Further Automating Maritime Navigation

Tom Hosker

Supervisor: David Bernhard

Declaration

To whom it may concern,

I have submitted this document for the Individual Project module of the MSc in Computer Science (conversion course) at the University of Bristol.

I hereby declare that all the work described and carried out herein is my own, and that it has not been submitted for any other qualification, except where stated explicitly to the contrary.

Tom Hosker

Acknowledgments

I owe a double debt to my supervisor, **David Bernhard**. Firstly, for agreeing to and continuing to supervise a slightly unusual project; but secondly, and more importantly, because he was loyal to me and to this project, and stuck his neck out for me during a difficult patch, when he really didn't have to. My only regret is that my sole means of repaying his kindness is writing what an all-round wonderful chap he is here. In a similar vein, **Conor Houghton** went above and beyond the call of duty helping me through the same difficult patch.

The hardware aspects of this project would have been little more than vague aspirations without the help of the Graduate Teaching Technology people, particularly **Beth Cotterell**. She never had anything but enthusiasm for my ideas and designs, of a sort which people normally reserve for the etchings of young children. Paired, as it always was, with perceptive and practical advice, the progress I've made would have been impossible without her. **James Filbin** and his assistants in the Hackspace were likewise indispensable.

Executive Summary

1. What the Project is About

1.1. Project Type. My project is a mixture of software and hardware, roughly in a ratio of 60% software to 40% hardware in terms of time spent.

1.2. Motivation. A ship's officer of the watch has to juggle a long list of duties in keeping his watch, including collision avoidance and monitoring ship stability and distress signalling equipment. But most of his attention is devoted to navigation. If navigation can be automated – that is, substantially more automated than it is at present – then this will allow the officer of the watch to spend more time on other safety critical duties, leading to safer ships, safer passengers and safer cargoes.

1.3. Aim. To devise a system of navigation which is substantially more automated than are used at present.

1.4. Objectives.

- To automate the verification of GPS using radar overlays;
- To automate the verification of GPS using heavenly bodies;
- To integrate the above into one system.

2. How the Project will be Conducted

2.1. Methodology. I will begin by building a program which can process and compare images, and which I will design according to a small set of images which I have selected or drawn myself. I will then gather real radar data from a vessel underway, and test my program against these data, making any prudent amendments. At the same time, I will develop a piece of hardware which is capable of measuring the position of a heavenly body in the sky, for which I shall devise a similar real world evaluation.

2.2. Deliverables.

- A program which compares radar data with a GPS position, and determines whether GPS has been verified or falsified;
- A piece of hardware which is capable of measuring the position of a heavenly body in the sky;
- A program which integrates the above into one system.

2.3. Added Value. Work done on this project will transform a set of disparate ready-made components – e.g. image processing libraries, accelerometers, Arduinos – into a more automated system of navigation than has yet been achieved.

Contents

CHAPTER 1

Introduction

This project report was compiled and submitted as part of the Individual Project module of the MSc in Computer Science conversion course at the University of Bristol in August and September 2018.

1. Background

1.1. Officer of the Watch.

1.1.1. *Definition.* In the popular imagination, any vessel large enough to be called a ship is at all times under the direct command of her captain: a cheerful old fellow with a neat white beard and sporting a peaked cap, who steers his vessel from an enormous wheel, all the while entertaining his crew with ribald banter. In reality, ships tend to remain at sea for many days at a time, even captains have to sleep occasionally, and, in any case, as with most senior officials the majority of a captain's job is paperwork. Thus a captain will delegate charge of his ship to a deputy, called an *officer of the watch*. Indeed, on most ships there are three such officers of the watch, who will almost always keep to the following timetable:

Time	OOW
0001-0400	Officer A
0400-0800	Officer B
0800-1200	Officer C
1200-1600	Officer A
1600-2000	Officer B
2000-2359	Officer C

TABLE 1. The standard watch pattern

1.1.2. *Duties.* The officer of the watch has charge of the ship. In practice, this means ensuring two things:

- Keeping the ship safe;
- Keeping the ship on her track, and on schedule.

While other factors certainly come into consideration – e.g. collision avoidance, monitoring the ship's stability, monitoring any ongoing operations, monitoring distress signalling equipment, etc – *navigation* is at the heart of both of the above. Thus, for most of his watches, navigation takes up most of a watchkeeping officer's attention.

1.1.3. *Strains and Stresses.* Compared to the general population, the life of an officer of the watch is hard. Having charge of a ship for eight hours a day, seven days a week, demands a great deal of concentration, and is exhausting, on top of which almost all officers will have other duties, such as maintaining life-saving apparatus, conducting cargo operations and passage planning. And compounding the strain of the job is the separation from loved ones, often for many months. To my knowledge, no formal academic study has been made on this subject, but, taking into account data published by the Mission to Seafarers [29], as well as my own experiences working at sea, it seems that seamen are far more vulnerable to major depression, alcoholism and suicide than the general population. Thus, if we think of an officer of the watch as being frequently tired, anxious and distracted, we shall have a reasonable estimation of his state of mind. And such states of mind, of course, may have a serious impact on a watchkeeping officer's abilities as a navigator.

1.2. Navigation.

1.2.1. *How Things Used to be Done.* From time immemorial until around 2005, navigation was conducted by making a series of pencil marks on a paper chart. The main task of the officer of the watch was to calculate a *fix* – based on visual bearings, dead reckoning, radar ranges, celestial data and, as GPS came into use during the 90s, GPS coordinates – and mark this position with a cross within a circle, with a time. A fix always corresponds to a past event – a standard policy is to fix every thirty minutes – so the officer of the watch would also place *expected positions* on the chart, the vessel's current position being somewhere between the last fix and the next expected position. In short: the job of an officer of the watch in his role as navigator was to fix his vessel's position every half an hour, and to have an educated guess for this position between times.

1.2.2. *How Things are Done at Present.* Electronic charts were introduced in the late 90s, and became mandatory for all commercial vessels from 01 Jul 2018. As expected, they changed the nature of navigation radically, and largely for the better. The main achievement of electronic charts is that GPS fixes are now added in an automatic, live and continuous fashion, which has had two interesting consequences:

- The task of the officer of the watch is no longer to produce fixes, which are made automatically, but to verify that GPS is working correctly;
- Fixes now correspond to current events, and not the past.

As for verifying (or falsifying) GPS, the officer of the watch achieves this by placing a manual fix on the chart. Such manual fixes will usually be made by taking visual bearings of prominent fixed objects, and/or radar ranges of points on the shore. If the vessel is too far out to sea to take such bearings or ranges, and if the weather and time of day will allow, the officer of the watch may then fix his position by celestial means, which involves taking the heights of heavenly bodies using a sextant and then calculating a fix literally by hand, with pen and paper, and consulting printed tables. In extreme circumstances, or for training purposes, he may also resort to more exotic means, such as taking soundings and horizontal sextant angles.

1.2.3. *Radar Integrated Overlay.* An interesting means of verifying GPS which has been developed in recent years is the so-called Radar Integrated Overlay

(RIO). By pressing a button on the electronic chart's console, RIO is activated, causing the splodges from the radar feed to be laid over the chart. At sea, radar mostly picks up the shoreline; thus, if the shoreline on the chart coincides with the shoreline which the radar has found, then this is strong evidence that one's GPS position is correct. Note that, at present, RIO is essentially a manual means of verification; the officer of the watch still has to check visually that the radar and the chart coincide.

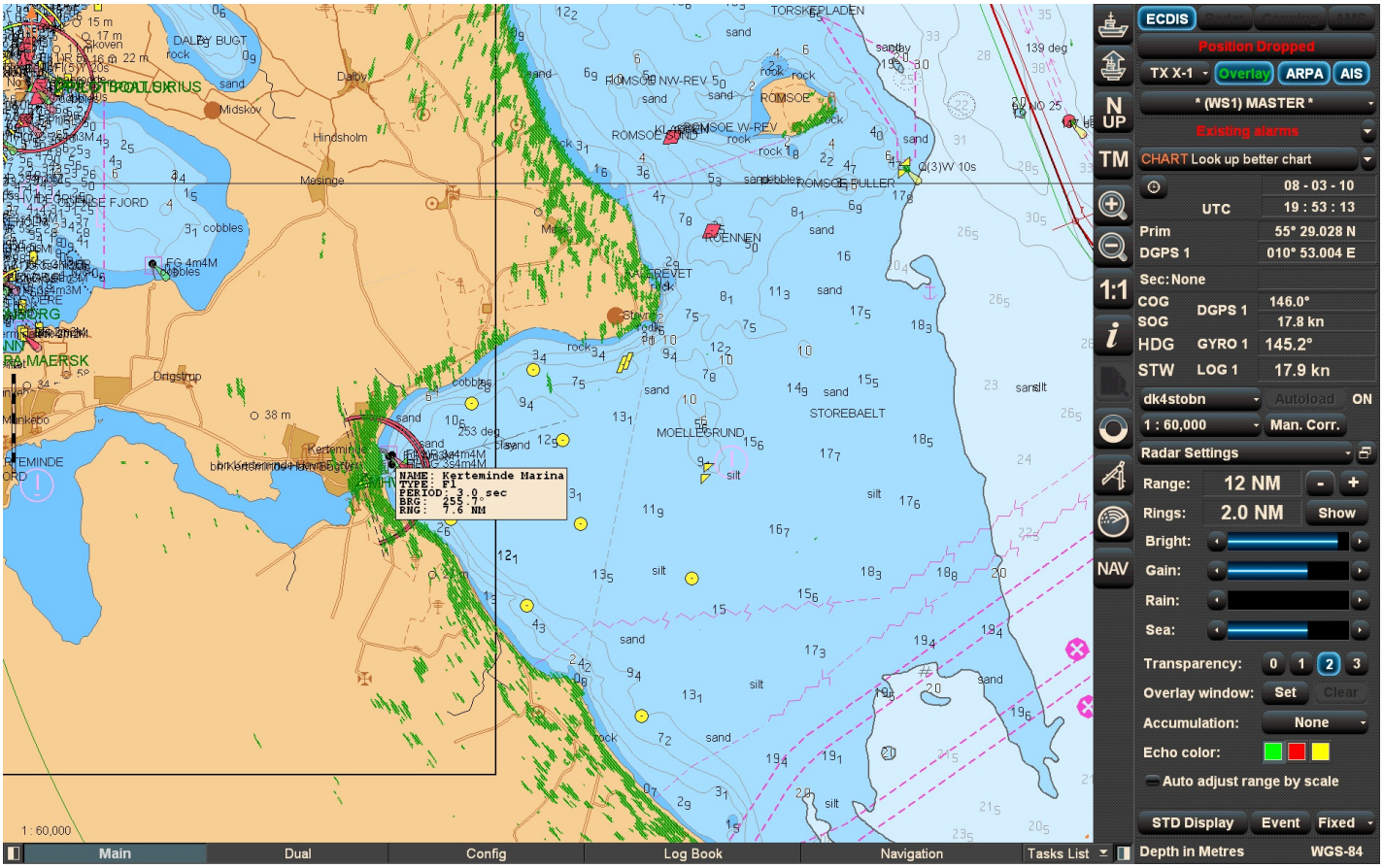

FIGURE 1. An example of an electronic chart with RIO. The dark green splodges are from the radar feed, and the light yellow regions represent land

2. Motivation

2.1. Problems with the Current Method. The current method of navigation, which we might call the *manual verification method*, is basically sound, and is substantially better than the previous, entirely manual, method. But it does have a number of correctable weaknesses.

The most prominent of these weaknesses is that, for the verification or falsification in question to be done correctly, it relies on the competence and conscientiousness of the officer of the watch. As I pointed out above, even the

best officers of the watch are prone to fatigue, distraction and depression. And then there are attacks of forgetfulness, to which all human beings are vulnerable. Taking into account all the other important duties which an officer of the watch has to be discharging on the bridge, this is not a good factor to rely on.

Another obvious weakness of the manual verification method is the length of time taken. For example, it takes around 3-5 minutes for an experienced officer of the watch to plan, execute and plot a fix using visual bearings, and between half an hour and an hour to make a celestial fix. Machines could do this much more quickly.

2.2. Morality and Ethics. There are good economic reasons for improving the current methods of maritime navigation. Groundings are expensive and eminently avoidable, while the improvements which I shall shortly be suggesting are very cheap. However, I tend to think that this kind of discussion misses the point. Instead, the most powerful arguments for improving the current methods of navigation are moral and humanitarian.

When ships run aground, people are often killed, and massive ecological damage can be done. The fates of the *Costa Concordia* and the *Exxon Valdez* are, respectively, the most horrifying examples of this in living memory. Improving the current methods of maritime navigation would reduce the likelihood of such disasters. But, more than that, automation, in relieving some of the enormous mental pressures on the officer of the watch, might reduce the incidence of all manner of maritime accidents.

3. My Proposal

3.1. Automated Verification. In §2.1 above I describe the current method of maritime navigation as a manual verification method, because it involves the officer of the watch checking GPS against his own manual fixes. My proposal is simply to replace this method with an *automated verification method*, wherein the electronic chart itself checks GPS against other data sources. It will, of course, be necessary to adapt the electronic chart and some of the associated hardware in order to facilitate this method – but these adaptations are precisely the meat of my project.

3.2. RIO-A. In §1.2.3 above I discuss RIO, or radar integrated overlay, a very efficient means of verifying GPS which is already widely used at sea. At present, RIO is essentially a manual means of verification, but it very much lends itself to automation. Thus the software component of my project will largely consist of transforming RIO into an automated version thereof, RIO-A, by adapting existing image processing libraries.

3.3. J117. Radar is an excellent navigational tool, and ought to be the primary means by which the electronic chart verifies GPS. But radar, when used for navigational purposes, is only really useful when one's vessel is within 24 nautical miles of land, after which visual bearings of points on the land are also usually impossible to acquire. If we are going to verify GPS this far out to sea, we are going to have to rely on celestial navigation.

In §1.2.2 above I give a condensed account of how celestial fixes are made at present, and in §2.1 I highlight how making such fixes can take a very long time.

Perhaps technological innovations could make this process quicker and more reliable? Thus the hardware component of my project will consist of developing a kind of electronic sextant, which I shall call J117.

4. The Wisdom of Automation

Two individuals have brought to my attention – the first in a less pointed, the second in a more pointed fashion – the question of whether it's really a good idea to automate a manual check. I've thought about this, and discussed the matter with a number of computer scientists, programmers and mariners whom I happen to know, and I've come to the conclusion that this isn't a serious objection.

Automated checks are so much part of the fabric of the modern world that it's easy to take their presence for granted: from the tests which software developers use to build their programs, to the checks performed every time you turn the ignition key on your car. At this stage, it would certainly be unwise to remove manual human checks from the business of navigation *entirely*. And, arguably, it would be prudent to retain some sort of manual check wherever human lives are at stake – although now we're venturing into the realm of philosophy rather than computer science. But it's both reasonable and appropriate for the bulk of checks to be performed by machines: in navigation as it is in almost any other sphere of human life.

CHAPTER 2

Literature Survey

1. Background Material

1.1. Autobiographical Note. I've spent a significant chunk of my life so far working at sea. Indeed, I had the honour of serving as an officer in the British Naval Service for almost five years, including service in the Mediterranean, the Persian Gulf and the Caribbean, as well as the North Sea, North Atlantic and west coasts of England and Scotland. During this time, I earned an Officer of the Watch (Deck) Unlimited certificate of competency. I'm not, by any stretch of the imagination, any kind of secret agent, but I have signed the Official Secrets Act, which restricts what I can write about in so public a document as an academic paper. Nevertheless, I will be drawing, in a general way, on my time at sea throughout this project.

1.2. Correct Navigation and Watchkeeping.

1.2.1. *STCW 95.* The primary source of legislation for how to keep a bridge watch is the *International Convention on Standards of Training, Certification and Watchkeeping for Seafarers*, adopted in 1978, substantially revised in 1995 and amended in 2010, commonly known as STCW 95 [24]. This convention sets out – amongst many other things – the basic expectations as to who is allowed to keep such a watch, and how the watch is to be kept.

Now the conventions of the International Maritime Organisation (IMO) work in the following manner. Once a convention has been adopted by the IMO, then all the member-states thereof are obliged, by the fact of their membership, to incorporate said convention into their national law. For example, in the UK this is achieved via the Merchant Shipping Act and the statutory instruments thereof. Only a handful of nations are not members of the IMO – all of which are landlocked[1] and none of which have significant merchant marines. Thus we can think of STCW 95 as a genuinely global piece of legislation.

1.2.2. *COLREGS.* The next most important source of legislation for how to keep a bridge watch is the set of *International Regulations for the Prevention of Collisions at Sea*, commonly known as the COLREGS or "the Rule of the Road" [25]. As the name would suggest, these regulations are chiefly concerned with dangers of collision rather than dangers of navigation,[2] but the COLREGS also contain important law and guidance which touch on issues of navigation, such as how vessels are to proceed through narrow channels or through fog. Rule 5, in particular, is of such relevance as to be worth quoting here in full:

[1]Although, as my Swiss supervisor might be keen to point out, landlocked nations can and sometimes do have substantial merchant marines.

[2]In maritime legalese, the term *dangers of collision* refers to vessels hitting other vessels, whereas the term *dangers of navigation* refers to running aground and the like.

> Every vessel shall at all times maintain a proper look-out by sight and hearing as well as by *all available means* appropriate in the prevailing circumstances and conditions [my emphasis] so as to make a full appraisal of the situation and of the risk of collision.

The COLREGS entered into British law, and the laws of most other nations, via the same route as other IMO conventions.

1.2.3. Bridge Procedures Guide. Produced by a cabal of shipowners, men with a real stake in keeping their property safe from harm, the *Bridge Procedures Guide* is chiefly concerned with what to do in various emergency situations, but it also contains excellent advice on general bridge watchkeeping [22]. Written with a marked commitment to clarity and readability, this is a first class no-nonsense guide to bridgemanship.

1.2.4. The Mariner's Handbook. Published by the UK Hydrographic Office, *The Mariner's Handbook* contains 99% of the information any seaman would ever need to know – or where to look find it [42]. Of particular interest is Chapter 2, which concerns the use of navigational equipment.

The *Handbook* also has something like the force of law, being a "statutory publication", that is a publication which every British-flagged vessel over a certain tonnage is obliged to carry.

1.2.5. *Cockcroft.* A somewhat jaundiced but very wise old captain once said to me that the COLREGS were not written to help seamen avoid collisions, but to help lawyers apportion blame after the fact. Cockroft's excellent *Guide*, known to mariners simply as "Cockcroft", translates the COLREGS into more practical wisdom [7].

1.3. Radar Equipment.

1.3.1. *SOLAS.* When inventing any new maritime navigational equipment, it would be prudent to consult the *International Convention for the Safety of Life at Sea*, known as SOLAS. This contains, amongst other things, extensive standards to which bridge equipment has to comply [23].

Again, SOLAS entered into British law, and the laws of most other nations, via the same route as other IMO conventions.

1.3.2. *NPS Seminar.* Published by the United States Navy's Naval Postgraduate School, these seminar slides are an excellent summary of the theory behind maritime radars, and address, albeit briefly, all the main problems and issues which radar is known to inflict on an officer of the watch [27].

1.3.3. *NGA Manual.* Published by one of the more improbable United States' government bodies, the National Geospatial-Intelligence Agency,[3] this is a more detailed look at the same topics as were covered by Jenn's seminar [31]. The first chapter seems to be of particular interest at this stage.

1.3.4. Simrad X-Band User Manual. Non-military vessels use two kinds of radar system almost exclusively. These two kinds are distinguished by the wavelength used: the 3cm "X-Band" radar, and the 10cm "S-Band" type. Since the strengths and weaknesses of each are nicely complimented by the other, most vessels large enough to be called ships will carry both system. However, as per SOLAS, only the carriage of the X-band type is a legal obligation. Moreover,

[3]Formerly known as the National Imagery and Mapping Agency.

the X-Band is the type primarily used for navigation – the smaller wavelength allowing for a sharper image of the coastline – and is the type used by RIO – although using S-Band in this regard is presumably possible. Thus X-Band might reasonably be thought of as the "main" radar type, and I shall treat it as such from here onwards.

1.4. GPS. Before I really get into this subsection, I should clear up a matter of terminology. On the one hand, my project's deliverables are all means of verifying satellite navigation systems – that is, satellite navigation systems in general, and not any one such system in particular. And yet, strictly speaking, the term "GPS" refers to a specific satellite navigation launched by the United States Department of Defense. Indeed, there are other such systems: Russia's Glonass, China's BeiDou, the EU's Galileo. So I should – again, strictly speaking – talk about a Global Navigation Satellite System (GNSS) rather than GPS. However, in practice the term GNSS is not widely understood, and the overwhelming majority of GNSS receivers used at sea are indeed GPS receivers. Thus I shall use the term "GPS" to mean both GPS specifically and GNSSs more generally. The context will determine which of the two I mean.

That being said, Glonass does have a substantial number of users. And, while being somewhat less reliable in general, it does enjoy, in addition to a more pleasant name than an anodyne initialism, a number of advantages over its main competitor, namely:

- More accurate positioning at high latitudes [16];
- Independence from the political interference to which GPS is sometimes subjected.[4]

I don't foresee any reason why I would need to compare the advantages of Glonass and GPS respectively any further in my project. But if such a reason does crop up, I'll be sure to make the most of it.

1.4.1. *"GPS Fundamentals".* Maersk, the famous shipping company, have produced an excellent series of concise articles on various navigational aids. This is one of them, and it covers everything a mariner would need to know off the top of his head about how GPS works and, more importantly, the errors from which it can suffer [2].

1.4.2. *"DGPS Fundamentals".* GPS is often subject to errors which are more or less constant over large regions of the globe – either from natural factors such as ionospheric disturbances or from political interference. Differential GPS (DGPS) is a means of correcting such errors. A ground station – which definitely knows where it is on the globe – monitors where GPS fixes its position, and then calculates the difference from the truth. This difference is then passed to DGPS users in the region, allowing for more accurate GPS fixes.

This document, also produced by Maersk, explains DGPS and the associated issues in a bit more detail than the potted version which I've just given [1].

1.4.3. *Misra & Enge.* Professors Pratap Misra and Per Enge put together the latest version of their monumental *Global Positioning System: Signals, Measurements, and Performance* in 2010. While this is considered the standard work

[4]Of course, Glonass is vulnerable to political interference – but not from the United States Federal Government.

on the subject, at almost six hundred pages I will treat this as the reference of last resort [28].

1.5. Electronic Charts. Another matter of terminology needs to be cleared up here. Although tentative experiments were underway for some years before then, electronic charts only began to be used by significant numbers of vessels in this century. Before then, paper charts were used; and paper charts formed the model for electronic charts. Thus the first electronic charts were simply scans – little more than enormous image files – of paper charts, and were called "raster charts" – although, sadly, such documents have little to do with Bob Marley. RNCs – "Raster Navigational Charts" – and ARCS – "Admiralty Raster Chart Service" – charts are both kinds of raster charts, but it's not worth explaining the distinction between the two here.

Raster charts were then developed into "vector charts". What is a vector chart? One might say that, as a .pdf version of a document is to a .png version of the same, so is a vector chart to a raster chart; a vector chart intelligently records where its features are in relation to each other and what their characteristics are. Indeed, a vector chart allows the characteristics of, for example, a lighthouse to be recorded to a level of detail for which there simply would not be room on a paper or raster chart. A vector chart which then meets certain standards laid down by the IMO is referred to as an "Electronic Navigation Chart" or ENC.

A computer which, in addition to receiving data from various feeds, e.g. GPS, gyrocompass and echo-sounder, displays an ENC in a form usable by a human, and also meets certain standards laid down by the IMO, is known as an "Electronic Chart Display and Information System" or ECDIS. Unlike ENCs, which are published by a few national government agencies, most notably the UK Hydrographic Office, a variety of different ECDIS systems exist, produced by a number of private companies: WECDIS and Transas are examples of these specific ECDIS systems.

While, strictly speaking, the correct term for an electronic chart is an "ENC" and the correct term for the machine which displays it is the "ECDIS", in practice I feel that this is unnecessary jargon, which, like the term "GNSS", is not widely understood. Thus from here onwards I shall use the term "electronic chart", or even just "the chart" – paper charts very much going the way of the dodo these days – to refer to both the ENC and the ECDIS on which it is displayed, and I shall only use the more technical terms where such precise language is really necessary.

1.5.1. Practical Use of ENCs. This publication by the UK Hydrographic Office is the standard publication on the correct use of electronic charts. It also contains some useful information on the official line regarding the correct use of RIO [40].

1.5.2. ECDIS Implementation. Although more of a guide for shipowners transitioning from paper charts to electronic, this is the only other Hydrographic Office publication of general application concerning electronic charts, and as such is probably worth me looking into [39].

1.5.3. Transas Manual. WECDIS is the iteration of ECDIS with which I'm most familiar. However, being restricted to the navies of Nato, WECDIS is not suitable for this project. Transas is a system with which I have a modicum of familiarity, and which is available, for a fee, to the general public, and so I shall

treat it as the standard iteration of ECDIS for the purposes of this project [38]. I shall also endeavour to obtain a copy of the manual for the ECDIS system used on the vessel on which I gather radar data (if Transas is not the system used thereon).

1.5.4. *Charts.* I've come across a nice website, which displays what seem to be Hydrographic Office charts covering the whole of the UK coast [19]. The legality of this is dubious, and, obviously, it would not be an acceptable means of navigation at sea. However, it may well still be a useful guide to the positions of various lights for the purposes of this project.

1.6. Positions of Heavenly Bodies.

1.6.1. Almanac. The Hydrographic Office, in association with the United States Navy, publishes a *Nautical Almanac* every year, which can be used to calculate the positions of various heavenly bodies – the sun and moon, the four useful planets Venus, Mars, Jupiter and Saturn, and the more incandescent stars – as they appear at any point on the globe [43].

1.6.2. *NavPac.* The calculations mentioned in the previous paragraph are not difficult, but they are rather long. Fortunately, the Hydrographic Office also produces a computer program, NavPac, which can perform these calculations rapidly and collate the results for presentation in a useful manner [41].

1.6.3. *Nav Workbook.* If the mathematics of NavPac and the *Almanac* seem overwhelming, an unusually clever mariner has produced a set of calculations which were actually used for navigation at sea, and which may serve as a set of useful worked examples for the above [21].

1.7. Known Faults with Electronic charts and GPS. Errors within GPS itself are extremely rare. To the best of my knowledge – and bearing in mind the difficulty of proving a negative assertion – no serious errors in GPS are known to have occurred since the (first) Gulf War, when deliberate widespread jamming was used [20]. Except for a major war, or the gravest kind of political instability within the United States – in either case, navigation will be the least of one's worries – it is highly unlikely that GPS will ever produce such errors.

Errors within the interface between the GPS feed and the electronic chart, on the other hand, are all too common. These usually occur due to some human error in setting up these systems. Instead of being fed a GPS position, the chart is fed a position from some other source, often a dead reckoning position, causing a discrepancy of anywhere from a few hundred metres to a few nautical miles. The famous groundings of the USS *Port Royal* [8] and the cruise liner the *Royal Majesty* [32] are both good examples of this. *These are precisely the sort of errors which I would hope my RIO-A program could detect.*

A third possibility is errors caused by high latitudes. At the time of writing, ships rarely venture to high latitudes, due to the constraints imposed by sea ice in its various forms: icebergs, the threat to ship stability from ice accumulation on the ship's superstructure, and the possibility of the ship being trapped and crushed by an expanding ice shelf. Thus the performance of GPS at high latitudes is little understood, although, because the constellation of GPS satellites is placed for navigation at less than 65 degrees of latitude, doubts have been raised in this regard [26]. I don't know anyone who works for the British Antarctic Survey, nor am I on friendly terms with any fishermen from north Norway, so I don't have

the resources to investigate this. But climate change may open up the sea routes on the northern coasts of Canada and Russia to commercial traffic, making this a potentially very lucrative topic to study.

2. Image Processing and Comparison

Image processing libraries exist in a great many programming languages. In order to limit this investigation to a number of options which I can reasonably look through in any detail, I'm going to restrict my search to those languages in which I have a decent level of fluency: namely C, Java and Python.

2.1. C.

2.1.1. *STB.* Named for the initials of its creator, STB is a relatively new and relatively unknown image processing language for the C programming language, so much so that it only really has a GitHub page in the way of formal documentation [4]. STB is minimalistic, yet it contains exactly the facility for loading up an image as an array of integers which is key to the program that I want to build.

The advantages of STB are really an exaggerated version of the severe virtues of C itself: it's clearly very stripped down, and some initial experiments indicate that it's extremely fast. On the other hand, doing anything more complicated with this library may be an uphill struggle, and I anticipate some angst when trying to integrate image processing into the other branches of my project.

2.1.2. *CIPS.* An older and more established alternative to STB is the C Image Processing System, or CIPS. This library is certainly better documented than its competitor, having a whole book of tutorials and explanatory material devoted to it [34].

The advantages of CIPS are that, being older and more established, it is less prone to unforeseen bugs, and there is more support available. On the other hand, its age does count against it and it may be out of date – although C itself has changed little in the past two decades – and, being a much larger library, the learning curve at the programmer's end may be unnecessarily steep.

2.2. Java.

2.2.1. *ImageIO.* ImageIO is part of the standard Java Development Kit, and thus is very well supported, including an extensive tutorial [9]. As one might expect, ImageIO exhibits the somewhat gentler virtues of Java generally: a middle way between the simplicity and performance of C, and Python's ease of use. Compromise, though, is commonly said to please no-one, and, in using this library, I doom myself to a program which is not as fast as possible, and which could always be easier to build and adapt.

2.2.2. *ImageJ.* ImageJ is another Java image processing library. It's not part of the standard Java Development Kit, and yet, curiously, it's extremely well documented, with an extensive user guide [14]. One reads on the thirteenth page of this text: 'ImageJ alone is not that powerful: it's [*sic*] real strength is the vast repertoire of Plugins that extend ImageJ's functionality beyond its basic core'. This quotation suggests that this library is not a good fit for this project. What I need instead is a very simple library which is easy to use, and which will be easy to integrate with other, not image related, branches.

2.3. Python.

2.3.1. *Pillow.* Pillow is a fork of the now defunct Python Imaging Library (PIL), from which it presumably derives its name. Unlike, say, Java's ImageIO, it's not quite an official or built-in library, but nevertheless it's very straightforward to install using the so-called pip facility, and is very well documented [6].

As one might expect from a Python library, coding with Pillow is a pleasant, intuitive experience, with minimal verbiage. Also, thanks to libraries such as pySerial, it would be very easy to integrate an image processing program with an Arduino-based hardware design. The great drawback of Pillow, as with any Python library, is the glacial slowness of an interpreted language. And, indeed, initial experiments indicate that Pillow can take as long as a few seconds to walk through each pixel of a large image.

2.3.2. *OpenCV.* OpenCV is a possible alternative to Pillow within Python, and also has good documentation [30]. Its main advantage over its competitor is speed: initial experiments indicate that OpenCV can perform basic image processing tasks two or three times more quickly than Pillow. On the other hand, OpenCV was built for the purposes of computer vision, as the name might suggest, and not for the simpler task of image processing. As such, OpenCV is much more verbose, and, in comparison to Pillow, rather tortuous from the point of view of the programmer. One could argue that, if performance is such an overriding concern, why are we considering using Python in the first place?

3. Hardware

3.1. Sensitivity to Attitude and Azimuth. In principle, building a device sensitive enough to its own attitude and azimuth that it could *of some use* for celestial navigation is clearly feasible, because this has been achieved already: see, for instance, the Sky Map Android application that Google developed [18]. However, two related propositions are less certain. Firstly, is it feasible, in principle, to build an electronic device that could be used to achieve celestial fixes of such accuracy that it would rival traditional celestial navigation? Secondly, is it feasible for me to build such a device myself?

I had a chat about these questions with the Graduate Teaching Technology people, who were very helpful, and who suggested that the following might be a sensible roadmap towards answering them:

- Identify the components of a smartphone which make it sensitive to attitude and azimuth;
- Obtain versions of said components;
- Glue the components into a device;
- Add to the device the necessary hardware and software so that it will output, perhaps to a screen, its current attitude and azimuth;
- Test whether the device is capable of finding the horizon and north;
- Add to the device the necessary hardware and software so that, given the time of day, the attitude and azimuth of the sun from a specific point on the globe can be determined;
- Test whether the device is capable of finding the sun from the before mentioned point on the globe.
- Repeat the previous two steps, but replacing the sun with Venus.

The first of these steps, of course, I can and will do now. In a helpful article, the website `gizmodo.com` states that a smartphone uses three components to determine its current attitude and azimuth [33]:

- An accelerometer,
- A Mems gyroscope,
- A magnetometer.

Let's have a look at these in a bit more detail:

3.1.1. *Accelerometer.* Dimension Engineering, a private company which produces electronic components, provides a good introduction to the general principles behind accelerometers [10]. They also provide a detailed description of a specific accelerometer which they make and sell [11]. This source, of course, has to be treated with a pinch of scepticism, since the primary motivation behind it is to sell a product.

3.1.2. *Mems Gyroscope.* The acronym "Mems" stands for "Micro-Electro-Mechanical Systems". A group of engineers at Northwestern University have put together a detailed account of how Mems gyroscopes work, as well as a few pages on possible applications [5].

3.1.3. *Magnetometer.* The accelerometer and gyroscope are each capable of determining the attitude of a device, and together can produce an even more accurate estimate thereof. In principle, these devices are also capable of determining azimuth, although whether they're actually used in this way, and how accurate they are in this regard, seems to vary from one model of smartphone to another. The magnetometer, on the other hand, is solely to do with determining azimuth.

Now magnetic bearings are not suitable for maritime navigation. Indeed, I myself have served on a ship in a region – the Gulf of Mexico – where the variation between true north and magnetic north was in excess of ten degrees. And ships themselves, being large objects made of steel, generate significant magnetic fields. Hence I'll have to ensure that my device determines its azimuth using the accelerometer and gyroscope only, or from another component which does not use magnetic north as a reference.

In addition to the above, Google has also published a brief guide to the motion sensors of which Android applications will typically make use [17].

3.2. Microcontrollers.

3.2.1. *Arduino Uno.* Any device which I build will need some kind of microcontroller, if only to pass data from the sensitive components onto a more traditional computer. Following a number of conversations with the Graduate Teaching Technology people, it was very much their opinion that the microcontroller best suited to my current needs and abilities was the Arduino Uno.

Arduino, the company which produces this microcontroller, publishes a continually updated data sheet [3]. Having read through this document and having conducted some informal experiments with this component, I conclude that the Arduino Uno offers a number of specific benefits, namely:

- The component is well supported, both by Arduino itself and by the wider community of component producing companies. See, for instance, SparkFun's *ADXL345 Hookup Guide* [37].

- The language which is used to program the component is very close to C. Thanks to modules which I've taken earlier in my current course, I'm more than comfortable programming in this language.
- The component is designed to interact well with the Ubuntu operating system. Given that this is the operating system installed on my main work laptop, the component should dovetail nicely with the rest of my project.
- The Graduate Teaching Technology people are very familiar with the component, and are both willing and able to help me with my project, a valuable source of expertise.

The only real drawback which I've been able to identify with the Arduino Uno, at least when compared to the leading alternatives, is the marginally higher cost. But, given the relative cheapness of all these components – none are much more the £20 – this is unlikely to be a significant factor in a research project.

3.2.2. *Alternative Microcontrollers.* A number of imitations of the Arduino Uno are available, including the Freeduino [15] and colourful Diavolino [12]. These imitations aim at being as close as possible to the real thing, so the their only real advantage is their slightly lower cost. Given that, in this context, price is not such a sensitive issue, the Arduino Uno is worth the extra £5.

Somewhat different small computers are available, such as the Raspberry Pi [35]. The crucial disadvantage here is that these components are capable of doing very much more than I require. The Raspberry Pi 3, for instance, is quite capable of running a full Ubuntu operating system. Even putting financial considerations to one side for a moment, I've only been programming for a year, and have next to no hardware experience. Things will go more smoothly with a simpler component.

CHAPTER 3

Execution

Approach

0.1. Five Milestones. As I laid out in the Executive Summary at the beginning of this document, my project is structured around the following five milestones:

(1) Build a RIO-A program, as described in §3.2 of the Introduction;
(2) Test this program against radar data gathered from a vessel underway;
(3) Build a J117 device, as described in §3.3 of the same;
(4) Test J117;
(5) Integrate RIO-A and J117 into one system.

0.2. Agile.

0.2.1. *Sprints, Scrums and Backlogs.* A *sprint* is generally held to be the fundamental component of the agile approach, and is often defined as a fixed period of intensive action towards a specific goal. Note the emphasis on a fixed period of time. A week is a fairly natural unit of time for human beings, and, left to my own devices, is the unit around which I tend to plan my life, so let's make our sprints a week long.

At first glance, the *scrum* – short meetings at the end of, or at regular intervals during, each sprint, where the team evaluates the progress made – seems redundant in the case of an individual working alone. It makes little sense to schedule a meeting with oneself. But on a deeper level the scrum is still a necessary component. Again, left to my own devices, I tend to follow a six day week: Monday to Friday I tend to work away as normal; Saturday is reserved for doing nothing useful whatsoever; then on Sunday, which really is the first day of the week, I tend to look through the tasks still to be done – one might call this my *backlog* – and make any plans or other arrangements necessary for the following five days.

0.2.2. *Expected Timeline.* With the above in mind, I then put together the following timeline. I assumed that a project takes from 10 Jun until 07 Sep. I allotted five and three weeks, respectively, for building RIO-A and J117, and two weeks and one week, respectively, for the testing thereof. The final three weeks were devoted to integrating everything into one system, and writing up.

No	Task	Begin	End
1	Build RIO-A	10 Jun	06 Jul
2	Build J117	08 Jul	27 Jul
3	Collect data; test RIO-A; adapt	29 Jul	10 Aug
4	Test J117	12 Aug	17 Aug
5	Integrate and write up	19 Aug	07 Sep

1. Building RIO-A

The code referenced in this section can be viewed **here**.

1.1. Version 1.

1.1.1. *Version 1.1.* I started off with an extremely simple program, based around a small T-shaped blob of green pixels. This T shape can then be *splodgified* – a term I coined myself, which means that a layer of green pixels is added around the green pixels already present – and a percentage match can be calculated with another image, with or without a variable number of *splodgifactions*.

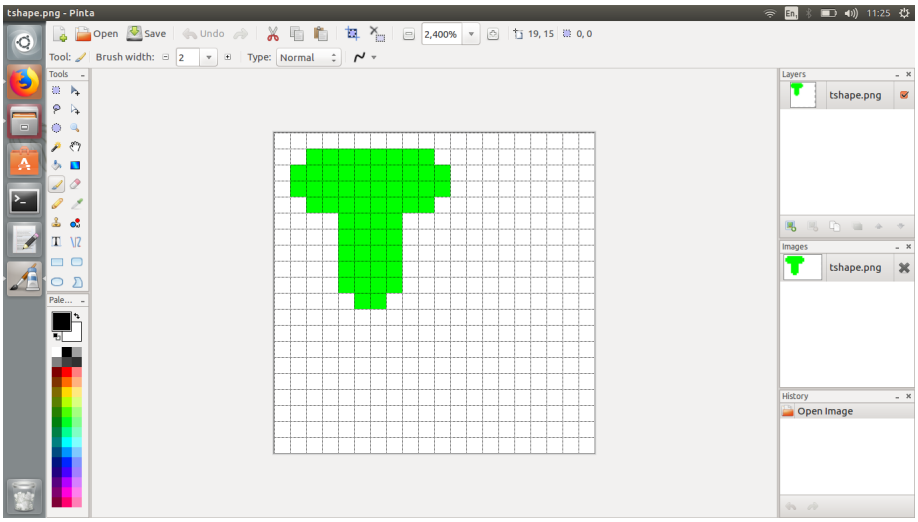

FIGURE 1. The original T shape

1.1.2. *Version 1.2.* I then made some minor adjustments, modifications and optimisations to turn version 1.1 of RIO-A into version 1.2.

1.1.3. *Version 1.3.* I then worked towards adapting my program so that it could examine, not a simple T shape, but an actual nautical chart, and the chart I chose for this was a very large-scale chart – these are often called *planning charts* – of the Gulf of Mexico.

Two important considerations need to be laid out at this point. Firstly, a chart of such a large scale would only be used at sea for planning, and not for plotting fixes; but I use it here because the coastline, particularly the shape of Florida, is well-known and distinctive, and because the scale of the chart makes little difference at this early stage. Secondly, while Electronic Navigational Charts (ENCs)[1] are *vector* graphics files, gaining proper access to such files would require a level of financial resources – in the thousands of pounds – that I simply don't have, as well as specialist software and hardware; thus throughout this project I've had to use the more readily available *raster* graphics equivalents of these files, a necessary evil for which I apologise in all sincerity and meekness.

In order to compare a chart to radar data, it is necessary to convert one of the two into a similar format to the other. Radar data is usually very simple, consisting of green pixels and empty space, so it makes more sense to *simplify*

[1]That is, the chart files approved by the IMO.

the chart, rather than vice versa. Charts are divided into different regions – land, drying heights, and then various depths of water – and then coloured accordingly. The exact colours used vary depending on the hydrographic authority which publishes the chart, but, for every chart I have ever seen, some shade of yellow is used for land, green for drying heights, blue for shallow water and white for deep water. For the raster charts which I've had to use, black (and, to a lesser extent, magenta) is an interesting case, since this is used both to separate regions and to add text and other markings. Thus my program's simplification process consists of (1) simplifying the colours to either yellow (for land) or white (for water) and (2) removing any black pixels by looking at the proximate non-black pixels. Finally, it is usually prudent to splodgify the image at least once.

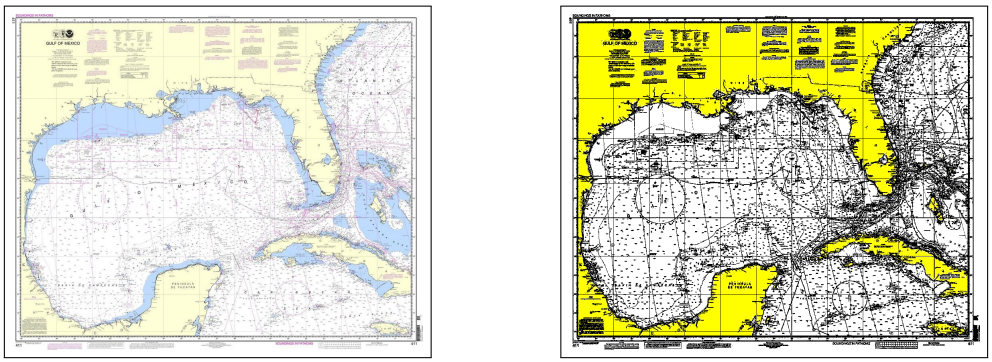

FIGURE 2. The original raster chart (left) and the image with the colours simplified

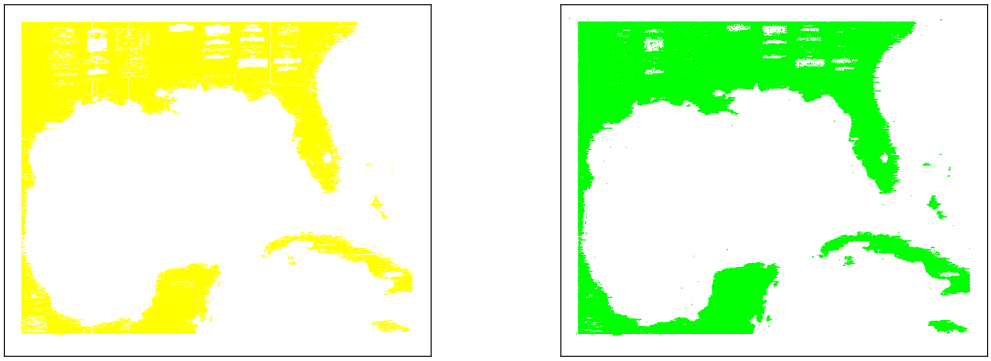

FIGURE 3. The simplified raster chart with the black pixels removed (left) and the image with one splodgifaction

1.1.4. *Versions 1.4 and 1.5.* The next two versions were dedicated to polishing, rather than radically restructuring, my program. As any half-decent programmer would at this stage, I looked for edge cases, and whether any had the potential to break my program, or to return seriously misleading results. The ones I found were all cleared up with only minor revisions to my code.

An important consideration when using manual RIO at sea is when to abandon its use. Official guidelines are usually that radar is not to be used for navigational purposes when more than 24 nautical miles from land; but, for instance, X-band

3cm radar is quite temperamental and affected by subtle changes in weather, and will often cease to return useful data at around, say, 16 nautical miles from land. At this early stage, we can simulate such a loss of data, in a crude way, by removing all of the landmasses in the Caribbean except for Jamaica; and, indeed, with a few simple revisions my program throws up the desired flag when examining such a file.

FIGURE 4. The Caribbean minus everything except Jamaica

1.1.5. *Wrist-and-Shoulder Questions.* How little data, exactly, is too little to verify or falsify GPS? For the time being, I've made the rule that if the number of green pixels is less than one fiftieth of the total number of pixels, then my program has too little data to make a decision either way. But, more broadly, this question is part of a class of questions which I call *wrist-and-shoulder questions*, because such questions are similar to asking, "Where does the hand become the wrist?" or, "Where does the arm become the shoulder?" Wrist-and-shoulder questions require clear and precise boundaries to be drawn whereas, in reality, no such boundaries exist.

Giving some sort of answer to wrist-and-shoulder questions is an unavoidable part of human life, and doubly so for programmers. And, as we shall see, answering such questions in a roughly sensible and scientific fashion will be an important element in the development of this project. Nevertheless, it's worth remembering that any answer to this sort of question is somewhat arbitrary.

1.2. Later Versions. RIO-A was developed further into versions 2.1 to 2.4. However, these versions were built after, and as a response to, the testing of the program against real radar data, and the integration of hardware. Thus versions 2.1 to 2.3 are discussed in §3, and version 2.4 in §5, of this chapter.

2. Building J117

2.1. Version A. J117, as I have built it, is itself a proof of concept, so Version A was a proof of concept for a proof of concept, and thus very simple. It was only a piece of wood with an Arduino Uno and accelerometer fastened to it with bolts.

On 10 Jul, this design was then subjected to the following test: is it capable of finding level? I found a slightly unsteady table, which I could gently and steadily manipulate from level to two or three degrees off from level. Whether or not the table was level was checked independently using an old-fashioned spirit level. As one can see from the below images, the results were entirely satisfactory.

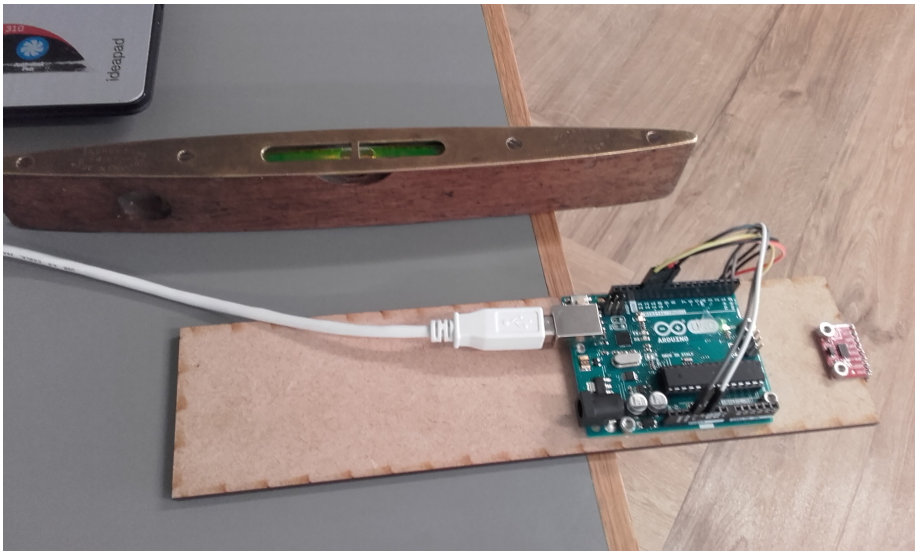

FIGURE 5. J117 on the slightly unsteady table, with the spirit level indicating that said table is currently level

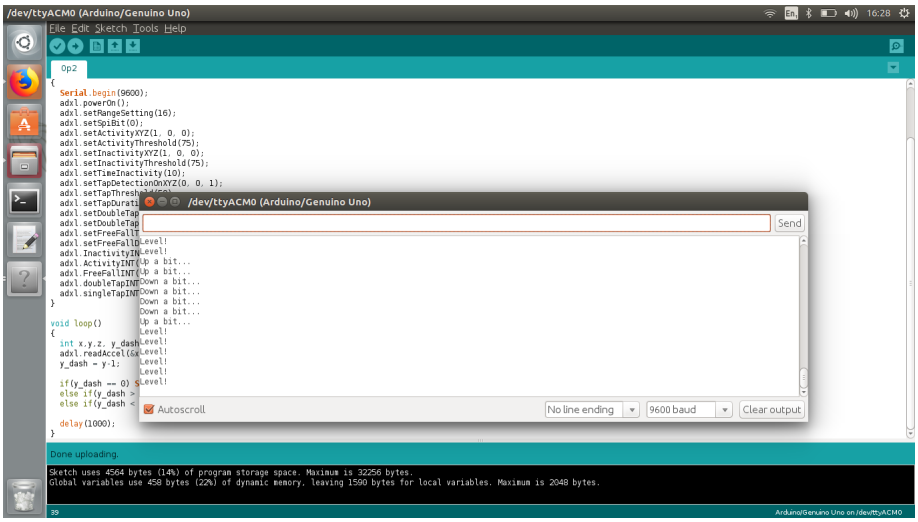

FIGURE 6. Screenshot showing the "serial" output from Version A, as it goes from a non-level to a level surface

2.1.1. *ADXL345.* The accelerometer used in Version A was SparkFun's ADXL345 [37]. While the component performed well in the above test, two major drawbacks became apparent, which meant that it would be unsuitable in building Version B. These were: (1) the component had no sensitivity to azimuth;

and (2) the component only seemed to output attitudes as numbers between 0.00 (for 0 degrees) and 40.00 (for 90 degrees). For these reasons, the component was replaced with another in building Version B.

2.2. Version B. Version B was built by building a similar device to Version A, but with the following modifications:

- The ADXL345 accelerometer was replaced with the Pmod NAV (see below for a discussion of this component);
- A pistol grip style handle was added to the base;
- A system of iron sights, comprised of two bolts in the fore part and one bolt in the rear part of the device, was added.

A clock component was also added to the device, although this later proved to be redundant.

The Pmod NAV was fastened to the wood using hot glue, and also – more to allay my paranoia about the fragility of hot glue than any practical purpose – a rubber band.

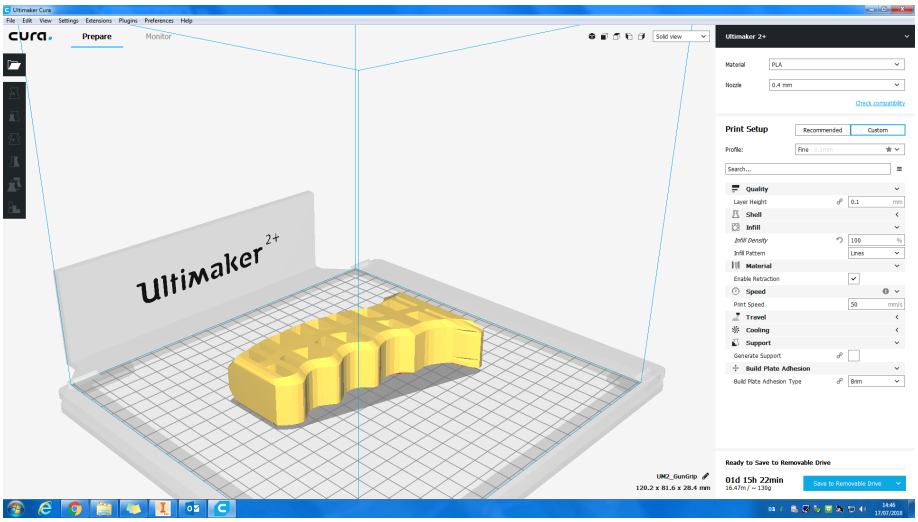

FIGURE 7. My first attempt at 3D printing my pistol grip as it appeared in Ultimaker Cura

I began creating my pistol grip by getting a 3D-printable design from Thingiverse, and, with some help from James Filbin in the Computer Science Department's "Hackspace", converted this into a printable template using Ultimaker Cura. This first attempt at printing the design, however, failed for two reasons:

- Printing the very dense layers of plastic caused the object to be moved by a few millimetres about half way through the print, causing two "out of sync" halves;
- The structures at very shallow angles were not supported, and, while still in a somewhat molten state, collapsed.

Thankfully, James and myself were able to reduce the density of the build, and to add in more support for the shallow angles, and the subsequent build was a success.

FIGURE 8. My first attempt at 3D printing my pistol grip, or what remains of it

The iron sights are simply three bolts, with some nuts added to given them a bit of extra height. The idea is that the user will line the back bolt up with the two front bolts, so that the three bolts all seem to be flush with each other. This provides a common frame of reference for what is considered level – or any other attitude – from which subsequent calibrations can be made.

FIGURE 9. An illustration showing how the iron sights are supposed to be used in measuring the height of a body

2.2.1. *Pmod NAV.* As I mentioned above, Version B uses the Pmod NAV as its sensitive component. Amongst other things, this tiny piece of hardware incorporates: an accelerometer, for measuring its attitude; a magnetometer, for

measuring its azimuth; and a gyroscope, for sharpening the accuracy of the previous two.

When I first starting building Version B, I had hoped to make use of the magnetometer and thus make my device sensitive to its own azimuth. I discuss the problems with using magnetometers at sea in §3.1.3 of my literature survey. However, I had hoped against hope that such problems would atrophy into insignificance on dry land, and, in any case, would be forgivable in a proof-of-concept design. In fact, the Pmod NAV's magnetometer proved to be wildly inaccurate – in the order of dozens of degrees – in several locations throughout Bristol, and despite myself and Beth Cotterell the Graduate Teaching Technologist trying a number strategies to correct it. Thus the azimuth sensitivity aspect of J117 was abandoned.

On a more positive note, J117's attitude sensitivity, via the Pmod NAV, proved to be more than satisfactory. This issue is precisely what is looked at in §4 below.

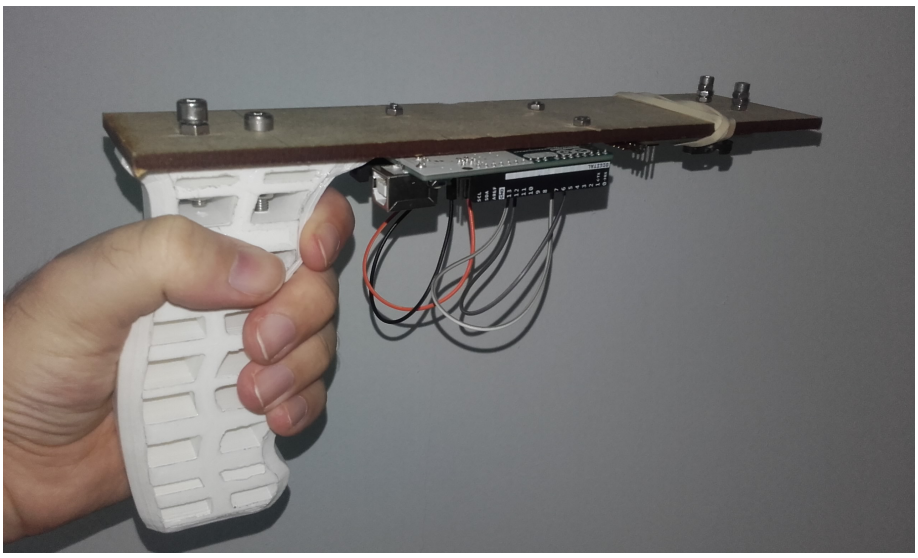

FIGURE 10. J117 Version B

3. Testing RIO-A

3.1. The Voyage of the *Lundy Murrelet*. On 25 Jul, between three o'clock in the afternoon and six o'clock, on the *Lundy Murrelet*, myself and her skipper Colin Eastman gathered radar data in and around the estuary of the rivers Taw and Torridge. The *Lundy Murrelet* was fitted with a Raymarine radar, of a model which is no longer produced, but which is broadly similar to a Quantum 18" radar scanner system [36].

We began collecting data in the sea itself, with the range set to two nautical miles, but, although the wind was calm, there was a heavy swell. The *Lundy Murrelet* is quite a small vessel, and my method for transferring data from the radar display was to take photographs of it using my digital camera; even with the assistance of a tripod I had bought for that purpose, it was impossible to get a good steady series of images of the display on the open sea. Thus we moved

back into the estuary, where the water was very much calmer, and where it was more appropriate to set the range to half a mile, although this meant forgoing an opportunity to gather the kind of littoral data which, ideally, I would have liked to test RIO-A against. Again, I apologise to the reader, and I recognise this as a shortcoming of my project.

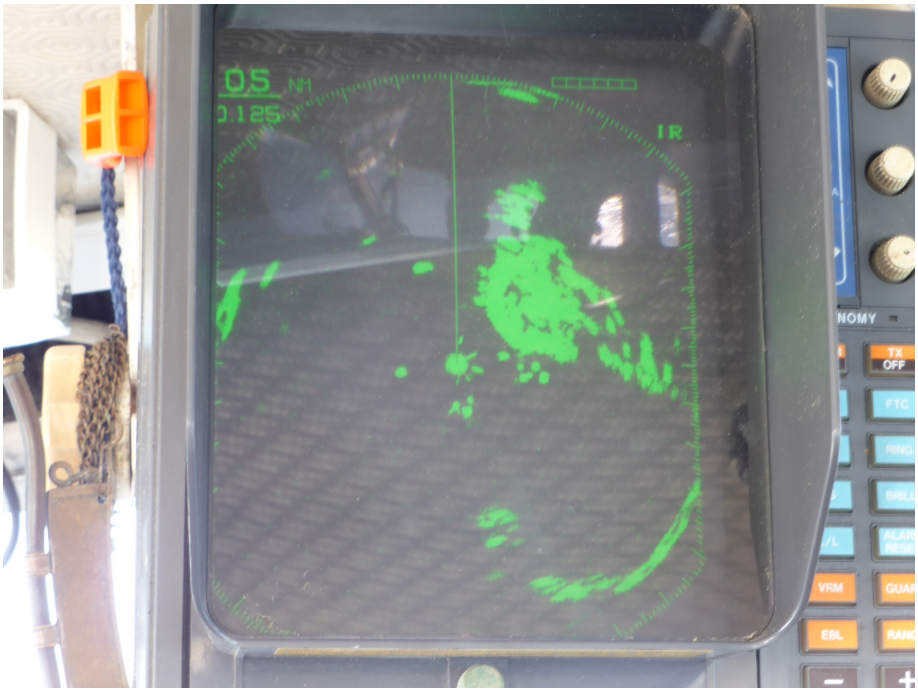

FIGURE 11. Raw radar data from the *Lundy Murrelet*

I then processed the raw data I had gathered into a usable form. The pictures I had taken of the radar display were converted into images composed entirely of green and white pixels, largely by hand. This was painstaking and, as the reader can imagine, very labour intensive. For this reason, myself and my supervisor agreed the best set of 30 images from the data I collected, and processed these, rather than the whole set of almost 400 images.

Fortunately, I'd remembered to turn on the "geo-tagging" facility on my camera before stepping on board the *Lundy Murrelet*. This caused a latitude and longitude, in addition to the time, to be written into the EXIF data for each image. This has allowed me to generate charts, centred on the latitude and longitude in question, which correspond to each image of radar data, an essential ingredient for testing RIO-A.

3.2. Analysis. To recap: RIO-A detects GPS errors from the radar data by finding green pixels where, according to the chart, there ought not to be any. Thus, in adjusting my RIO-A program to analyse the gathered data, it became necessary to answer the following question: how many unaccounted for green pixels, exactly, are enough to conclude that there is a GPS fault? This, of course, is a good example of the wrist-and-shoulder questions I mentioned earlier (see §1.1.5 of this chapter).

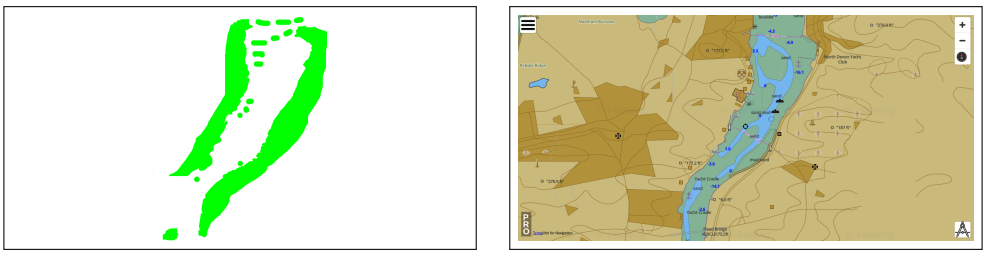

FIGURE 12. An example of the processed radar data, with the corresponding chart

In practice, the way I answered this question was to pick a round number which seemed sensible at first glance, test it out, and, unless it caused any obvious catastrophes, stick with it. By such a process I arrived at a figure of 10,000 pixels unaccounted for.

But a more scientific and, unless time is a serious factor, much better method of answering such questions is to perform the following analysis. Firstly, let's call the figure that we're adjusting Δ. Next, let's define a *false positive* in this context as occurring when a GPS fault is reported when no such fault exists, and a *false negative* as vice versa. Now let's carry out a number of simulations, having RIO-A examine twenty radar data images on each run, with a GPS fault, of variable size and direction, occurring for images 11-20. These are the results:

Run	*GPS Fault*	Δ	*Correct*	*False +ve*	*False –ve*
1	None	10,000 pixels	20	0	N/A
2	None	5,000 pixels	8	12	N/A
3	None	20,000 pixels	20	0	N/A
4	70m east	10,000 pixels	16	0	4
5	70m east	5,000 pixels	15	5	0
6	70m east	20,000 pixels	10	0	10
7	140m east	10,000 pixels	20	0	0
8	140m east	5,000 pixels	15	5	0
9	140m east	20,000 pixels	16	0	4

TABLE 1. Table of simulations in RIO-A with various GPS faults and values of Δ between 5,000 and 20,000 pixels

Such analysis would suggest that my initial guess for Δ of 10,000 pixels was about right. Certainly, 5,000 pixels is far too small a threshold, with lots of false positives, and 20,000 pixels is far too great, with many false negatives. But this analysis is rather crude; 5,000 and 20,000 are rather stark alternatives to 10,000 pixels. What would happen if we tried 8,000 and 12,000 instead?

Please turn over.

Run	*GPS Fault*	Δ	*Correct*	*False +ve*	*False –ve*
1	None	10,000 pixels	20	0	N/A
2	None	8,000 pixels	20	0	N/A
3	None	12,000 pixels	20	0	N/A
4	70m east	10,000 pixels	16	0	4
5	70m east	8,000 pixels	18	0	2
6	70m east	12,000 pixels	14	0	6
7	140m east	10,000 pixels	20	0	0
8	140m east	8,000 pixels	20	0	0
9	140m east	12,000 pixels	15	5	0

TABLE 2. Table of simulations in RIO-A with various GPS faults and values of Δ between 8,000 and 12,000 pixels

Looking through this analysis, setting Δ to 8,000 pixels actually performs better than 10,000 pixels. I wonder what setting it to 7,000 pixels would look like in practice?

Run	*GPS Fault*	Δ	*Correct*	*False +ve*	*False –ve*
1	None	7,000 pixels	17	3	N/A
2	70m east	7,000 pixels	18	1	1
3	140m east	7,000 pixels	18	2	0

TABLE 3. Table of simulations in RIO-A with various GPS faults and Δ set to 7,000 pixels

Since setting Δ to 7,000 pixels results in a slightly worse performance than setting it to 8,000 pixels, and since it's not worth the trouble of looking at fractions of a thousand pixels at this stage, let's keep Δ at 8,000 pixels from here until the end of the project.

Now, having set Δ to an appropriate figure, let's look at the results for RIO-A on a range of GPS faults. This time, we're going to look at the full run of 30 images, which includes an interesting circumstance in the last two images which I'll discuss later, and which consistenly throws up a false positive. Of these 30 images, image 1 and images 11-20 may include a GPS fault, as below.

Run	*GPS Fault*	*Correct*	*False +ve*	*False –ve*
1	None	28	2	N/A
2	700m east	28	2	0
3	1100m south	28	2	0
4	70m east	26	2	2
5	110m south	23	2	5
6	140m east	28	2	0
7	220m south	28	2	0
8	50m east	23	2	5

TABLE 4. Table of simulations in RIO-A with various GPS faults

For a full discussion of the above results, see §2.1 of the Conclusions chapter.

3.3. Versions 2.1-2.4. In order to conduct the above tests of RIO-A, and in response to the same, changes were made to the underlying code. Some low level adjustments were made to allow RIO-A to compare series of images, rather than one image at a time. Then I added some code to check whether GPS was placing our position as over land, which, as one would hope, causes a GPS fault to be reported immediately. At this time I also made the necessary modifications to integrate J117 into RIO-A.

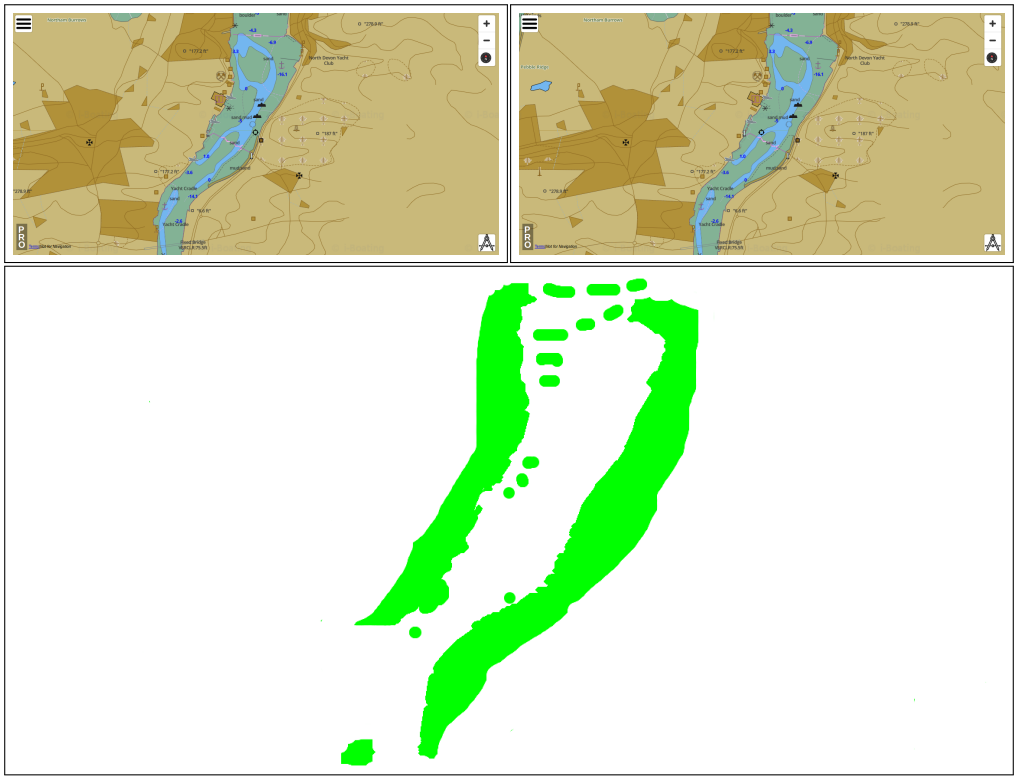

FIGURE 13. Distance to land checks in action: the top left image image shows the incorrect chart position, to the right is the correct position, and beneath is the radar feed. Checking the distance to the east bank alerts the user to the GPS error

3.3.1. *Distance to Land Checks.* Following a demonstration of my progress to Doctors Bernhard and Houghton, which prompted a three-way discussion of the problem of *transversisms* (see §1.2 of the Conclusions chapter for an explanation of this term), I decided to insert a system of primitive *distance to land checks* into my code.

The basic idea here is very simple: in addition to checking the number of unaccounted for green pixels, we can also check if the distance to land on the chart – measured from the centre of the image – matches the distance to land on the radar. Something as apparently straightforward as this, sadly, has its own quirks and subtleties. A major factor is that radar is prone to noise. Indeed, I remember as a young officer being told off for tuning a radar so that it was "gin clear" and not "lightly speckled" as it ought to have been; the light speckles prove that the noise has been tuned down to *just* short of perfection. A naive

distance to land check will simply return the distance to the first green pixel, and will not differentiate between noise and land. There are of course ways of getting round this problem; our check could look, not for the first green pixel, but for the first block of n pixels. But how many pixels together, exactly, is land, and how many is just noise? This is another wrist-and-shoulder question, the answering of which comes with its own, hopefully lesser, quirks and subtleties. Regrettably, I've not found or been able to make enough time to develop this line of enquiry to its conclusion.

Fortunately, the radar system from which I gathered my data suffered from a remarkably low level of noise; the low range that was in use was probably also a factor here. Thus in version 2.4 of RIO-A I've been able to implement a naive distance to land check, with surprisingly pleasing results. For example, with a GPS fault of 140m east, this version of RIO-A is able to detect the error just by performing the distance to land check. Fair enough, the number of unaccounted for green pixels is also well above the required threshold, but this is only because our vessel is in an estuary, and thus we have two distinct edges of land to navigate by. If the west bank of the river were absent, the resulting transversism would mean that no fault would be detected by looking for green pixels; but by checking the distance to the east bank, we would be saved from a false negative. (See figure 13 above for an illustration of precisely this example.)

4. Testing J117

The testing of J117's angle-measuring accuracy was carried out in Bristol Cathedral. This site was chosen because it is a large public building near where I happened to be at the time, the dimensions of which are well known. Indeed, according to `timeref.com`, the length of the nave is 125 feet, and its height is 52ft. I took the measurements sitting down, for greater stability, so my height of eye, while variable, was always close to 4 feet, and my chair was located almost exactly six feet from the west door. Thus, doing some basic trigonometry, the angle of the top far corner of the nave from my position was:

$$\begin{aligned}\theta &= \arctan(\frac{52-4}{125-6}) \\ &= \arctan(\frac{48}{119}) \\ &= 21.96724340481...\end{aligned}$$

Holding up J117 to said top far corner, I had the device's serial output print the angle recorded every second. At the end of the experiment, I took a screenshot on my laptop, preserving the data. I got thirty good readings this way, the mean of which was 21.863636363636, with a standard deviation of 0.85903902818676.

There is more than a little room for interpretation in these results. For one thing, the unsteadiness of my hands was a significant factor in the variance observed. One might also cast doubt on the quality of the data for the cathedral's dimensions: is the nave really *exactly* 52 feet tall? A difference of just six inches will make a significant difference in this context. But I don't have the resources to measure the heights of tall buildings personally, so, for the experiment to having any meaning whatsoever, I have to assume that the data provided is correct.

Having said all that, a rather loose sort of conclusion, that the accuracy of J117 is somewhere between a tenth of a degree and a degree, is perfectly consistent with the results. Splitting the difference, an educated guess, based on the data observed, is that the accuracy of J117 is around half a degree.

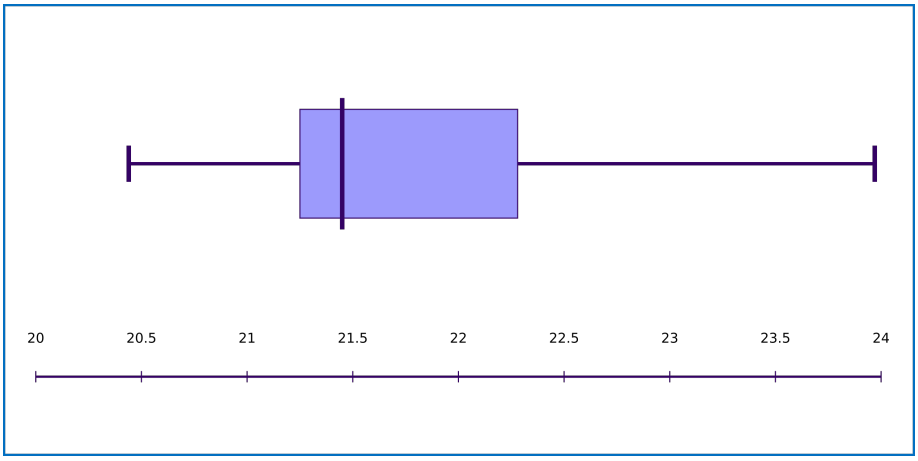

FIGURE 14. Box-and-whisker diagram for the data collected from Bristol Cathedral. Variation due to hand tremors alone would be normally distributed; thus the presence of a positive skew in these data indicates that another source of variation is also present

5. Integration

5.1. RIOJA. With some prompting from Dr Bernhard, and having developed RIO-A and J117 independently, as above, I then took the final significant step in the development of my project, and integrated the two systems into one. This was surprisingly straightforward; indeed, Python's pySerial package does a lot of the heavy lifting in terms of getting Python to talk to an Arduino.

As part of the process of integration, I've written a Python class which (1) given a time, a latitude and a longitude, can calculate the angle of elevation of the sun, and (2) can extract a given image's EXIF data from a database. Thus, for the various runs of thirty radar data images discussed above, RIOJA is able to calculate whether the GPS position is verified or falsified by J117, based on the time and position in that image's EXIF data.

5.2. Integration Testing. RIOJA works as one would expect: the two halves, RIO-A and J117, are deliberately kept as separate as possible, and each functions properly alongside the other without any interference. When running RIO-A on the standard run of thirty images from within RIOJA, one gets the same results as when running RIO-A as a stand-alone program.

On 03 Sep 2018, at my parents' home in South Yorkshire, I tested J117 from within RIOJA. Wearing the appropriate eye-wear, I observed the sun at the altitude at which my program predicted it would appear, to within a margin of error roughly equivalent to that recorded in §3 of this chapter.

CHAPTER 4

Conclusions

1. Known Faults

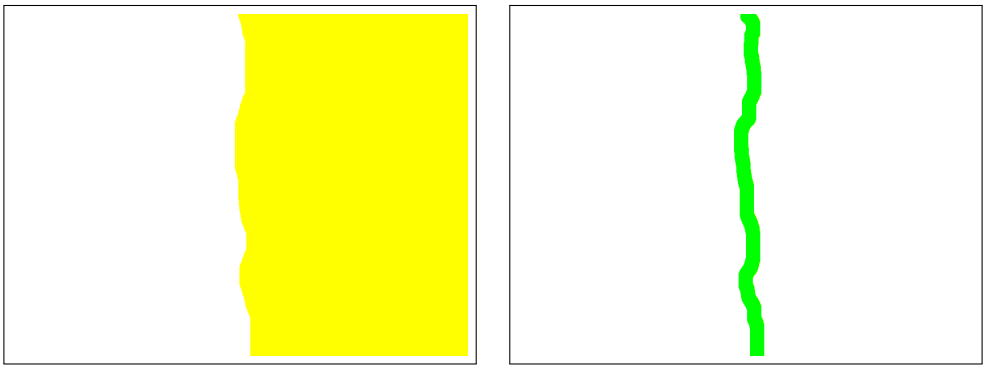

FIGURE 1. The chart (left) and radar (right) displays when a parallelism is occurring. In this case GPS might be offset by, say, 0.1NM due north, i.e. GPS is putting us slightly to the north of where we really are

1.1. Parallelism.

1.1.1. *Description.* A *parallelism* – as I use the term in this context – occurs when there is a GPS error which is parallel to the direction of the coast line. RIO-A cannot detect an error because the two outlines – the shape of the coast from the point of the current GPS fix, and the shape of the coast from the point of view of the vessel's true location – are almost identical.

1.1.2. *Seriousness.* Danger, from the point of view of the navigator, is synonymous with land and shallow water. With that in mind, for the time being I judge that this error is only of **minor** significance, for the following reasons:

- The user is not given a false sense of security; the distance and direction of the nearest point of danger is still correct.
- A parallelism can only occur when certain factors come together: a smooth, regular coast line, with no outlying islands or buoys, and a GPS error which is parallel to that coast line. This means that, although parallelisms are possible, they would nonetheless be very unusual.

Note, however, that the above conclusions assume that any dangers to navigation would correspond to, or would be located close by, features on the radar display. While I believe this assumption is justified – for obvious reasons, isolated rocks and shallow patches are almost always accompanied by lighthouses, buoys, lightships, etc – this is another factor which ought to be taken into account if RIO-A was ever put into practice.

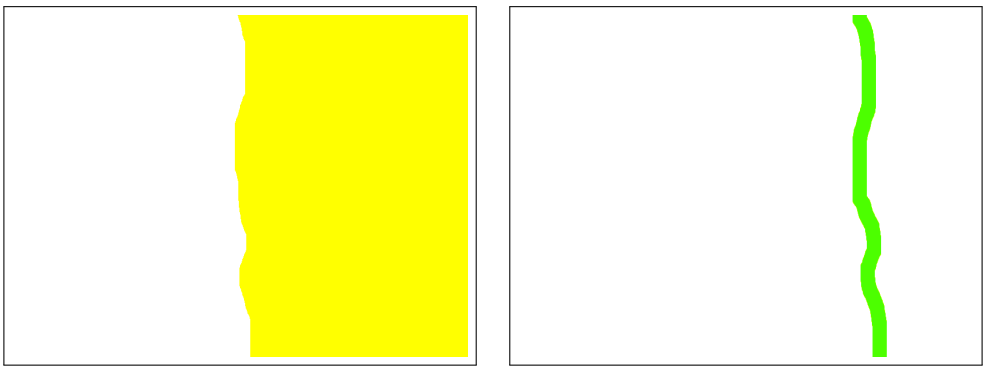

FIGURE 2. The chart (left) and radar (right) displays when a transversism is occurring. In this case GPS might be offset by, say, 0.1NM due east, i.e. GPS is putting us slightly to the east of where we really are

1.2. Transversism.

1.2.1. *Description.* A *transversism* – as I use the term in this context – occurs when there is a GPS error which is roughly at right angles to the direction of the coast line, and is towards, and not away from, the land. RIO-A cannot detect an error because the green pixels, while being misaligned with the coast line, are still over the land, and thus are not reported as being unaccounted for.

1.2.2. *Seriousness.* For the time being I judge that this error is of **moderate** significance, for the following reasons:

- The user is never given a false sense of security; danger is always reported as being *closer* than it really is, and not further away.
- On the other hand, if a crude version of RIO-A were used at sea, transversisms would have more than a realistic chance of occurring; thus it is possible that the user might be left with an undetected GPS fault, albeit a less dangerous one, for some time.

1.2.3. *Mitigation.* The principal strategy for mitigating this fault, at least from a programmer's point of view, must be a secondary means of identifying a GPS fault, complementary to the primary means of counting unaccounted for green pixels. The most simple form of such a secondary means that I can imagine would involve comparing the distance to land on the chart with the same distance on the radar feed. Indeed, in §3.3.1 of the Execution chapter, I've implemented just such a system, with encouraging results. Distance to land checks come with their own issues and caveats, but none, to my knowledge, are insurmountable. If I had more time to work on this project, this would be one of the first areas I would investigate further.

Other secondary means of identifying GPS faults include using machine learning techniques to draw out the coastline from the radar data; this reconstructed coastline can then be compared to the coastline as it appears on the chart. I have to admit that, with only one year of programming under my belt, such techniques are beyond me at present. May others come and succeed where I have failed.

1.3. Stationary Vessels. The radar systems fitted on large ships are generally capable of distinguishing between fixed and moving objects. So vessels

underway ought not to present any problem. However, radar itself has no way of distinguishing between *stationary* vessels and land. Two or three small craft will not make much of difference to RIO-A as I have designed it; but this presents a serious problem when navigating in or near busy harbours. Indeed, as the reader can see in the image below, I came across precisely this problem when collecting my radar data. At the time of writing, I can't think of any straightforward solution to this issue. All I can say in mitigation is that ships generally have a large number of personnel on the bridge, including a local pilot, and tend to navigate by traditional means when passing through such waters.

FIGURE 3. An image from the radar data I collected, showing a large concentration of stationary vessels. In this case, the concentration was significant enough to cause a false positive when looking for a GPS fault

1.4. Radar Faults. In my professional experience, radar has always been one of the more reliable items of bridge equipment. Nevertheless radar is not infallible, and is prone to a number of known faults. Of these, several are significant from a navigational point of view:

- Technical issues with the radar system itself include double echoes, side lobe error, attenuation error, etc. See the documents listed in the radar section of my Literature Survey for a detailed discussion of these.
- Larger waves, or, if the ship in question is particularly large, structures on the deck, can be mistaken for land. This issue is exacerbated by a rough sea state and poor radar tuning.
- With certain wavelengths, rainfall can cause large patches of sea to appear as land on a radar display. The commonly used X-band 3cm system seems to be particularly vulnerable to this issue.
- I have not witnessed personally, but have listened to several firsthand accounts concerning, snowfall and sandstorms having a similar effect to rainfall on certain radar wavelengths.

Note also that, in order to determine the true direction from which returning signals are coming, a ship's radar usually has a feed from the gyrocompass. Thus any compass error is likely to cause a corresponding error on the radar display.

2. Evaluation

2.1. RIO-A. *See the tables in §3.2 of the Execution chapter for the evidence behind the claims made here.*

The results for RIO-A have been encouraging, and I have a system here which essentially works, although a few more weeks of development would be needed before a further round of testing at sea. The system is clearly able to detect large GPS faults – in the order of several hundred metres – when navigating close to land. With regard to smaller faults: the system begins to suffer from false negatives once the GPS fault is reduced to below 100m, and seems, roughly speaking, to suffer from false negatives about 50% of the time once said fault is reduced to 50m.

To put these figures into context: the width of a large ship is around 30m. In principal, the location *within the bridge* from which one takes a fix ought not to make a difference; thus 30m is often treated as the theoretical limit of accuracy for maritime navigation. Also, RIO-A is intended to run comparisons every few *seconds*. In that context, several *consecutive* false negatives would of course be a cause for grave concern, but several *intermittent* such errors – as is the case here – would not be a significant problem: the user will still be alerted that there is a GPS fault in a prompt fashion. With all that in mind, I am very much satisfied that RIO-A has a false negative rate of around 50% with a GPS fault of 50m, and a false negative rate which rapidly approaches zero once the GPS fault is raised to a figure greater than 100m.

RIO-A does suffer from a number of known faults, both tangible and theoretical. Indeed, we can observe a mild case of parallelism (see §1.1 of this chapter for an explanation of this term) in the data I've collected. This occurs because of the north-south direction of the estuary I was on when the data was collected; when a GPS fault of 110m north is introduced, RIO-A exhibits a disappointing and uncharacteristic false negative rate of 50%. RIO-A might also be vulnerable to transversisms (see §1.2 of this chapter), although distance to land checks (§3.3.1 of the the Execution chapter) have shown the potential to remove most of this risk. Stationary vessels present another serious known fault (see §1.3 of this chapter), and consistently caused two false positives to crop up in my data.

The *speed* of RIO-A is also a matter of some significance. Ideally, we would like one comparison to be performed in, at most, two or three seconds, the time it takes a radar transmitter to rotate once. At present, RIO-A takes around five seconds. However, my dear friend Dr Steven Charlton, currently of the Max Planck institute in Germany, has put together a version of RIO-A in C, and which runs a comparison in around a second. Thus we can conclude that, were RIO-A to be developed further, speed would not be a limiting factor.

2.2. J117. *See §4 of the Execution chapter for the evidence behind the claims made here.*

The results for J117 have also been encouraging, but I don't yet have a system which would be worth testing at sea. At present, J117 is capable of measuring elevations with an accuracy of about half a degree. This might sound impressive to a layman, but, within the context of celestial navigation, it simply isn't good enough. When plotting a celestial fix, an error of one degree in the height of the body observed corresponds to an error of sixty nautical miles on the chart; so

any kind of position derived from J117 would only be accurate to thirty miles, an almost comically large figure.

This low level of accuracy is largely due to the cost of components: the Pmod NAV, J117's sensitive component, cost the university about £20 to purchase for me. If I had been able to get hold of, for instance, the MTI-10-2A5G4 [13], then I might well have been able to turn out a much more accurate device, but this would have cost around £600. There is also room for improvement with the steadiness with which the device is held. The device is currently shaped like a pistol; it might be worth redesigning it into a more rifle-like format, which would improve steadiness; although it's worth noting that modern sextants, J117's nearest competitor, are all of a more pistol-like shape.

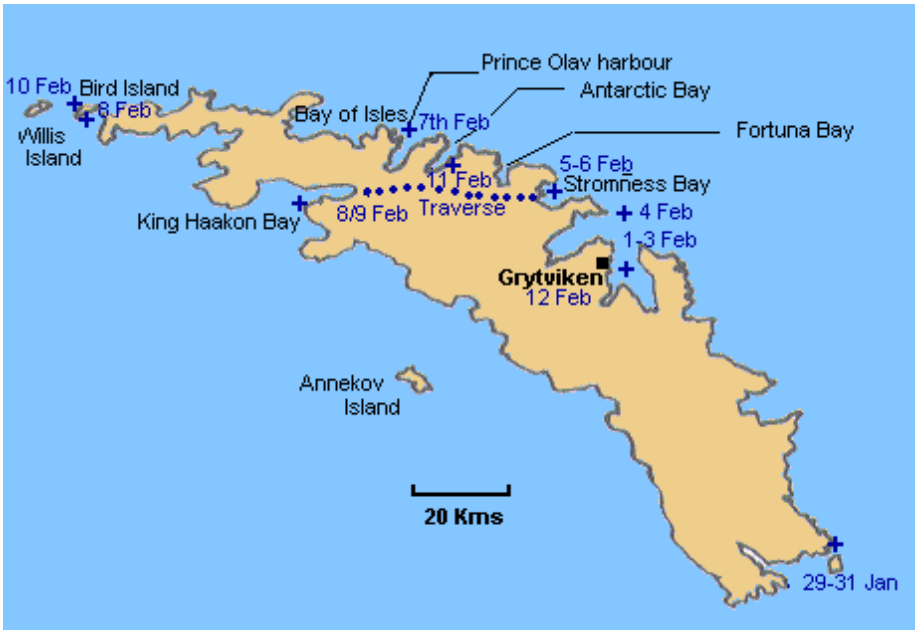

FIGURE 4. South Georgia. While Shackleton had intended to land near Stromness, he in fact landed at King Haakon Bay, on the wrong side of the island

As disappointing as the current performance of J117 may seem, it has to be put into the context of what any other system of celestial navigation is capable of achieving at present. Traditional sextants are scarcely any better. While I distinctly remember being taught, during my training as a navigator, that celestial fixes, as performed by an experienced officer, are accurate to within two or three nautical miles, this, frankly, is a pious fraud. For the proof of my point of view, we can look at one of the most famous journeys conducted using celestial navigation alone, the voyage of the *James Caird*.

On the eve of the First World War, Ernest Shackleton and his crew set out for the so-called Imperial Trans-Antartctic Expedition, but his ship, the *Endurance*, was trapped and ultimately crushed by pack ice in the Weddell Sea. The explorer and his men eventually relocated to Elephant Island, a blob of land shaped like an elephant's head a hundred odd miles off the Antarctic mainland. From Elephant Island, Shackleton took five of his strongest men, and, on the *James Caird*, the

most robust of *Endurance*'s lifeboats, sailed the eight hundred miles to South Georgia.

The fact that Shackleton and his navigator Frank Worsley managed to find South Georgia at all in the vastness of the Southern Ocean is pretty astonishing, and I don't want to say anything to denigrate that achievement. However, the voyage of the *James Caird* does still demonstrate the shortcomings of celestial navigation rather neatly. Granted, Shackleton and his men were sailing on a rickety little boat through the world's roughest ocean, but they had all amenities of present-day celestial navigation: charts, a sextant, nautical tables and an almanac. They were also some of the best navigators in the world at that time, and had a very strong motivation – their very lives were at stake – to navigate to the best of their abilities. But the *James Caird* still landed in King Haakon Bay, completely the wrong side of the island, and not near Stromness as intended. Given that South Georgia is almost a hundred miles long, this must have involved some large navigational errors. Indeed, Frank Worsley himself described the voyage's navigation as 'a merry jest of guesswork' [44].

2.3. RIOJA. RIOJA is an integration of RIO-A and J117 into one system. At present, J117 is of negligible practical use; so the practical efficacy of RIOJA is the same as that of RIO-A.

3. Further Development

I'm pleased with what I've been able to achieve in the last four months. Nevertheless, the work I've done so far cries out for further development, and there are specific steps I would have taken given more time, more money and access to certain resources.

Given a few more weeks:

- I would have made a graphical interface for RIOJA, making its calculations much more user-friendly and transparent;
- I would have developed a more sophisticated system of distance to land checks.

Given more money, I would have bought a more accurate accelerometer for J117. More money might also have bought me access to the other resources necessary to develop this project to its natural conclusion, namely:

- Access to the vector graphics files which constitute Electronic Navigational Charts;
- Access to the raw data which a radar transmits to an electronic chart;
- Access to the source code behind a modern Electronic Chart Display and Information System;
- Access to the source code behind a modern Automatic Radar Plotting Aid, particularly as concerns its ability to distinguish between moving and stationary objects;
- Access to the bridge of a large ship navigating a few miles off an interesting (hydrographically speaking) coastline.

The measures I've just highlighted would be expensive, would require the help of experts in at least two different fields, and would require at least several months, more likely several years, to exploit properly. If all this seems like a

trouble for one small avenue of research, bear in mind that success here would significantly improve the safety and efficiency of one of the most vital – and lucrative – activities on the planet. And, as per the famous saying: all excellent things are as difficult as they are rare.

Glossary

Altitude: In this context, the angle measuring the height above the horizon of a heavenly body.

Attitude: In this context, the angle between a line, in three dimensional space, and the horizontal plane.

Azimuth: In this context, the angle between a line and north, measured clockwise.

Chart: The nautical analogue of a map.

COLREGS: *International Regulation for the Prevention of Collisions at Sea.*

DGPS: Differential GPS. A method of verifying and improving the accuracy of GPS using ground stations.

ECDIS: Electronic Chart Display and Information System. The nautical analogue of a car's satellite navigation system.

ENC: Electronic Navigational Chart. These are, essentially, the files which an ECDIS will display – although there's a bit more to it than that.

Glonass: A satellite navigation system similar to GPS backed by the Russian Federation.

Hydrography: The nautical analogue of cartography, or map-making.

IMO: International Maritme Organisation. An agency of the United Nations.

Officer of the watch: The leader of a given watch on a sea-going vessel. An officer of the watch might be thought of as equivalent to the driver of a car or the pilot of an aircraft – although there's a bit more to it than that.

Radar Integrated Overlay: A facility in various ECDIS systems which overlays the radar feed onto the chart, and thereby allows the officer of the watch to verify GPS.

RIO: Radar Integrated Overlay.

SOLAS: *International Convention for the Safety of Life at Sea.*

STCW 95: *International Convention on Standards of Training, Certification and Watchkeeping for Seafarers.*

S-Band: A form of radar with a wavelength of 10cm.

Transas: An example of a commercial ECDIS.

WECDIS: Warship ECDIS.

X-Band: A form of radar with a wavelength of 3cm.

Bibliography

[1] A P Møller-Mærsk A/S. "DGPS Fundamentals". In: *Position Referencing System Training Manual* (2015).

[2] A P Møller-Mærsk A/S. "GPS Fundamentals". In: *Position Referencing System Training Manual* (2015).

[3] Arduino. *Arduino Uno Rev3*. https://store.arduino.cc/usa/arduino-uno-rev3. Accessed: 14 Aug 2018.

[4] Sean Barrett. *stb*. https://github.com/nothings/stb. Accessed: 15 Aug 2018.

[5] Aaron Burg et al. *MEMS Gyroscopes and Their Applications*. http://clifton.mech.northwestern.edu/~me381/project/done/Gyroscope.pdf. Accessed: 24 Apr 2018.

[6] Alex Clark. *Pillow*. https://pillow.readthedocs.io/en/latest. Accessed: 15 Aug 2018.

[7] A N Cockcroft and J N F Lameijer. *A Guide to the Collision Avoidance Rules*. 2003.

[8] William Cole. *Hawaii-based ship's grounding detailed*. http://the.honoluluadvertiser.com/article/2009/Jul/07/ln/hawaii907070350.html. Accessed: 17 Aug 2018. 2009.

[9] Data Flair. *Java Image Processing – A Complete Guide*. https://data-flair.training/blogs/java-image-processing/. Accessed: 15 Aug 2018.

[10] Dimension Engineering LLC. *A beginner's guide to accelerometers*. https://www.dimensionengineering.com/info/accelerometers. Accessed: 24 Apr 2018.

[11] Dimension Engineering LLC. *Buffered ±6g Accelerometer*. https://www.dimensionengineering.com/products/de-accm6g. Accessed: 24 Apr 2018.

[12] Evil Mad Scientist. *Diavolino*. https://shop.evilmadscientist.com/productsmenu/tinykitlist/180. Accessed: 14 Aug 2018.

[13] Farnell. *MTI-10-2A5G4*. http://www.farnell.com/datasheets/1859197.pdf?_ga=2.141895226.922732234.1536077925-539079088.1531139737&_gac=1.35785364.1536077925.EAIaIQobChMItqK32cLJ3AIVFPhRCh37jwJ5EAAYASAAEgIbGvD_BwE. Accessed: 01 Sep 2018.

[14] Tiago Ferreira and Wayne Rasband. *ImageJ User Guide*. https://imagej.nih.gov/ij/docs/guide/user-guide.pdf. Accessed: 15 Aug 2018.

[15] Freeduino.org. *Freeduino Open Designs*. http://www.freeduino.org/freeduino_open_designs.html. Accessed: 14 Aug 2018.

[16] Vladimir Glotov. *GLObal NAvigation Satellite System (GLONASS)*. https://web.archive.org/web/20161016065341/http://147.102.106.44/ILRS_W2009/docs/PP02B_GLOTOV_GLONASS.pdf. Accessed: 19 Apr 2018.

[17] Google LLC. *Android Motion Sensors*. https://developer.android.com/guide/topics/sensors/sensors_motion.html. Accessed: 24 Apr 2018.

[18] Google LLC. *Sky Map*. https://play.google.com/store/apps/details?id=com.google.android.stardroid&hl=en_GB. Accessed: 24 Apr 2018.

[19] Gps Nautical Charts. *Bristol Channel (UKHO Chart 1179)*. http://www.gpsnauticalcharts.com/main/uk/1179_0-bristol-channel-nautical-chart.html. Accessed: 21 Apr 2018.

[20] Larry Greenmeier. *GPS and the World's First 'Space War'*. https://www.scientificamerican.com/article/gps-and-the-world-s-first-space-war/. Accessed: 17 Aug 2018.

[21] Tom Hosker. *Navigational Workbook*. 2016.

[22] International Chamber of Shipping. *Bridge Procedures Guide*. 5th ed. 2016.

[23] International Maritime Organisation. *International Convention for the Safety of Life at Sea*. 2016.

[24] International Maritime Organisation. *International Convention on Standards of Training, Certification and Watchkeeping*. 2010.

[25] International Maritime Organisation. *International Regulations for Preventing Collisions at Sea*. 2016.

[26] Jacek Januszewski. "Global Satellite Navigation Systems at High Latitudes, Visibility and Geometry". In: *Annual of Navigation* (2016).

[27] David Jenn. "Radar Fundamentals". In: *Naval Postgraduate School Seminars* (2007).

[28] Pratap Misra and Per Enge. *Global Positioning System: Signals, Measurements, and Performance*. 2010.

[29] Mission to Seafarers. *Seafarers Happiness Index*. https://www.missiontoseafarers.org/seafarershappiness. Accessed: 12 Aug 2018.

[30] Alexander Mordvintsev. *Getting Started with Images*. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html. Accessed: 15 Aug 2018.

[31] National Geospatial-Intelligence Agency. *Radar Navigation and Maneuvering Board Manual*. 2001.

[32] National Transportation Safety Board. *Grounding of the Panamanian Passenger Ship* Royal Majesty. https://www.ntsb.gov/investigations/AccidentReports/Reports/MAR9701.pdf. Accessed: 17 Aug 2018. 1997.

[33] David Nield. *All the Sensors in Your Smartphone, and How They Work*. https://fieldguide.gizmodo.com/all-the-sensors-in-your-smartphone-and-how-they-work-1797121002. Accessed: 24 Apr 2018.

[34] Dwayne Phillips. *Image Processing in C*. http://homepages.inf.ed.ac.uk/rbf/BOOKS/PHILLIPS/cips2ed.pdf. Accessed: 15 Aug 2018.

[35] Raspberry Pi Foundation. *Raspberry Pi 3 Model B*. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Accessed: 14 Aug 2018.

[36] Raymarine. *Marine Radar Scanners*. http://www.raymarine.com/marine-radar/. Accessed: 20 Aug 2018.

[37] SparkFun. *ADXL345 Hookup Guide*. https://learn.sparkfun.com/tutorials/adxl345-hookup-guide. Accessed: 14 Aug 2018.

[38] Transas Ltd. *NAVI-SAILOR 4000/4100 ECDIS (Version 2.00.009) User Manual*. 2009.
[39] United Kingdom Hydrographic Office. *Admiralty Guide to ECDIS Implementation, Policy and Procedures*. 2014.
[40] United Kingdom Hydrographic Office. *Admiralty Guide to the Practical Use of ENCs*. 2013.
[41] United Kingdom Hydrographic Office. *NavPac and Compact Data 2016-2020*. 2015.
[42] United Kingdom Hydrographic Office. *The Mariner's Handbook*. 2016.
[43] United Kingdom Hydrographic Office. *The Nautical Almanac 2018*. 2017.
[44] Frank Worsley. *Shackleton's Boat Journey*. 1933.