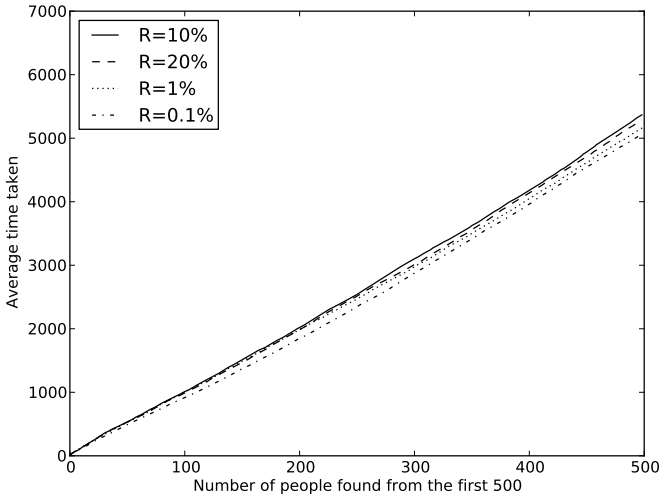


Figure D.3: The time taken to find 500 people, averaged over 20 base maps for four reduction values (R from equation 3.3) for the Average method

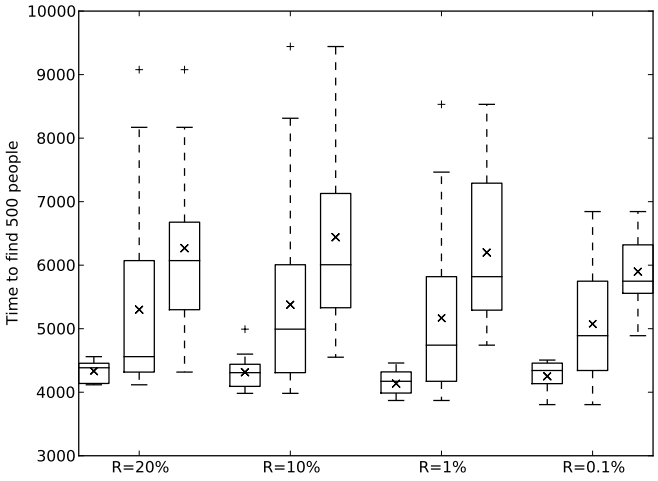


Figure D.4: The time taken to find 500 people using the Average method, averaged over 20 Base Maps for four reduction values (R from equation 3.3). The three plots show results from Set A, all maps, and Set B respectively

while the closer the navigation values are set to being the true average of their neighbours the better the response, over the range tested this improvement is only marginal.

Fig D.4 shows the box plots of the times produced by this method. The box plots change relatively little between variable levels; the ranges are shorter for Set A, but are mainly similar across the two map sets.

The equation for the Average method (from equation 3.3), at the best recorded diffusion level, is for Set A:

$$l_t(i,j) = \max \begin{cases} \frac{(1-0.01)\sum l_{t-1}(i^N,j^N)}{8} \\ L_{t-1}(i,j) \end{cases} \tag{D.2}$$

And for Set B:

$$l_t(i,j) = \max \begin{cases} \frac{(1-0.001)\sum l_{t-1}(i^N,j^N)}{8} \\ L_{t-1}(i,j) \end{cases} \tag{D.3}$$

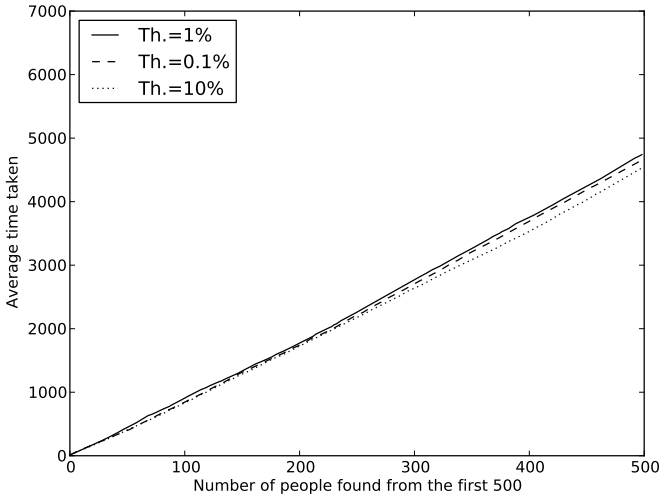


Figure D.5: The time taken to find 500 people, averaged over 20 base maps for three threshold values for the Median method

D.3 Median

The results of the Median method are quite interesting. The simulation was run with thee threshold values — 10%, 1% and 0.1% — where the threshold value represented the amount of change below which a cell would cease to propagate its value to its neighbours. Figure D.5 shows the averaged results for all maps from both map sets, with the two map sets A and B being shown in Figures D.6 and D.7 respectively.

In Set A, the results from thresholds of 10% and 1% are almost identical, with 0.1% being faster. It would therefore be consistent with the results to say that the smaller the threshold the faster the response.

In Set B, the results are by comparison considerably further spread out. They are also in the opposite order — with 0.1% being the slowest and 10% being the fastest.

This inconsistency averages out somewhat across the two map sets, such that the positions in Figure D.5 are not particularly far from each other. The change is likely to be a product of the differences between the two map sets. A threshold of 10% will be producing navigation maps which are not substantially changed from their corresponding base maps — the algorithm will not have been able to make as

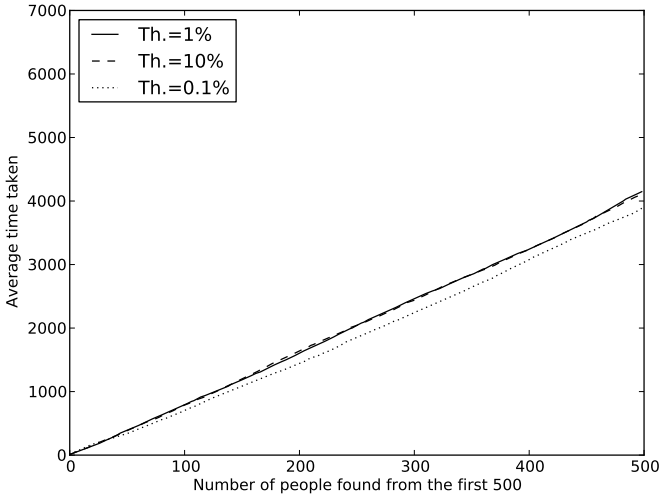


Figure D.6: The time taken to find 500 people, averaged over the 10 maps of Set A for three threshold values for the Median method

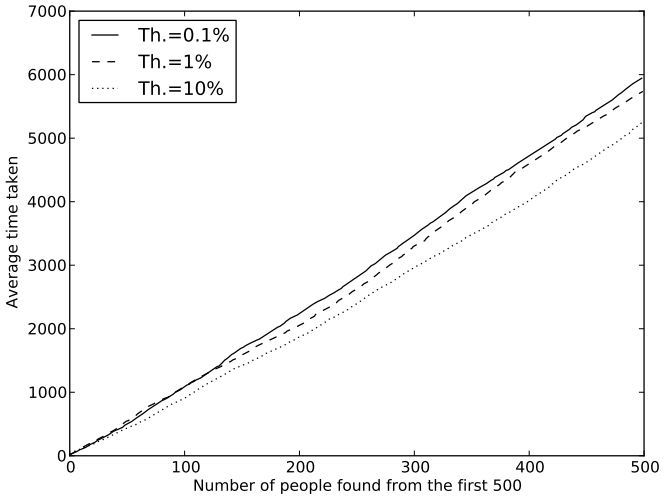


Figure D.7: The time taken to find 500 people, averaged over the 10 maps of Set B for three threshold values for the Median method

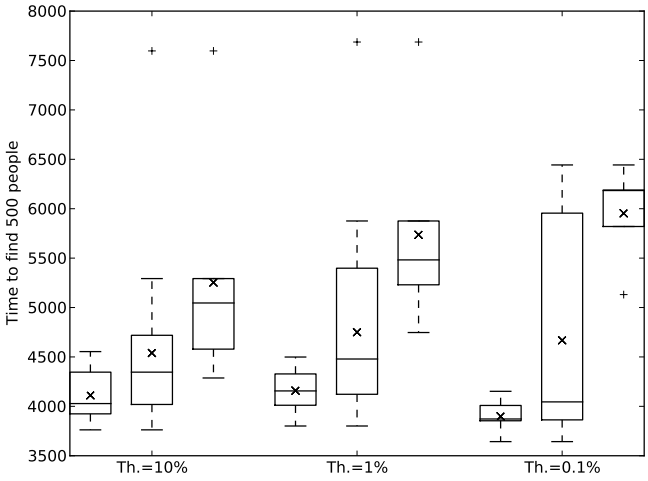


Figure D.8: The time taken to find 500 people using the Median method, averaged over 20 Base Maps for three threshold values. The three plots show results from Set A, all maps, and Set B respectively

many changes as would have been possible with lower thresholds. The first map set has base values which are assigned randomly — outside of the 100 cells at the same value there is no reason to assume that any neighbouring cells are at a similar value. In the second set, however, it is more likely for any cell to contain neighbours of a similar base value. As there are naturally occurring 'hills' that may be climbed, the Median method with a high threshold value might get a better response on these maps than those created at random.

Looking at the box plots in Fig D.8, the plots from Set A are rather less clear as to the effect of the threshold value, with 0.1% being slightly better. The plots from Set B show a much more definite trend, with the higher percentages giving better results.

It should be noted that threshold values from all methods refer to the propagation of values. Cells will be updated only if one of their immediate neighbours have been changed since it was last updated; a neighbouring cell is only considered to have changed if the difference between its current value and its value at the previous iteration is greater than the threshold. A high threshold will therefore stop values from propagating to as great an extent as a low threshold.

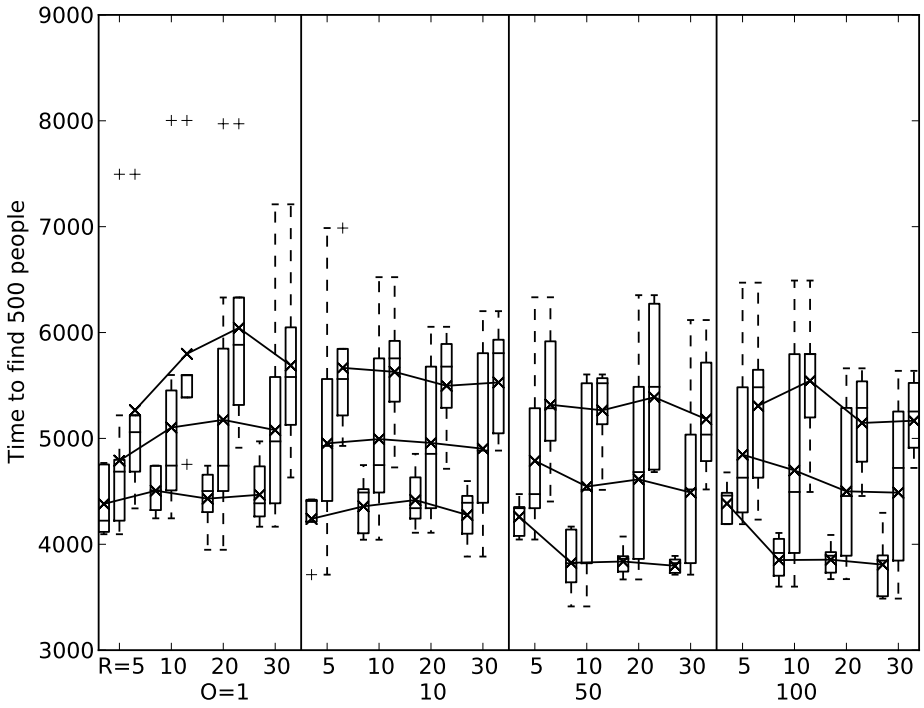


Figure D.9: The time taken to find 500 people using the Absolute Distance method, averaged over 20 Base Maps for four levels of radius and four offsets (O and R from equation 3.5). The three plots show results from Set A, all maps, and Set B respectively

D.4 Absolute Distance

Comparing the results from changing offsets and radii (as shown in the boxplots in Fig D.9), the best radius (R in equation 3.5) for a given situation appears to depend on the current offset (O in equation 3.5) and vice versa. This is because the offset changes whether the robot prefers close cells or high valued ones. If it has been set to greatly prefer close cells, a high radius won't make any difference, so for low offsets the lower radii will provide faster responses. For higher end values of offset, it will be looking far afield and caring a lot less about the distance — to a greater extent than is necessary for some of the radii. Changes in offset will therefore matter less in this context, as it will have saturated the distance; it will know the locations of all of the high points in the area, and the ordering of them will not change beyond

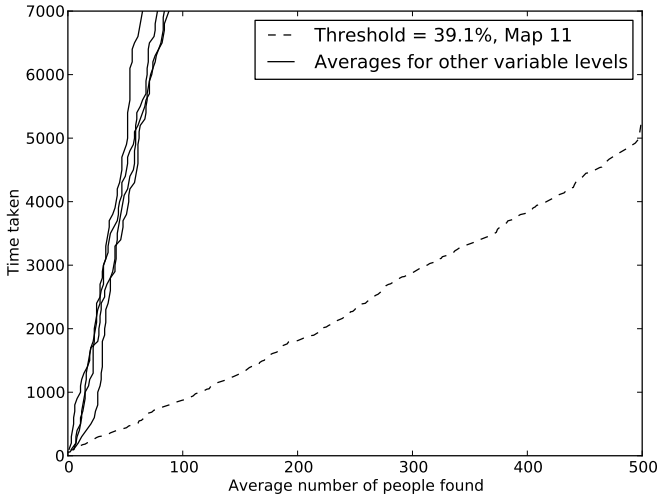


Figure D.10: The time taken to find 500 people, averaged over 20 base maps for four threshold values for the Low Pass Filter method. Failed to complete all but one iteration, which is shown separately and is not included in the other averages.

a certain level. It is therefore important to pick a good match between radius and offset.

The equation for the Absolute Distance method (from equation 3.5), at the best recorded diffusion levels for Set A, Set B, and all maps respectively, becomes:

$$l_t(i,j) = \max\left\{ \frac{L_{t-1}(i',j')}{k+50} \middle\| \text{ where } k \leq 30 \right. \tag{D.4}$$

$$l_t(i,j) = \max\left\{ \frac{L_{t-1}(i',j')}{k+100} \middle\| \text{ where } k \leq 20 \right. \tag{D.5}$$

$$l_t(i,j) = \max\left\{ \frac{L_{t-1}(i',j')}{k+100} \middle\| \text{ where } k \leq 30 \right. \tag{D.6}$$

D.5 Low Pass Filter

The Low Pass Filter method regularly became stuck in simulations. The reason for this is likely to be due to artefacts from the Fourier transform. As these are caused by the base values of cells from different places, when the robot reaches one of these

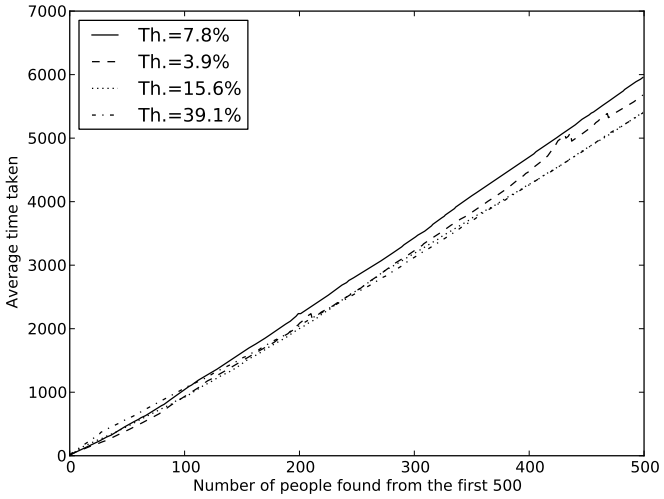


Figure D.11: The time taken to find 500 people, averaged over 20 base maps for four threshold values for the Low Pass Filter method, as modified to include Base Map multiplication

false peaks it is unable to escape — reducing the base value underneath the peak will not affect the base values of the cells causing the peak.

Across the four variable levels and across twenty maps, only one reached completion. Figure D.10 shows the resulting traces; averaged by time rather than by number as they were all incomplete, for all except the second map of Set B at the highest threshold of 39.1%, which was the only complete one.

As mentioned in Section 3.4.5.1, the Low Pass Filter method was modified such that the result from the Fourier transform was multiplied by the Base Value for every cell in the Navigation Layer. This gave a considerably improved completion rate, although none of the threshold levels managed to complete all 20 maps. The threshold which became stuck least often was at 39.1%. The results from this are given in Figure D.11. The lines are notably less smooth than those of other methods; the reason for this is that some of the results have been partly averaged from extrapolated data. The most affected is 3.9%; 4/19ths of this has been partially extrapolated. It should be noted, however, that all extrapolated results are from Set B.

Figure D.12 gives the box plots for the Low Pass Filter method, after multipli-

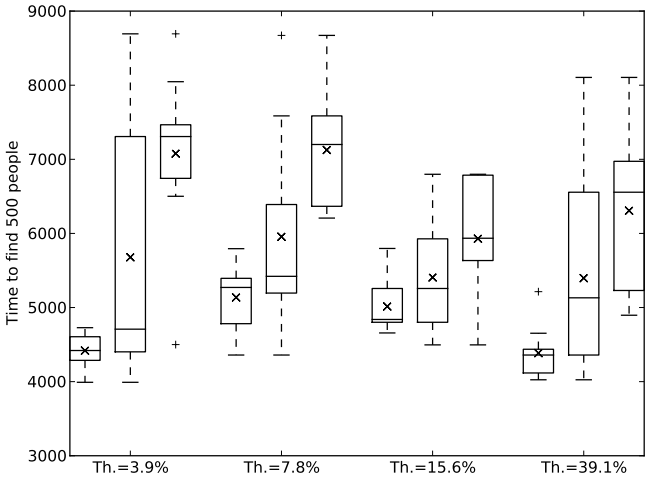


Figure D.12: The time taken to find 500 people using the Low Pass Filter method, as modified to include Base Map multiplication, averaged over 20 Base Maps for four threshold values. The three plots show results from Set A, all maps, and Set B respectively

cation with the Base Map. The maximum threshold would appear from both map sets to give the best result; this is clouded, however, by first in Set A the minimum threshold giving such a surprisingly good set of results — the trend for the other threshold levels would point towards greater thresholds giving better results — and secondly by its spread of times in Set B being somewhat greater than for the other threshold levels. This leads to the boxplots across all 20 maps being large and fairly similar for both maximum and minimum levels.

It should be noted that threshold values from all methods refer to the propagation of values. Cells will be updated only if one of their immediate neighbours have been changed since it was last updated; a neighbouring cell is only considered to have changed if the difference between its current value and its value at the previous iteration is greater than the threshold. A high threshold will therefore stop values from propagating to as great an extent as a low threshold.

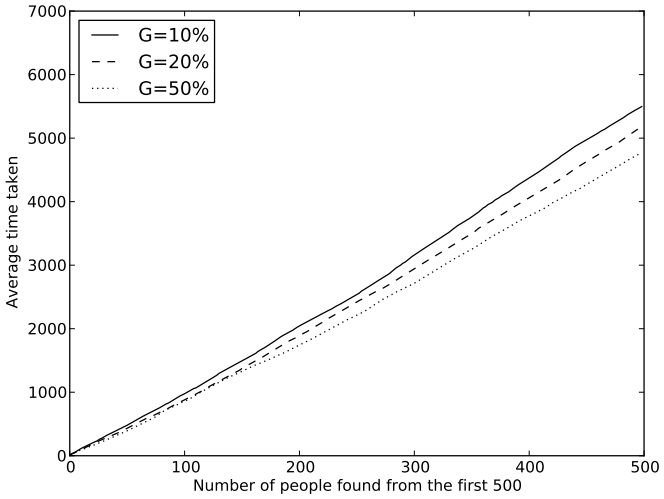


Figure D.13: The time taken to find 500 people, averaged over 20 base maps for four gradient levels (G from equation 3.6) for the Conservative method

D.6 Conservative

The results from the Conservative method — using equation 3.6 — can be seen in Fig. D.13. The results for this appear, while very consistent across both map sets, to be both counter intuitive and contradictory to the results obtained from other methods. The greater the gradient — i.e. the steeper the slope between two cells needs to be before any movement is permitted — the faster the response. This at first would seem counter intuitive, as low gradient values should permit cells with high Base Values to be propagated through the Navigation Layer for a much greater distance. In other methods, the variable values which permit a greater dissemination of data have tended to give a faster response. The reason why this has not had the same effect in this circumstance is hypothesised to be due to the conservative nature of this method — as a cell with a high base value has its value spread out the navigation value of that cell will itself fall, in contrast to other methods where it would retain its high value. This method warrants further investigation, however, to confirm this.

This method became stuck for all maps in Set A at a gradient of 5%, but for no others at any gradients.

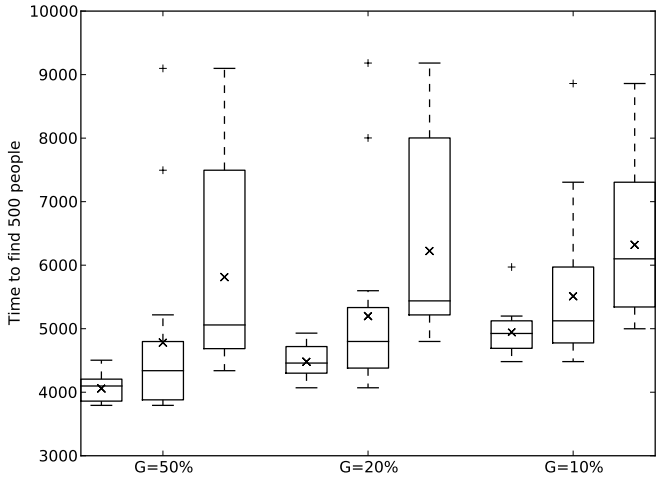


Figure D.14: The time taken to find 500 people using the Conservative method, averaged over 20 Base Maps for four gradient levels (G from equation 3.6). The three plots show results from Set A, all maps, and Set B respectively

Figure D.14 shows the boxplots for this method; for Set A and overall they confirm the trend described above. The results for Set B are rather less clear, as the range of results for each gradient is larger.

The equation for the Conservative method (from equation 3.6), at the best recorded diffusion level, is:

$$\left.\begin{aligned} l_t(i,j) &= l_{t-1}(i,j) - 1/16 \\ l_t(i^N,j^N) &= l_{t-1}(i^N,j^N) + 1/16 \end{aligned}\right\| \text{if } l_{t-1}(i,j) - l_{t-1}(i^N,j^N) > 0.5 \qquad \text{(D.7)}$$

D.7 Neighbour Difference

This algorithm, as described by Repenning,[53] has one variable to modify. The method does, however, require a threshold — a change below which cells will no longer propagate their new value to their neighbours. It has therefore been tested with two variables — a threshold and a diffusion level (D in equation 3.7). Three thresholds were tested, at 10%, 1% and 0.1%, and four diffusion levels, at 0.050, 0.075, 0.100 and 0.125. The range of D was only taken as far as 1/8, as above

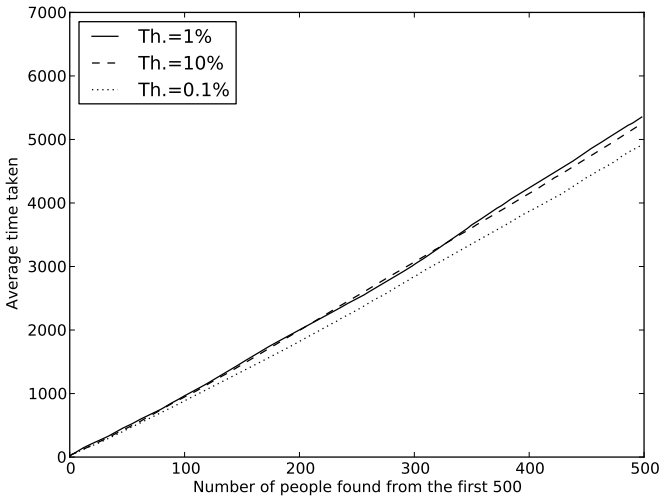


Figure D.15: The time taken to find 500 people, averaged over 20 base maps at four levels of diffusion for three threshold values, for the Neighbour Difference method

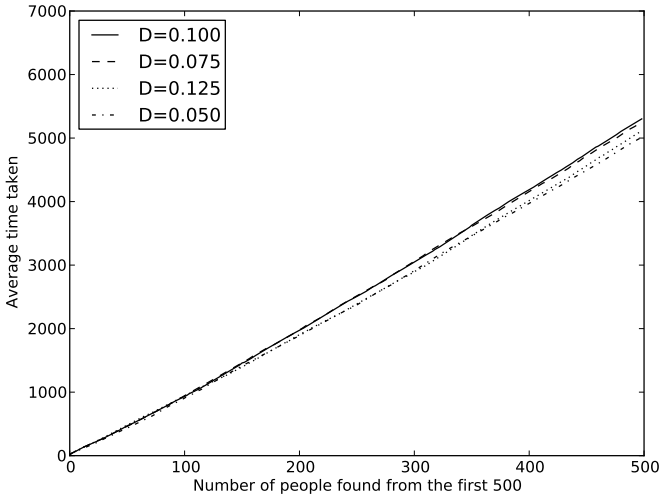


Figure D.16: The time taken to find 500 people, averaged over 20 base maps for four levels of diffusion (D from equation 3.7) at three threshold values, for the Neighbour Difference method

$\frac{1}{\text{number of neighbours}}$ the method proved unstable.

D is described not as a means of differentiating between distance over importance but as a method of setting the speed at which the data is propagated across the map. In this regard, the change in D should not have an effect on the speed of the algorithm, as for this simulation all data was fully propagated between iterations of robot movement. As the paper by Repanning is aimed towards creating artificial intelligences for computer games, having a variable for speed of propagation is more logical than for a search and rescue scenario where it is preferable for all data to be available as soon as is practicably possible.

The results from this method are shown in Figures D.15 and D.16, averaged across different thresholds and different diffusion levels, with the corresponding box-plots shown in Figures D.17 and D.18. As can be seen, there is little change in speed for any change in either variable. The reason for this could be because the diffusion level described by Repanning is not designed to make a difference to the timing (it being solely a speed of propagation variable).

The data for this method is of less use than it might be, and is in some respects less reliable than that from other methods, as it failed to complete many maps for a range of variable levels.

The seeming lack of correlation between the graphs produced when varying either variable (mirrored in each individual map set) would lead to the assumption that either the speed of this method is comparatively constant, or that any skew produced by the extrapolated results is sufficient to cover any noticeable trend. The variable levels selected for comparing this method with others were D=0.050 at a threshold of 0.1%, as this combination was marginally faster than others in both map sets.

Further experimentation could be performed for a greater range of threshold and diffusion values to ascertain whether a trend becomes apparent.

The equation for the Neighbour Difference method (from equation 3.8), at the best recorded diffusion level, is:

$$l_t(i,j) = l_{t-1}(i,j) + 0.05 \sum \left(l_{t-1}(i^N, j^N) - l_{t-1}(i,j) \right) \tag{D.8}$$

It should be noted that threshold values from all methods refer to the propagation

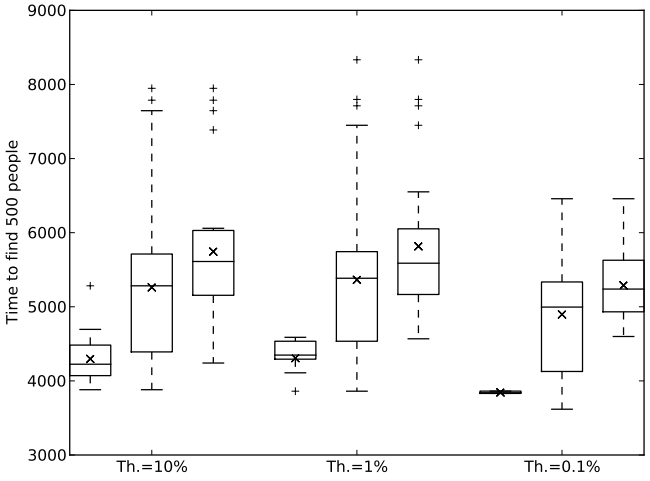


Figure D.17: The time taken to find 500 people using the Neighbour Difference method, averaged over 20 Base Maps at four levels of diffusion for three threshold values. The three plots show results from Set A, all maps, and Set B respectively

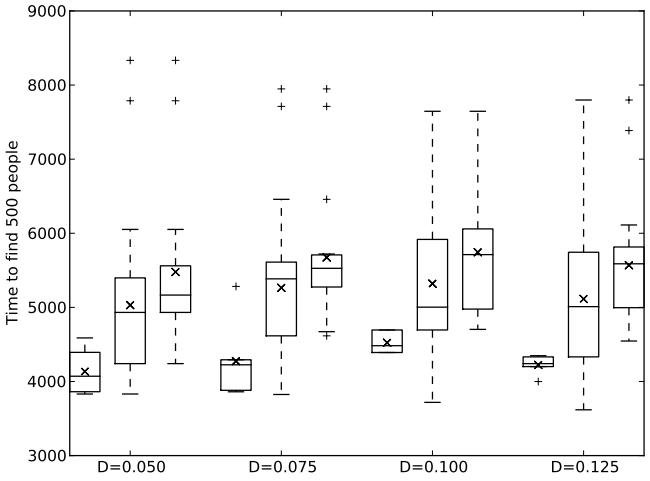


Figure D.18: The time taken to find 500 people using the Neighbour Difference method, averaged over 20 Base Maps for four levels of diffusion (D from equation 3.7) at three threshold values. The three plots show results from Set A, all maps, and Set B respectively

of values. Cells will be updated only if one of their immediate neighbours have been changed since it was last updated; a neighbouring cell is only considered to have changed if the difference between its current value and its value at the previous iteration is greater than the threshold. A high threshold will therefore stop values from propagating to as great an extent as a low threshold.

Appendix E

Game theory

One way of looking at collaboration is to take a game theory approach. Game theory is a way of looking at problems in terms of the 'payoff' given to the various 'players' based on their behaviour.

An example of this is the Prisoner's Dilemma. This is a two player game, where both players have been caught by the police for a serious crime. The police have enough evidence to convict both on minor charges, but not enough to convict them of the serious offense. Each is questioned individually. If both remain silent, they both receive minor prison sentences. If one informs on the other, the informer goes free and the other receives a long prison sentence. If both inform then they both receive a medium length sentence. This can be expressed as a payoff matrix (see Table E.1), where the two values are the respective payoffs for the two players.

Although the optimum decision for the two prisoners together would be to collaborate, this behaviour has a considerably greater risk associated with it and is not the best possible outcome for each player individually. If each of the options are examined separately, defection (informing on the other prisoner) will always be the safer option — if they also inform, the sentence is reduced from 1 year to 3 months. If they do not, the sentence is reduced from 1 month to nothing. However, as both players know this it is most likely that both will defect and receive the second worst option instead of collaborating and receiving the second best.

Many situations can be modelled as games, with varying numbers of players

	Collaborate	Defect
Collaborate	1 month, 1 month	1 year, Free
Defect	Free, 1 year	3 months, 3 months

Table E.1: Prisoner's Dilemma

	H	T
H	+1,-1	-1,+1
T	-1,+1	+1,-1

Table E.2: Penny Matching

and various payoff matrices. It is useful, then, to be able to work out an optimum strategy for each player based on the probable behaviour of the other players. This can be referred to as a Nash equilibrium — a situation where each player makes the best decision that they can, taking into account the planned decisions of all other players. To illustrate, take the case where each player makes a decision and then tells it to all other players. If no players wish to change their decisions based on this new knowledge of their opponents' plans, then the players are in a Nash equilibrium.

The decisions produced in a Nash equilibrium do not always have to be binary (a Pure strategy) — that is, they need not be a firm decision for a single particular choice which will be taken regardless of their opponents' decisions. Instead a Mixed strategy may be used, where each decision is given a set probability for how likely it is to be followed. This allows for some decisions to be considered more likely than others while allowing all options to remain possible.

Two examples of where this would be useful are the Penny Matching Game and the El Farol Bar problem. In the first, two players gamble by comparing coins, which they have secretly decided the orientation of (heads or tails). If both say the same, the first player gains both coins. If both say different things, the second player gains both coins (see Table E.2).

In the latter, a particular bar is seen as a good place to go if and only if it is not too crowded. A large group of people each independently make the decision whether or not to go to the bar. If more than 60% of them go, they will have a better time staying at home. If less than 60% of them go, they will have a better time at the

bar.

In both of these cases a pure strategy is not the best strategy — always picking heads, or always going to the bar, would not be a winning plan. There is no single set decision that, when presented with the decisions of the other players, would always be still seen as being the best decision. Instead a probability level is given to show how good each respective option is — in the case of the coin matching, this would be a 50% probability for either heads or tails.

Games need not be competitive (where the goals of all players are against each other); the players can also work together, either in a cooperative game (where there are opportunities for players to help each other into a win-win situation) or a collaborative game (where all players work together as a team, and the outcome for the team gives the winning conditions for the players).[248] Search and Rescue can be considered a collaborative game.

Many games — particularly competitive games — are zero sum; that is, one player's gain is the others' equivalent loss, and it is impossible to improve the result for one player without worsening the result for the others.[1] In cooperative games or collaborative games such as working together to find a missing person, however, the game usually ceases to be zero sum.[2] The players aren't just aiming to maximise their own personal profits, they're also working towards a common goal — in this case, finding a missing person.

Non-zero-sum games can be reduced to zero-sum games by adding an extra player to represent the global change[249] — a fictitious player who gains precisely what all other players lose overall, and vice versa. Other than this, the fictitious player has no effect on the game — he makes no moves of his own.

As mentioned above, there are many different types of game which real world problems can be described in terms of. Search and rescue could be seen as being similar to a Search game, also known as a Pursuit-Evasion game if the hider

[1]The penny matching game is a good example of this — whatever one player's gain is the other's automatic loss

[2]It should be noted that games being competitive and being zero sum are not mutually dependent — there are many games where all players are aiming to win by beating their opponents, but by improving their own positions some of their opponents' positions are also improved. Similarly there are collaborative strategies where some individuals in a team sacrifice themselves for the good of the group, despite taking an equivalent loss to the gain of others.

is moving.[250; 251] One good example of this is the Princess and the Monster game.[252] In this there are two characters — a Princess (who doesn't want to be found), and a Monster (who wants to find the princess). These are both capable of moving at a set speed, but importantly neither character knows the location of the other until or unless capture occurs (when both characters are on the same space).

Pursuit-Evasion games and Search games have one aspect which makes them unsuitable for direct comparison, however, in that they assume that the person being searched for is actively trying not to be found. In a search and rescue situation this is unlikely to be the case — while a small number of people will evade searchers, the majority will either be oblivious (not realising that they are being searched for — whether this is through mental problems or simply because they do not know they have been reported missing) or will be actively attempting to be found. In this case the game could be considered as a cooperative one, where the missing person is a player in the same way as the searchers.

This is known as a Rendezvous game,[253; 251] and it is far more comparable to the search and rescue problem. The basic idea is that there are two players who need to meet each other. Neither knows the other's location, and neither can detect the other until they both occupy the same space. They both need to decide a strategy for moving to minimise the time they take to find each other.

In order to fully describe the search and rescue problem varying probability levels also need to be taken into account — the potential payoff for searching high probability areas is going to be greater than that for low probability areas.

Taking the more concrete example of UAV path planning, each UAV could be given a payoff which is a function of the likelihood of finding the person (based on the probability of the areas covered). If a swarm of UAVs all worked collaboratively, they could aim to collectively maximise the team's payoff based on their individual planned moves. This might, for example, entail some UAVs not following 'optimum' routes if by doing so they increase the concentration of UAVs searching a small area at the expense of slightly less important areas.

Appendix F

GNSS[254]

The world's first Global Navigation Satellite System (GNSS) was conceived when researchers at MIT analysing signals coming from the Sputnik space probe noticed a Doppler shift that varied depending on the satellite's position; or, conversely, on the position of the receiver if the position of the satellite was known.

F.1 Satellite systems

The first GNSS — Transit — was developed by the US Navy. First designed in 1958, and fully operational only six years later by 1964, Transit was comprised of 4–7 satellites in low altitude polar orbits at 1100km. The system was designed for ships and submarines, and had an accuracy of around 25m. It was for the most part only usable for calculating 2d positions,[1] and only worked with stationary[2] receivers. The sparse constellation meant that there was only ever at most one satellite in view at a time, with waits of up to 100 minutes between satellites. The satellites transmitted a 150MHz and 400MHz signal which included their current location, and receivers used the Doppler shift from the signal to calculate their position on the satellite's ground path. Measurements involved recording the signal from the satellite for up to 20 minutes. Transit was decommissioned in 1996.

The Russian Tsikada system, developed soon after, worked along similar lines.

[1] 3d positions could be obtained from many satellite passes over the course of days

[2] Or very slow-moving, if the exact speed and bearing were known

In the US further systems were developed with marginal improvements, such as the slightly faster Timation system, or the Air Force's 621B system, which was capable of giving altitude measurements.

The current US GNSS — Navstar GPS — was approved by the Department of Defense in 1973, had its first satellites launched in 1978, and was declared fully operational in 1995. It was originally a military system, but was opened to civilians in 1993. Initially this was with 'Selective Availability' — a random error which was automatically put onto the civilian signal — and on the understanding that the US military could withdraw its availability at any time. This meant that the 'Standard Positioning Service' available to civilians was many times less accurate than the encoded 'Precise Positioning Service' available to the military. Selective Availability was turned off in 2000, and as the satellites in the constellation are updated the option for switching it back on is being removed from the system.

Other systems[3] exist, and work in similar ways, but are not covered in more detail here.

F.2 Segments

There are three segments which make up the GPS system: Space, Control, and User.

F.2.1 Space segment

This is a constellation of 24 or more satellites in 12 hour orbits. There are currently 31 satellites in six non-geosynchronous orbits (6 groups of 4 or more satellites per orbit), inclined at 55 degrees to the equator in an orbit with radius 26,560km. At least four satellites are in view at any point from anywhere on Earth,[4] with 6–8 being more common, although coverage at higher altitudes is notably worse than over the rest of the planet.

[3] Such as Galileo, GLONASS, and Beidou

[4] Assuming a clear view of the sky

F.2.2 Control segment

This is made up of Monitoring Stations and Ground Antennae around the world, and one Master Control Station at the Schreiver[5] Air Force Base near Colorado Springs. The monitoring stations receive signals from the satellites and pass them on to the master control station. This analyses telemetry from the satellites to ensure that they are functioning as expected, and uses the data to calculate the new ephemeris and clock corrections which are then reuploaded to the satellites via the ground antennae.

F.2.3 User segment

GPS receivers are commonly and commercially available worldwide, built into a wide range of devices. The receivers calculate their positions based on the signals received from the satellites.

F.3 Signals

The GPS satellites continuously transmit signals. There are two main carrier frequencies used for these (although others exist[6]): L1 at 1575.42MHz, and L2 at 1227.6MHz.[7] Using two frequencies means that some of the errors inherent in the system can be calculated by measuring the offset between the two signals. L1 is for civilian use, and L2 is for military use (although some civilian use is now permitted)

The satellites transmit two ranging codes, for calculating the time of flight from satellite to receiver: the Coarse/Acquisition (C/A) code (on L1), and the Precision (P) code (on both L1 and L2).

F.3.1 Coarse/Acquisition

This is a 1023 bit long pseudorandom number, modulated onto the carrier at 1.023MHz and repeating every 1ms. A form of phase modulation called binary phase shift key-

[5] formerly Falcon

[6] These are mainly for classified payloads such as the Nuclear Detonation Detection System, but are also used for civilian Safety of Life signals

[7] These are timed using the satellites' multiple atomic clocks, running at a base frequency of 10.23MHz

ing (BPSK) is used, where a 0 bit leaves the carrier unchanged and a 1 bit multiplies the carrier by -1. The sequences are Gold codes, which gives them good auto and cross correlation properties. They also have a low correlation between satellites and are highly orthogonal to each other, which means that multiple satellites can all transmit simultaneously on the same frequency without interfering with each other.

F.3.2 Precision

This works in a similar way to the C/A code, but uses a much longer pseudorandom number — 6.1871×10^{12} bits, broadcast at 10.23MHz and repeating weekly. As it's sent on both the L1 and L2 carrier frequencies it's transmitted 90 degrees out of phase to the C/A code. This code is normally encrypted (known as the Y code, or P(Y)), as it's designed for military use.

F.4 Ranging

The range is calculated by cross correlating a locally created code against the received code. This method would only work if the signals could be calculated identically and with identical start points — in other words, requiring perfectly synchronised clocks.

Distances are required from three known locations to an unknown location in order to calculate its position in 2D, using trilateration. The measurements in the GPS system give distances which are dependent on the clock difference between the receiver and the satellite. As all satellites are synchronised (they have (multiple) highly accurate atomic clocks, and any offsets are measured by the ground control stations and rebroadcast by the satellite for correction), the receiver will be offset the same amount from all the satellites. If four satellites are compared against then there are four unknowns and four equations, which removes the issue of clock synchronisation:

$$p^k = \sqrt{(x^k - x)^2 + (y^k - y)^2 + (z^k - z)^2} - b \tag{F.1}$$

For four or more satellites at positions (x^k, y^k, z^k), with pseudoranges p^k (the

distances between the receiver and each of the satellites), the four unknowns of receiver position (x, y, z) and receiver clock offset b can be calculated. A useful side effect of this is that the receiver can then know the time to a very high degree of accuracy.

F.5 Navigation message

The satellites transmit extra data, which is added over the C/A and P codes using modulo-2 addition. This is the Navigation Message, which contains several pieces of information about the satellite which it is being transmitted from as well as the constellation as a whole. The navigation message contains the health status of the satellite, its clock bias, ephemeris data (the satellite's position and velocity, for obtaining an accurate position for the receiver), and the almanac — a set of reduced precision ephemeris data for all satellites, allowing the receiver to make a rough estimate as to which satellites are available and where.

This information allows the receiver to calculate precisely the positions of all satellites that it is using for calculations and how they are moving (to within a few metres). Typically the ephemeris data has a far greater requirement for being up to date than the almanac, and will be updated in terms of hours as opposed to days.

The navigation message transmits at 50 bits per second, with a bit duration of 20ms, taking 12.5 minutes to send the entire message. The essential pieces of information (ephemeris and clock parameters) are repeated every 30s.

Appendix G

Projections

The surface of the Earth presents many difficulties to cartographers. It is possible to take a model of the Earth — say a sphere — and project onto that a grid such that every point could be given an exact location in terms of latitude, longitude and altitude. The Earth does not, however, have a regular shape — even when only taking into account the mean sea level. The surface of the planet is constantly changing through continental shift, meaning that while each plate is internally consistent they are permanently in a state of motion relative to each other. There are therefore many mathematical models or projections that are used to describe sections of the planet, usually picked for being internally accurate for the area required, and typically being consistent only over at most one tectonic plate. These are good for local measurements and have the advantage that a particular point at sea level will usually both keep the same latitude and longitude permanently and will have a zero reading for altitude from sea level.

These local projections cannot, however, be used to map the entire globe. The reasons for this are twofold — they would not retain a consistent datum, and they would not map globally to an accurate representation of the planet's surface. Projections therefore exist that are designed to encompass the entire globe; while being locally less accurate, they give a consistent error range across the planet and are immune to shifting plates. These projections are used by GNSS systems such as GPS.

Note that the grid system that is used is independent of the projection — Easting and Northing can map to Latitude and Longitude with a simple conversion, while elevation remains constant; changing between the different projections requires a full transformation.

The four systems that are relevant in this case are:

OSGB36

Ordinance Survey of Great Britain, 1936 revision — gives a locally accurate surface over the mainland UK, based on the 1830 Airy ellipsoid;

ED50

European Datum, 1950 — a Europe wide system based on the Hayford Ellipsoid, 1909 (aka International Ellipsoid, 1924);

ETRS89

European Terrestrial Reference System, 1989 — Another European system, locked to the European plate, but based on the GRS80 (Geodetic Reference System, 1980); and

WGS84

World Geodetic System, 1984 — a global system used by GPS based on the GRS80.

Another problem that is caused by the multiple systems is the discrepancies between the datum and the Terrestrial Reference Frame (TRF). The datum is the mathematical description of the ellipsoid, and as such has perfect accuracy as it is a definition. The TRF is the embodiment of this; a set of known points that are specified as being accurate with each other from which all other points derive their location. In the case of GPS, this is the constellation of satellites; in the case of OS, it is the triangulation points. These are marked with great accuracy, but are still liable to errors. These errors cause no internal problems as each entire system is internally coherent — each and every GPS reading is accurate to the satellite positions and so comparable with each other, but is not necessarily accurate to the original datum from which the system is derived. This means that there are likely to be errors in conversion between systems.

As both GPS and SRTM are based on the WGS84 ellipsoid, this should not cause an error in the data collected — only if compared to altitudes calculated by other means.

Bibliography

[1] Maritime and Coastguard Agency, "Search and Rescue Framework for the United Kingdom of Great Britain and Northern Ireland," Department for Transport, Tech. Rep. MCA/187, April 2008.

[2] D. Tripp, "Working with search and rescue helicopters," RAF, Tech. Rep., 2011.

[3] B. O. Koopman, *Search and Screening*. Pergamon, 1980.

[4] J. R. Frost, "Principles of search theory," 1999.

[5] *Land Search and Rescue Addendum to the National Search and Rescue Supplement to the International Aeronautical and Maritime Search and Rescue Manual*, 1st ed., National Search and Rescue Committee, November 2011.

[6] R. J. Koester, *Lost Person Behavior*. dbS Productions, 2008.

[7] D. C. Cooper, Ed., *Fundamentals of Search and Rescue*. Jones and Bartlett, 2005.

[8] *A Simple Guide to Conducting Ground Search and Rescue Detection Experiments*, U.S. Department of Homeland Security, United States Coast Guard, Office of Search and Rescue (G-RPR), May 2006.

[9] R. Q. Robe and J. R. Frost, "A method for determining effective sweep widths for land searches: Procedures for conducting detection experiments," The National Search and Rescue Committee, Tech. Rep. DTCG39-00-D-R00009 / DTCG32-01-F-000022, September 2002.

[10] C. Twardy, "A brief intro to search theory," 2012, accessed August 2012. [Online]. Available: http://sarbayes.org/search-theory/brief-intro-to-sar-theory-1-of-4/

[11] D. C. Cooper, J. R. Frost, and R. Q. Robe, "Compatibility of land sar procedures with search theory," U.S. Department of Homeland Security, United States Coast Guard, Operations (G-OPR), Tech. Rep. DTCG39-00-D-R00009 / DTCG32-02-F-000032, December 2003.

[12] R. Koester, D. C. Cooper, J. R. Frost, and R. Q. Robe, "Sweep width estimation for ground search and rescue," US Department of Homeland Security, United States Coast Guard, Operations (G-OPR), Tech. Rep., December 2004.

[13] *National Search and Rescue Manual*, Australian National Search and Rescue Council; Australian Maritime Safety Authority, July 2011.

[14] P. Niedfeldt, R. Beard, B. Morse, and S. Pledgie, "Integrated sensor guidance using probability of object identification," in *American Control Conference*, June 2010, pp. 788–793.

[15] D. C. Cooper, "The application of search theory to land search: Adjustment of probability of area," 2000.

[16] T. M. Kratzke, L. D. Stone, and J. R. Frost, "Search and rescue optimal planning system," in *2010 13th Conference on Information Fusion (FUSION)*, July 2010, pp. 1–8.

[17] M. Goodrich, B. Morse, D. Gerhardt, J. Cooper, M. Quigley, J. Adams, and C. Humphrey, "Supporting wilderness search and rescue using a camera-equipped mini UAV," *Journal of Field Robotics*, vol. 25, no. 1, pp. 89–110, 2008.

[18] D. Perkins, P. Roberts, and G. Feeney, "Missing person behaviour — an aid to the search manager," Mountain Rescue Council and Centre for Search Research, Tech. Rep., June 2003, 1st Edition.

[19] ——, "The U.K. missing person behaviour study," Mountain Rescue (England and Wales) and The Centre for Search Research, Tech. Rep., March 2011.

[20] D. O'Connor and J. R. Frost, "Controversial topics in inland SAR planning, with clarifications and rebuttals by J. R. Frost," Northeast Wilderness Search And Rescue (NEWSAR)," White paper, February 2004.

[21] G. Gibb and P. Woolnough, "Missing persons: Understanding, planning, responding," Grampian Police, Tech. Rep., April 2007.

[22] *Land Search Operations Manual*, 7th ed., Australian National Search and Rescue Council, November 2010.

[23] S. Waharte, N. Trigoni, and S. J. Julier, "Coordinated search with a swarm of UAVs," in *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON-09)*, June 2009, pp. 1–3.

[24] S. Waharte, A. Symington, and N. Trigoni, "Probabilistic search with agile UAVs," in *IEEE International Conference on Robotics and Automation (ICRA 2010)*, May 2010, pp. 2840–2845.

[25] Y. Yang, A. A. Minai, and M. M. Polycarpou, "Evidential map-building approaches for multi-UAV cooperative search," in *American Control Conference*, June 2005, pp. 116–121.

[26] C. Twardy and A. Oboler, *SORAL Reference Manual*, 2nd ed., May 2003.

[27] J. M. Bownds, M. J. Ebersole, D. Lovelock, D. J. O'Connor, and R. J. Toman, "Win CASIE III," accessed March 2012. [Online]. Available: http://www.wcasie.com/

[28] SAR Technology, "Incident commander pro," accessed October 2011. [Online]. Available: http://sartechnology.ca/sartechnology/ST_ProgramOverview.htm

[29] K. A. Yost and A. R. Washburn, "Solving large-scale allocation problems with partially observable outcomes," Naval Postgraduate School.

[30] Daniel H. Wagner Associates, "Computer assisted search planning (CASP) 2.0," 2005, accessed November 2010. [Online]. Available: http://www.wagner.com/technologies/missionplanning/search-optimization/casp2.html

[31] M. D. McDaniel, "Agent-based model of lost person wayfinding," Master's thesis, University of California, Santa Barbara, March 2010.

[32] N. Jones and C. Twardy, "Mapscore lost person model rating system," 2012, accessed August 2012. [Online]. Available: http://mapscore.sarbayes.org/

[33] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, no. 1, pp. 269–271, 1959.

[34] S. Cameron, S. Hailes, S. Julier, S. McClean, G. Parr, N. Trigoni, M. Ahmed, G. McPhillips, R. de Nardi, J. Nie, A. Symington, L. Teacy, and S. Waharte, "SUAAVE: Combining aerial robots and wireless networking," in *25th Bristol International UAV Systems Conference*, 2010, pp. 1–14.

[35] W. T. L. Teacy, J. Nie, S. McClean, G. Parr, S. Hailes, S. Julier, N. Trigoni, and S. Cameron, "Collaborative sensing by unmanned aerial vehicles," in *3rd International Workshop on Agent Technology for Sensor Networks (ATSN-09)*, May 2009, pp. 13–16.

[36] J. Tisdale, Z. Kim, and J. K. Hedrick, "Autonomous UAV path planning and estimation," *IEEE Robotics and Automation Magazine*, vol. 16, no. 2, pp. 35–42, June 2009.

[37] F. Bourgault, T. Furukawa, and H. Durrant-Whyte, "Coordinated decentralized search for a lost target in a bayesian world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2003, pp. 48–53.

[38] T. Chung, M. Kress, and J. O. Royset, "Probabilistic search optimization and mission assignment for heterogeneous autonomous agents," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2009, pp. 939–945.

[39] H. Choset, "Coverage for robotics — a survey of recent results," *Annals of Mathematics and Articial Intelligence*, no. 31, pp. 113–126, 2001.

[40] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics*, vol. 34, no. 1, pp. 718–725, February 2004.

[41] R. Dasgupta and K. Cheng, "Distributed coverage of unknown environments using multi-robot swarms with memory and communication constraints," Department of Computer Science, University of Nebraska at Omaha, Tech. Rep. cst-2009-1, May 2009.

[42] S. Hert, S. Tiwari, and V. Lumelsky, "A terrain covering algorithm for an AUV," *Autonomous Robots*, vol. 3, no. 2–3, pp. 91–119, June 1996.

[43] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 918–933, October 1999.

[44] A. Zelinsky, R. Jarvis, J. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *International Conference on Advanced Robotics*, 1993, pp. 533–538.

[45] Y. Altshuler, V. Yanovsky, I. A. Wagner, and A. M. Bruckstein, "The cooperative hunters — efficient cooperative search for smart targets using UAV swarms," in *Second International Conference on Informatics in Control, Automation and Robotics (ICINCO), the First International Workshop on Multi-Agent Robotic Systems (MARS), Barcelona, Spain*, 2005, pp. 165–170.

[46] S. Kim, K. Kim, and T.-H. Kim, "Human aided cleaning algorithm for low-cost robot architecture," in *Human-Computer Interaction — HCI Intelligent Multimodal Interation Environments*, J. A. Jacko, Ed., no. 3. 12th International Conference, HCI International, 2007, pp. 366–37.

[47] Y. Jin, Y. Liao, A. A. Minai, and M. M. Polycarpou, "Balancing search and target response in cooperative unmanned aerial vehicle (UAV) teams,"

IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics, vol. 36, no. 3, pp. 571–587, June 2006.

[48] M. M. Polycarpou, Y. Yang, and K. M. Passino, "A cooperative search framework for distributed agents," in *IEEE International Symposium on Intelligent Control*, 2001, pp. 1–6.

[49] M. Flint, E. Fernandez-Gaucherand, and M. Polycarpou, "Cooperative control for uavs searching risky environments for targets," in *42nd IEEE Conference on Decision and Control*, December 2003, pp. 3567–3572.

[50] M. Flint, M. Polycarpou, and E. Fernandez-Gaucherand, "Cooperative control of multiple autonomous uavs searching for targets," in *41st IEEE Conference on Decision and Control*, December 2002, pp. 2823–2828.

[51] F. McPherson and R. Holden, Eds., *Oxford English Dictionary*, online ed. Oxford University Press, 2012.

[52] S. Waharte and N. Trigoni, "Supporting search and rescue operations with UAVs," in *Robots and Security (ROBOSEC)*, September 2010, pp. 142–147.

[53] A. Repenning, "Collaborative diffusion: programming antiobjects," in *OOPSLA '06: Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*. New York, NY, USA: ACM, 2006, pp. 574–585.

[54] C. Luo, S. X. Yang, and M. Meng, "Real-time map building and area coverage in unknown environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 1736–1741.

[55] Y. Guo and M. Balakrishnan, "Complete coverage control for nonholonomic mobile robots in dynamic environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 1704–1709.

[56] D. W. Gage, "Randomized search strategies with imperfect sensors," in *SPIE Mobile Robots VIII*, September 1993, pp. 270–279.

[57] L. Petricca, P. Ohlckers, and C. Grinde, "Micro- and nano-air vehicles: State of the art," *International Journal of Aerospace Engineering*, p. 214549, February 2011.

[58] "Uavforge," 2012, a DARPA / SSC Atlantic collaborative initiative to design, build and manufacture advanced small UAV systems. [Online]. Available: http://www.uavforge.net/

[59] G. de Croon, M. Groen, C. de Wagter, B. Remes, R. Ruijsink, and B. van Oudheusden, "Design, aerodynamics, and autonomy of the delfly," *Bioinspiration and Biomimetics*, vol. 7, no. 2, p. 025003, June 2012.

[60] N. Tesla, "Tesla's new device like Bolts of Thor," *New York Times*, p. 8(3), 8th December 1915.

[61] M. E. Dempsey, "Eyes of the army — u.s. army roadmap for unmanned aircraft systems 2010-2035," U.S. Army UAS Center of Excellence, Tech. Rep., 2010.

[62] D. Schneider, "Drone aircraft: How the drones got their stingers," *IEEE Spectrum*, pp. 47–51, January 2011.

[63] A. Ollero, J. R. M. de Dios, and L. Merino, "Unmanned aerial vehicles as tools for forest-fire fighting," in *International Conference on Forest Fire Research*, 2006, pp. 1–14.

[64] L. Merino, F. Caballero, J. R. M. de Dios, and A. Ollero, "Cooperative fire detection using unmanned aerial vehicles," in *Robotics and Automation*, 2005, pp. 1884–1889.

[65] A. Budiyono, W. Adiprawita, and B. Riyanto, "Modelling and control of unmanned flying robots," in *Electrical Engineering and Informatics (ICEEI)*, July 2011, pp. 1–7.

[66] S. R. Herwitz, S. Dunagan, D. Sullivan, R. Higgins, L. Johnson, Z. Jian, R. Slye, J. Brass, J. Leung, B. Gallmeyer, and M. Aoyagi, "Solar-powered UAV mission for agricultural decision support," in *Geoscience and Remote Sensing*, 2003, pp. 1692–1694.

[67] S. Varakliotis, S. Hailes, R. D. Nardi, and M. Ahmed, "Maximising resources: UAV and cognitive radio technologies in the emergency services arena," *Journal of the British Association of Public Safety Communications Officials*, vol. 16, no. 6, pp. 28–29, November 2010.

[68] K. Kaaniche, B. Champion, C. Pegard, and P. Vasseur, "A vision algorithm for dynamic detection of moving vehicles with a UAV," in *Robotics and Automation*, 2005, pp. 1878–1883.

[69] US Department of the Air Force; Air Force Materiel Command, "Airborne sense and avoid (ABSAA)," July 2012, solicitation Number: FA8620-12-X-0001.

[70] C. A. Mendenhall, "Integrating the unmanned aircraft system into the national airspace system," Master's thesis, National Defense University, Joint Forces Staff College, Joint Advanced Warfighting School, June 2011.

[71] Civil Aviation Authority, "Unmanned aircraft operations in the West Wales area — introduction of new danger areas EG D202, EG D202A, EG D202B and EG D202C and a further sub-division of EG D201 to form EG D201E," National Air Traffic Services Ltd, Tech. Rep. AIC: Y 052/2011, June 2011.

[72] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *International Symposium on Experimental Robotics*, December 2010, pp. 361–373.

[73] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *International Conference on Robotics and Automation*, May 2011, pp. 2520–2525.

[74] P. Ferrat, S. Neukom, C. Gimkiewicz, and T. Baechler, "Ultra-miniature omnidirectional camera for an autonomous flying micro-robot," in *SPIE Photonics Europe*, 2008, pp. 7000–22.

[75] J. Roberts, T. Stirling, J. Zufferey, and D. Floreano, "3-D relative positioning sensor for indoor flying robots," in *Autonomous Robots*, 2012, pp. 5–20.

[76] X. Yang, L. Mejias, and T. S. Bruggemann, "A spatial collision avoidance strategy for uavs in a non-cooperative environment," in *International Conference on Unmanned Aircraft Systems (ICUAS12)*, 2012.

[77] A. A. Paranjape, J. Kim, and S.-J. Chung, "Closed-loop perching of aerial robots with articulated flapping wings," *IEEE Transactions on Robotics*, 2012, (Under review).

[78] J. W. Roberts, R. Cory, and R. Tedrake, "On the controllability of fixed-wing perching," in *American Control Conference (ACC)*, 2009, pp. 2018–2023.

[79] A. L. Desbiens, A. T. Asbeck, and M. R. Cutkosky, "Landing, perching and taking off from vertical surfaces," *International Journal of Robotics Research*, vol. 30, no. 3, pp. 355–370, March 2011.

[80] M. Kovac, J. Germann, C. Hurzeler, R. Siegwart, and D. Floreano, "A perching mechanism for micro aerial vehicles," *Micro-Nano Mechatronics*, vol. 5, pp. 77–91, 2010.

[81] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *Robotics and Automation*, 2011, pp. 763–770.

[82] I. Palunko, R. Fierro, and P. Cruz, "Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload: A dynamic programming approach," in *Robotics and Automation*, 2012, pp. 2691–2697.

[83] M. Muller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *Intelligent Robots and Systems*, 2011, pp. 5113–5120.

[84] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[85] A. Schoellig and R. D'Andrea, "Optimization-based iterative learning control for trajectory tracking," in *European Control Conference*, 2009, pp. 1505–1510.

[86] A. Schoellig, J. Alonso-Mora, and R. D'Andrea, "Independent vs. joint estimation in multi-agent iterative learning control," in *IEEE Decision and Control*, 2010, pp. 6949–6954.

[87] D. H. Shim, H. J. Kirn, and S. Sastry, "Hierarchical control system synthesis for rotorcraft-based unmanned aerial vehicles," in *AIAA Guidance, Navigation, and Control*, August 2000, p. 4057.

[88] F. Delle Fave, A. Rogers, Z. Xu, S. Sukkarieh, and N. R. Jennings, "Deploying the max-sum algorithm for coordination and task allocation of unmanned aerial vehicles for live aerial imagery collection," in *Robotics and Automation*, 2012, pp. 469–476.

[89] P. Doherty, J. Kvarnstrom, F. Heintz, D. Landen, and P.-M. Olsson, "Research with collaborative unmanned aircraft systems," in *Cognitive Robotics*, ser. Dagstuhl Seminar Proceedings, no. 10081, 2010, pp. 1862–4405.

[90] J. Kvarnstrom and P. Doherty, "Automated planning for collaborative UAV systems," in *11th IEEE International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2010, pp. 1078–1085.

[91] M. Valenti, B. Bethke, G. Fiore, J. How, and E. Feron, "Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery," in *AIAA Conference on Guidance, Navigation and Control*, August 2006, p. 6200.

[92] A. Ryan, M. Zennaro, A. Howell, R. Sengupta, and J. Hedrick, "An overview of emerging results in cooperative UAV control," in *43rd IEEE Conference on Decision and Control*, vol. 1, December 2004, pp. 602–607.

[93] G. Homann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. Tomlin, "The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC)," in *23rd Digital Avionics Systems Conference*, November 2004, pp. 12–E–4.

[94] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple

micro UAV testbed," *IEEE Robotics and Automation Magazine*, pp. 56–65, September 2010.

[95] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *International Conference on Robotics and Automation*, May 2012, pp. 477–483.

[96] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, "Cooperative grasping and transport using multiple quadrotors," in *Distributed Autonomous Robotic Systems*, November 2010, pp. 545–558.

[97] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction of cubic structures with quadrotor teams," in *Robotics: Science and Systems*, June 2011, pp. 117–185.

[98] S. Lupashin, A. Schllig, M. Sherback, and R. DAndrea, "A simple learning strategy for high-speed quadrocopter mult-flips," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 1642–1648.

[99] O. Holland, J. Woods, R. D. Nardi, and A. Clark, "Beyond swarm intelligence: The ultraswarm," in *IEEE Swarm Intelligence*, June 2005, pp. 217–224.

[100] R. D. Nardi and O. E. Holland, "Ultraswarm: A further step towards a flock of miniature helicopters," in *Swarm Robotics*, 2006, pp. 116–128.

[101] R. D. Nardi, O. Holland, J. Woods, and A. Clark, "SwarMAV: A swarm of miniature aerial vehicles," in *21st Bristol International UAV Systems Conference*, 2006.

[102] D. Shim, H. Chung, H. J. Kim, and S. Sastry, "Autonomous exploration in unknown urban environments for unmanned aerial vehicles," in *AIAA Guidance, Navigation, and Control*, August 2005.

[103] M. Colton, L. Sun, D. Carlson, and R. Beard, "Multi-vehicle dynamics and control for aerial recovery of micro air vehicles," *Vehicle Autonomous Systems*, vol. 9, pp. 78–107, 2011.

[104] R. Leishman, J. Macdonald, T. McLain, and R. Beard, "Relative navigation and control of a hexacopter," in *IEEE Robotics and Automation*, 2012, pp. 4937–4942.

[105] J. How, "Multi-vehicle flight experiments: Recent results and future directions," in *NATO Conference*, May 2007.

[106] M. Valenti, B. Bethke, D. Dale, A. Frank, J. McGrew, S. Ahrens, J. P. How, and J. Vian, "The MIT indoor multi-vehicle flight testbed," in *IEEE International Conference on Robotics and Automation*, April 2007, pp. 2758–2759.

[107] A. Richards, J. Bellingham, M. Tillerson, and J. How, "Coordination and control of multiple UAVs," in *AIAA Guidance, Navigation and Control*, August 2002, p. 4588.

[108] T. Stirling, J. Roberts, J.-C. Zufferey, and D. Floreano, "Indoor navigation with a swarm of flying robots," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 4641–4647.

[109] T. Stirling, S. Wischmann, and D. Floreano, "Energy-efcient indoor search by swarms of simulated ying robots without global information," *Swarm Intelligence*, vol. 4, no. 2, pp. 117–143, February 2010.

[110] F. D. Fave, A. Farinelli, A. Rogers, and N. R. Jennings, "A methodology for deploying the max-sum algorithm and a case study on unmanned aerial vehicles," in *Innovative Applications of Artificial Intelligence*, 2012, pp. 2275–2280.

[111] A. Symington, S. Waharte, S. J. Julier, and N. Trigoni, "Probabilistic target detection by camera-equipped UAVs," in *IEEE International Conference on Robotics and Automation (ICRA 2010)*, May 2010, pp. 4076–4081.

[112] L. Mejias, P. Campoy, I. F. Mondragon, and P. Doherty, "Stereo visual system for autonomous air vehicle navigation," in *6th IFAC Symposium on Intelligent Autonomous Vehicles*, vol. 6, no. 1, 2007, pp. 203–208.

[113] P. Rudol and P. Doherty, "Human body detection and geolocalization for UAV search and rescue missions using color and thermal imagery," in *IEEE Aerospace Conference*, 2008, pp. 1–8.

[114] T. E. Noll, S. D. Ishmael, B. Henwood, M. E. Perez-Davis, G. C. Tiffany, J. Madura, M. Gaier, J. M. Brown, and T. Wierzbanowski, "Technical findings, lessons learned, and recommendations resulting from the Helios Prototype vehicle mishap," in *NATO/RTO AVT-145 Workshop on Design Concepts, Processes and Criteria for UAV Structural Integrity*, May 2007, p. 3.4.

[115] B. Chapman, S. Hensley, and Y. Lou, "The JPL UAVSAR," *Alaska Satellite Facility — News and Notes*, vol. 7, no. 1, 2011.

[116] Civil UAV Assessment Team, "Earth observations and the role of UAVs: A capabilities assessment," NASA, Tech. Rep. v.1.1, August 2006.

[117] "QinetiQ files for three world records for its zephyr solar powered UAV," accessed August 2012. [Online]. Available: http://www.qinetiq.com/news/pressreleases/Pages/three-world-records.aspx

[118] G. Tournier, M. Valenti, J. How, and E. Feron, "Estimation and control of a quadrotor vehicle using monocular vision and moire patterns," in *AIAA Conference on Guidance, Navigation and Control*, August 2006, p. 6711.

[119] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *Robotics and Automation*, May 2009, pp. 2878–2883.

[120] B. Steder, G. Giorgio, C. Stachniss, S. Grzonka, A. Rottmann, and W. Burgard, "Learning maps in 3d using attitude and noisy vision sensors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 644–649.

[121] A. Xu and G. Dudek, "A vision-based boundary following framework for aerial vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '10)*, October 2010, pp. 81–86.

[122] R. He, A. Bachrach, M. Achtelik, A. Geramifard, D. Gurdan, S. Prentice, J. Stumpf, and N. Roy, "On the design and use of a micro air vehicle to track and avoid adversaries," *The International Journal of Robotics Research*, no. 29, pp. 529–546, 2010.

[123] M. Andriluka, P. Schnitzspan, J. Meyer, S. Kohlbrecher, K. Petersen, O. von Stryk, S. Roth, and B. Schiele, "Vision based victim detection from unmanned aerial vehicles," in *Intelligent Robots and Systems (IROS)*, 2010, pp. 1740–1747.

[124] B. Morse, C. Engh, and M. Goodrich, "UAV video coverage quality maps and prioritized indexing for wilderness search and rescue," in *Human Robot Interaction*, 2010, pp. 227–234.

[125] I. R. Nourbakhsh, K. Sycara, M. Koes, M. Yong, M. Lewis, and S. Burion, "Human-robot teaming for search and rescue," *IEEE Pervasive Computing*, vol. 4, no. 1, pp. 72–79, March 2005.

[126] W. T. L. Teacy, G. Chalkiadakis, A. Farinelli, A. Rogers, N. R. Jennings, S. McClean, and G. Parr, "Decentralized bayesian reinforcement learning for online agent collaboration," in *Autonomous Agents and Multiagent Systems*, June 2012, pp. 417–424.

[127] S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar, "Optimized stochastic policies for task allocation in swarms of robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, August 2009.

[128] T. Schouwenaars, A. Stubbs, J. Paduano, and E. Feron, "Multi-vehicle path planning for non-line of sight communication," in *American Control Conference*, 2006, pp. 5757–5762.

[129] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," *IEEE Proceedings (special issue on Multi-Robot Systems)*, vol. 94, no. 7, pp. 1384–1397, July 2006.

[130] G. Hattenberger, S. Lacroix, and R. Alami, "Formation flight: evaluation of autonomous configuration control algorithms," in *IEEE Intelligent Robots and Systems (IROS 2007)*, October 2007, pp. 2628–2633.

[131] A. Calise and D. Preston, "Swarming/flocking and collision avoidance for mass airdrop of autonomous guided parafoils," in *AIAA Guidance, Navigation, and Control*, 2005, p. 6477.

[132] D. J. Nowak, I. Price, and G. B. Lamont, "Self organized UAV swarm planning optimization for search and destroy using swarmfare simulation," in *Winter Simulation Conference*, 2007, pp. 1315–1323.

[133] C. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," in *Conference on Computer Graphics (SIGGRAPH)*, vol. 21, no. 4, 1987, pp. 25–34.

[134] *Agent Technology for Disaster Management (ATDM), First International Workshop on*, May 2006.

[135] A. Robinson, "FAA authorizes Predators to seek survivors," August 2006, accessed June 2010. [Online]. Available: http://www.af.mil/news/story.asp?storyID=123024467

[136] E. Ackerman, "Japan earthquake: Global hawk UAV may be able to peek inside damaged reactors," *IEEE Spectrum — Automaton*, March 2011.

[137] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Distributed cooperative search using information-theoretic costs for particle filters with quadrotor applications," in *AIAA Guidance, Navigation, and Control*, August 2006, p. 6576.

[138] "Robocup rescue," accessed August 2012. [Online]. Available: http://www.robocuprescue.org/

[139] "Rules discussion notes (2010)," December 2009, international Rescue Robot Workshop.

[140] *Search and Rescue Challenge Mission, rules and judging criteria*, 1st ed., UAV Challenge: Outback Rescue, May 2012.

[141] A. Posch and S. Sukkarieh, "UAV based search for a radio tagged animal using particle filters," in *Australasian Conference on Robotics and Automation (ACRA)*, December 2009.

[142] P. J. Duffett-Smith and G. Woan, "The CURSOR radio navigation and tracking system," *Journal of Navigation*, vol. 45, no. 2, pp. 157–165, May 1992.

[143] L. A. Merry, R. M. Faragher, and S. Scheding, "Comparison of opportunistic signals for localisation," in *7th IFAC Symposium on Intelligent Autonomous Vehicles*, G. Indiveri and A. M. Pascoal, Eds., vol. 7, no. 1, 2010, pp. 109–114.

[144] T. Hancock, "GPS tracking device," Master's thesis, School of Engineering and Computing Sciences, University of Durham, May 2010.

[145] *LM75 Digital Temperature Sensor and Thermal Watchdog with Two-Wire Interface*, National Semiconductor, June 1996.

[146] J. Watkins, "Remote GPS tracking for the general market," Master's thesis, Durham University School of Engineering, 2008.

[147] *GM862 Family Hardware User Guide*, 1st ed., Telit, September 2011.

[148] *MSP430F15x, MSP430F16x, MSP430F161x Mixed Signal Microcontroller*, Slas368g ed., Texas Instruments, October 2002, revised March 2011.

[149] Y. Ming, "GPS / GPRS tracking system," Master's thesis, Durham University School of Engineering, June 2007.

[150] R. Richardson, "A GPS based solution for traffic monitoring and congestion reduction," Master's thesis, Durham University School of Engineering, 2007.

[151] G. Wen, "GPS tracker with temperature sensor," Master's thesis, Durham University School of Engineering, June 2007.

[152] Y. Zhou, "Measurement of GPS tracker's sensitivity and development," Master's thesis, Durham University School of Engineering, June 2008.

[153] E. King, "The integration of environmental audio noise sampling onto GPS/GPRS trackers for noise mapping," Master's thesis, Durham University School of Engineering, 2009.

[154] Python, *Python Documentation*, v2.7.3 ed., The Python Standard Library, section 11.5. marshal — Internal Python object serialization.

[155] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616 (Draft Standard), Internet Engineering Task Force, June 1999, updated by RFCs 2817, 5785, 6266, 6585. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt

[156] *TR-102 Personal Tracker User Manual*, v.1.7 ed., GlobalSat, October 2008.

[157] *SmartONE User Manual*, 1st ed., Globalstar.

[158] C. Cheung, *MeiTrack MVT 600 User Guide*, v.2.6 ed., Shenzhen Meiligao Electronics Co., Ltd, July 2012.

[159] ——, *MeiTrack VT400 User Guide*, v.2.5 ed., Shenzhen Meiligao Electronics Co., Ltd, August 2012.

[160] *CelloTrack Solar — Overview*, Version 1.0 ed., Pointer Telocation, January 2010.

[161] *ATrack Wifi Offline Tracking Device AY5i Hardware Specification*, ATrack Technology Inc.

[162] *TinyTelemetry Datasheet*, ImmersionRC.

[163] *Instruction Manual for EagleEyes FPV Station*, Document version 1.19 (corresponds to software version 10.39 or later) ed., Eagle Tree Systems, 2011.

[164] B. Garrabrant, *TinyTrak3Plus Owner's Manual*, version 1.03 ed., Byonics, 2005.

[165] A. Krop-Benesch, *GPS Plus Collar User's Manual*, version: 1.6.3 ed., Vectronic Aerospace, March 2012.

[166] GPSGate AB, "GPS Gate Server," accessed August 2012. [Online]. Available: http://gpsgate.com/products/gpsgate_server

[167] Globalsat, "Ezfinder," accessed August 2012. [Online]. Available: http://ezfinder.com.tw/

[168] TraqLogic, "GPS Tracking Platform: WebtraQ," accessed August 2012. [Online]. Available: http://www.traqlogic.com/software.html

[169] MeiTrack, "Meitrack — every step in tracking," accessed August 2012. [Online]. Available: http://www.trackingmate.com/

[170] *GM862-GPS-OT User Guide*, 1st ed., Nosh Merchant Ltd OpenGPS, March 2010.

[171] M. S. Grewal, L. R. Weill, and A. P. Andrews, *Global Positioning Systems, Inertial Navigation and Integration.* John Wiley and Sons, 2001.

[172] *Training Manual for GPS Operation*, ND Department of Transportation, Surveys and Photogrammetry, March 2008.

[173] NAVSTAR, "Global positioning system standard positioning service performance standard," United States of America Department of Defense, Tech. Rep. 4th Edition, September 2008.

[174] William J. Hughes Technical Center, NSTB/WAAS T&E Team, "Global positioning system (GPS) standard positioning service (SPS) performance analysis report," Federal Aviation Administration (FAA) GPS Product Team, Tech. Rep. 73, April 2011, reporting Period: 1 January – 31 March 2011.

[175] "Openstreetmap," http://www.openstreetmap.org/, accessed 2011.

[176] "Openstreetmap stats report," http://www.openstreetmap.org/stats/data_stats.html, accessed 2011.

[177] T. G. Farr, P. A. Rosen, E. Caro, R. Crippen, R. Duren, S. Hensley, M. Kobrick, M. Paller, E. Rodriguez, L. Roth, D. Seal, S. Shaffer, J. Shimada,

J. Umland, M. Werner, M. Oskin, D. Burbank, and D. Alsdorf, "The shuttle radar topography mission," *Reviews of Geophysics*, vol. 45, p. 2005RG000183, 2007.

[178] NASA, "Shuttle radar topography mission — final precision data product descriptions," http://www2.jpl.nasa.gov/srtm/datafinaldescriptions.html, 2001, accessed 2011.

[179] Y. Gorokhovich and A. Voustianiouk, "Accuracy assessment of the processed SRTM-based elevation data by CGIAR using field data from USA and Thailand and its relation to the terrain characteristics," *Remote sensing of environment*, vol. 104, no. 4, pp. 409–415, 2006.

[180] G. Sun, K. J. Ranson, V. I. Kharuk, and K. Kovacs, "Validation of surface height from shuttle radar topography mission using shuttle laser altimeter," *Remote Sensing of Environment*, no. 88, pp. 401–411, 2003.

[181] H. Reuter, A. Nelson, and A. Jarvis, "An evaluation of void filling interpolation methods for SRTM data," *International Journal of Geographic Information Science*, vol. 21, no. 9, pp. 983–1008, 2007.

[182] J. Gonçalves and J. C. Fernandes, "Assessment of SRTM-3 DEM in Portugal with topographic map data," in *Proceedings of the EARSeL Workshop 3D-Remote Sensing*, Porto, Portugal (European Association of Remote Sensing Laboratories), June 2005.

[183] P. Elsner and M. Bonnici, "Vertical accuracy of shuttle radar topography mission (SRTM) elevation and void-filled data in the Libyan Desert," *International Journal of Ecology & Development*, vol. 8, no. 7, pp. 66–80, 2007.

[184] A. Jarvis, J. Rubiano, A. Nelson, A. Farrow, and M. Mulligan, *Practical use of SRTM data in the tropics – Comparisons with digital elevation models generated from cartographic data*, working document no. 198 ed., Centro Internacional de Agricultura Tropical (CIAT), 2004.

[185] A. Helm, A. Braun, S. Eickschen, and T. Schune, "Calibration of the shuttle radar topography mission X-SAR instrument using a synthetic altimetry data model," *Canadian Journal of Remote Sensing*, vol. 28, no. 4, pp. 573–580, 2002.

[186] B. Smith and D. Sandwell, "Accuracy and resolution of shuttle radar topography mission data," *Geophysical Research Letters*, vol. 30, no. 9, p. 20, 2003.

[187] G. C. Miliaresis and C. V. E. Paraschou, "Vertical accuracy of the SRTM DTED level 1 of Crete," *International Journal of Applied Earth Observation and Geoinformation*, vol. 7, no. 1, pp. 49–59, 2005.

[188] G. Smith, "Fife weather station raw data files," 2006–2011, accessed 2011. [Online]. Available: http://www.fifeweather.co.uk/

[189] S.-S. Jan, D. Gebre-Egziabher, T. Walter, and P. Enge, "Improving GPS-based landing system performance using an empirical barometric altimeter confidence bound," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 44, no. 1, pp. 127–146, january 2008.

[190] H. Zhenhai and H. Shengguo, "Data fusion approach based on high precision barometric altimeter and GPS," in *Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on*, May 2009, pp. 1–4.

[191] J. Kim and S. Sukkarieh, "A baro-altimeter augmented INS/GPS navigation system for an uninhabited aerial vehicle," in *The 6th International Symposium on Satellite Navigation Technology Including Mobile Positioning & Location Services*, 2003, p. 19.

[192] I.-H. Whang and W.-S. Ra, "Simple altitude estimator using air-data and GPS measurements," in *Proceedings of the 17th World Congress*. The International Federation of Automatic Control, July 2008, pp. 4060–4065.

[193] M. Castillo-Effen, "Cooperative localization in wireless networked systems," Ph.D. dissertation, University of South Florida, 2007.

[194] G. Mao and B. Fidan, *Localization Algorithms and Strategies for Wireless Sensor Networks.* Information Science Reference, 2009.

[195] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *ACM International Conference on Mobile Computing and Networking (MobiCom)*, Sept 2003, pp. 81–95.

[196] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," *IEEE Personal Communications Magazine*, vol. 7, no. 5, pp. 28–34, October 2000.

[197] R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a global coordinate system from local information on an ad hoc sensor network," in *IPSN'03 Proceedings of the 2nd international conference on Information processing in sensor networks*, 2003, pp. 333–348.

[198] D. Niculescu and B. Nath, "Ad Hoc Positioning System (APS) Using AOA," in *IEEE Conference on Computer Communications (Infocom)*, vol. 3, April 2003, pp. 1734–1743.

[199] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Transactions on Information Systems*, vol. 10, no. 1, pp. 91–102, January 1992, also published as Technical Report 92.1.

[200] G. Borriello, A. Liu, T. Offer, C. Palistrant, and R. Sharp, "Walrus: wireless acoustic location with room-level resolution using ultrasound," in *MobiSys 05: Proceedings of the 3rd international conference on Mobile systems, applications, and services.* ACM Press, 2005, pp. 191–203.

[201] K. Vandikas, L. Kriara, T. Papakonstantinou, A. Katranidou, H. Baltzakis, and M. Papadopouli, "Empirical-based analysis of a cooperative location-sensing system," in *Autonomics '07: Proceedings of the 1st international conference on Autonomic computing and communication systems*, 2007, p. 2.

[202] P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF-based User Location and Tracking System," in *IEEE Conference on Computer Communications (Infocom)*, March 2000, pp. 775–784.

[203] N. K. Sharma, "A weighted center of mass based trilateration approach for locating wireless devices in indoor environment," in *4th ACM international workshop on Mobility management and wireless access*, 2006, pp. 112–115.

[204] M. Roslee and S. N. Othman, "Node Positioning in ZigBee Network Using Trilateration Method Based on the Received Signal Strength Indicator (RSSI)," *European Journal of Scientific Research*, vol. 46, no. 1, pp. 48–61, 2010.

[205] J. Hightower, R. Want, and G. Borriello, "SpotON: An indoor 3D location sensing technology based on RF signal strength," University of Washington, Tech. Rep. UW CSE 2000-02-02, 2000.

[206] K. Chintalapudi, A. P. Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *MobiCom '10: sixteenth annual international conference on Mobile computing and networking*, 2010, pp. 173–184.

[207] A. M. Ladd, K. E. Bekris, A. Rudys, G. Marceau, L. E. Kavraki, and D. S. Wallach, "Robotics-based location sensing using wireless ethernet," in *ACM International Conference on Mobile Computing and Networking (MobiCom)*, Sept 2002, pp. 227–238.

[208] V. Seshadri, "A bayesian sampling approach to in-door localization of wireless devices using received signal strength indication," Department of Computer Science and Engineering, University of Texas at Arlington, Tech. Rep. CSE-2003-35, 2003, also submitted as an M.S. thesis.

[209] F. Evennou and F. Marx, "Improving positioning capabilities for indoor environments with wifi," in *EUSIPCO 2005*, Sept 2005.

[210] A. Howard, S. Siddiqi, and G. S. Sukhatme, "An experimental study of localization using wireless ethernet," in *The 4th International Conference on Field and Service Robotics*, July 2003, pp. 145–153.

[211] B. Li, "Terrestrial mobile user positioning using tdoa and fingerprinting techniques," Ph.D. dissertation, School of Surveying and Spatial Information Systems, University of New South Wales, Feb 2006.

[212] J. Letchner, D. Fox, and A. LaMarca, "Large-scale localization from wireless signal strength," in *AAAI*, July 2005, pp. 15–20.

[213] Cisco, "Wi-fi based real-time location tracking: Solutions and technology," Cisco Systems," White paper, 2006.

[214] Y. Gwon, R. Jain, and T. Kawahara, "Robust indoor location estimation of stationary and mobile users," in *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, March 2004, pp. 1032–1043.

[215] B. N. Schilit, A. LaMarca, G. Borriello, W. G. Griswold, D. McDonald, E. Lazowska, A. Balachandran, J. Hong, and V. Iverson, "Challenge: Ubiquitous location-aware computing and the 'Place Lab' initiative," in *1st ACM International Workshop on Wireless Mobile Applications and Services on WLAN (WMASH 2003)*, September 2003, pp. 29–35.

[216] T. Gallagher, B. Li, A. Kealy, and A. G. Dempster, "Trials of commercial wi-fi positioning systems for indoor and urban canyons," in *International Global Navigation Satellite Systems Society IGNSS Symposium 2009*, 2009.

[217] A. Popescu, "Geolocation API Specification," W3C, Tech. Rep., 2012.

[218] M. Maroti, B. Kusy, G. Balogh, P. Volgyesi, K. Molnar, A. Nadas, S. Dora, and A. Ledeczi, "Radio interferometric geolocation," in *3rd ACM Conference on Embedded Networked Sensor Systems (SenSys) (2005)*, 2005, pp. 1–12.

[219] N. Barber, "Replenishment at sea (RAS) positioning project," University of Durham, School of Engineering," 3H Design Report, March 2000.

[220] L. Girod, M. Lukac, V. Trifa, and D. Estrin, "The design and implementation of a self-calibrating distributed acoustic sensing platform," in *SenSys 06:*

Proceedings of the 4th international conference on Embedded networked sensor systems, 2006, pp. 71–84.

[221] S. Sen, R. R. Choudhury, and S. Nelakuditi, "SpinLoc: Spin Once to Know Your Location," in *HotMobile '12: Proceedings of the Twelfth Workshop on Mobile Computing Systems and Applications*, 2012, p. 12.

[222] P. Steggles and S. Gschwind, "The Ubisense Smart Space Platform," in *The 3rd International Conference on Pervasive Computing*, May 2005, pp. 73–76.

[223] P. M. Maxim, S. Hettiarachchi, W. M. Spears, D. F. Spears, J. Hamann, T. Kunkel, and C. Speiser, "Trilateration localization for multi-robot teams," in *Sixth International Conference on Informatics in Control, Automation and Robotics, Special Session on MultiAgent Robotic Systems (ICINCO08)*, 2008, pp. 301–307.

[224] C. Depenthal and J. Schwendemann, "IGPS — a new system for static and kinematic measurements," *Optical 3-D Measurement Techniques IX*, vol. 1, 2009.

[225] A. Ward, A. Jones, and A. Hopper, "A new location technique for the active office," *IEEE Personal Communications*, vol. 4, no. 5, pp. 42–47, October 1997.

[226] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, "The anatomy of a context-aware application," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, 1999, pp. 59–68.

[227] W. M. Spears, J. C. Hamann, P. M. Maxim, T. Kunkel, R. Heil, D. Zarzhitsky, D. F. Spears, and C. Karlsson, "Where are you?" in *Swarm Robotics, Second International Workshop, SAB 2006*, 2006, pp. 129–143.

[228] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *6th ACM MOBICOM (International Conference on Mobile Computing and Networking)*, August 2000, pp. 32–43.

[229] M. Kushwaha, K. Molnar, J. Sallai, P. Volgyesi, M. Maroti, and A. Ledeczi, "Sensor node localisation using mobile acoustic beacons," in *IEEE International Conference on Mobile Adhoc and Sensor Systems*, Nov 2005, p. 491.

[230] A. Savvides, C.-C. Han, and M. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *ACM MOBICOM*, 2001, pp. 166–179.

[231] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*, 2002, pp. 147–163.

[232] J. Lee, Z. Lin, P. Chin, and C. Law, "Non-synchronised time difference of arrival localisation scheme with time drift compensation capability," *IET Communications*, vol. 5, no. 5, pp. 693–699, March 2011.

[233] M. Youssef, A. Youssef, C. Rieger, U. Shankar, and A. Agrawala, "Pinpoint: An asynchronous time-based location determination system," in *4th international conference on Mobile systems, applications and services (ACM MobiSys '06)*, 2006, pp. 165–176.

[234] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan, "Beepbeep: A high accuracy acoustic ranging system using cots mobile devices," in *ACM Sensys Conference on Embedded Networked Sensor Systems*, 2007, pp. 1–14.

[235] J. Barnes, C. Rizos, J. Wang, D. Small, G. Voigt, and N. Gambale, "High precision indoor and outdoor positioning using locatanet," *Journal of Global Positioning Systems*, vol. 2, no. 2, pp. 73–82, 2003.

[236] Locata, "Locata technology primer," Locata Corporation," Version 1.3, Feb 2005.

[237] F. Winkler, E. Fischer, E. Graß, and G. Fischer, "A 60 GHz OFDM Indoor Localization System Based on DTDOA," in *14th IST Mobile and Wireless Communication Summit*, June 2005, pp. 2311–2322.

[238] J. Werb and C. Lanzl, "Designing a positioning system for finding things and people indoors," *IEEE Spectrum*, vol. 35, no. 9, pp. 71–78, September 1998.

[239] D. Giustiniano and S. Mangold, "CAESAR: Carrier Sense-Based Ranging in Off-The-Shelf 802.11 Wireless LAN," in *CoNEXT '11: Proceedings of the Seventh Conference on emerging Networking EXperiments and Technologies*, 2011, p. 10.

[240] J. Hightower and G. Borriello, "A survey and taxonomy of location sensing systems for ubiquitous computing," University of Washington, Department of Computer Science and Engineering, Tech. Rep. UW CSE 01-08-03, 2001.

[241] R. Mautz, "Overview of current indoor positioning systems," *Geodesy and Cartography*, vol. 35, no. 1, pp. 18–22, 2009.

[242] F. A. Jenet and T. A. Prince, "Detection of variable frequency signals using a fast chirp transform," *Physical Review D*, vol. 62, no. 12, p. 122001, November 2000.

[243] F. Jenet, P. Charlton, J. Edlund, L. Wen, T. Creighton, T. Prince, and M. Tinto, "Generalization of the fast chirp transform algorithm," FCT Group, Tech. Rep., May 2003, accessed March 2013. [Online]. Available: http://www.srl.caltech.edu/fct/chirpreport03.pdf

[244] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, electronic version ed. California Technical Pub., 1998.

[245] W. Buescher, *Spectrum Lab manual*, v2.77 ed., February 2012.

[246] T. V. Baak, "Experiments with a PC sound card," LeapSecond.com amateur high precision clock analysis, 2005, accessed 2011. [Online]. Available: http://www.leapsecond.com/pages/sound-1pps/

[247] *Introduction to the Cospas-Sarsat System (C/S G.003)*, 6th ed., Cospas-Sarsat, October 2009.

[248] J. P. Zagal, J. Rick, and I. Hsi, "Collaborative games: Lessons learned from board games," *Simulation and Gaming*, vol. 37, no. 1, pp. 24–40, March 2006.

[249] J. Von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, 3rd ed. Princeton University Press, 1953.

[250] J. P. Hespanha, M. Prandini, and S. Sastry, "Probabilistic pursuit-evasion games: A one-step Nash approach," in *39th Conference on Decision and Control*, December 2000, pp. 2272–2277.

[251] S. Alpern and S. Gal, *The Theory of Search Games and Rendezvous.* Kluwer Academic Publishers, 2003.

[252] A. Y. Garnaev, "A remark on the princess and monster search game," *International Journal of Game Theory*, vol. 20, no. 3, pp. 269–276, 1992.

[253] S. Alpern, "Rendezvous search games," in *Wiley Encyclopedia of Operations Research and Management Science*, J. J. Cochran, Ed. John Wiley and Sons, Inc, 2010.

[254] P. Misra and P. Enge, *Global Positioning System — Signals, Measurements, and Performance.* Ganga-Jamuna Press, 2001.