Using GPS and Accelerometer Data for Rowing Race Tracking

Harm Hermsen

October 2013

Master Thesis

Artificial Intelligence

Dept of Artificial Intelligence,

University of Groningen, The Netherlands

First supervisor:
Dr. Marco Wiering (Artificial Intelligence, University of Groningen)

Second supervisor:
Prof. Marco Aiello (Distributed Systems, University of Groningen)

university of groningen / faculty of mathematics and natural sciences / artificial intelligence

Abstract

Rowing is one of the oldest Olympic Sports and still internationally popular, especially among students. Races are often held side by side on a track over a distance of 2000 meters in six lanes, where spectators are positioned near the finish line. Due to the length of the track only the last 30 seconds of the race can be physically seen and the preceding battle is lost. In this thesis a solution is presented which uses small, cheap and easily available commercial electronic components to be able to present a live tracker of the full race to the audience along the track and at home. The solution consist of an onboard device with GPS and accelerometer sensors, a microcontroller and a ZigBee communication module, and a land based receiver which interacts with the data processing and visualization software. Communication over 2000 meters distance is achieved using mesh networking, which allows for roving nodes. To minimize network traffic and save on battery capacity, a computationally cheap algorithm is presented which can be executed on the onboard microprocessor and applies knowledge about the anatomy of the rowing stroke to obtain orientation invariant accelerometer measurements. Because of this the onboard device can be placed on the boat regardless of the orientation, making the solution as unobtrusive to the athletes as possible. Two types of peak detection methods are compared to perform stroke detection in real-time on the microcontroller. Furthermore sensor fusion methods are examined to improve GPS location accuracy by incorporating data from a static GPS receiver and the onboard accelerometer. It is shown that an onboard device can be made for under € 100,- and can report location, speed and stroke rate in real-time.

Contents

1 Introduction

Rowing is one of the oldest Olympic Sports and still internationally popular. The combination of synchronism, wonderful scenery and fierce competition makes it a good sport to watch. Races usually take place side by side on a canal or lake over a distance of 2000 meters in six lanes, although head races are also held from fall to early spring. During the Olympic Games and World Cups, the spectators that are positioned near the finish-line, are provided with live video footage of the race from start to finish. Because of the length of the track, it is impossible to watch the full race by eye and from the grandstand only the last 250 meters of these races can be seen physically. During the somewhat smaller national regattas, live video coverage is out of budget. Spectators then only see the last 30 seconds of the race and are only informed about what happened up to the final battle by intermediate timing results from the 500 meter, 1000 meter and 1500 meter marks.

This thesis discusses the usage of easily available consumer sensors and electronics for live visualization of rowing matches to home and live spectators. This project is set up as part of the service expansion program of Time-Team Regatta Systems. Time-Team is a company that provides regatta management and timing services for all Dutch national regattas and some smaller competitive rowing regattas which do not comply with the strict KNRB regulations of the national regattas [1]. Time-Team currently provides intermediate (500m, 1000m, 1500m) and finish time results but wants to extend the information provision to include live race tracking. This way, spectators can be provided with more race information and therefore it would be more attractive to watch the races (either live from the finish-line or on the Internet). Live race tracking as such could include additional data, like speed and stroke rates, besides the necessary distance-travelled information.

The objective is to achieve live race tracking using small, cheap and easy to acquire electronic parts. The price of the onboard devices should be kept as low as possible since 80–100 of these devices need to be at hand to provide all boats that are concurrently on the water with live tracking. The monitoring equipment should be essentially unobtrusive such that the athletes are virtually unaware of it being present. Only this way they can fully focus on their effort for the race. A GPS sensor is used for the absolute localization and an accelerometer is used for stroke detection and potentially improving distance-travelled data when fusing the information with the GPS. The data gathered by GPS and accelerometer need to be processed by an onboard microcontroller and wireless sent to the shore. At the central data collector an algorithm determines the progression of the race (the distance to the start position of the boat) and shows the race details on an interface. The information contains details about the progression of the full race including speed and strokes per minute. Additionally, the speed information could also be used to show boats catching up on each other or to augment real time video footage.

Since a live rowing tracker has to deal with communication, knowledge extraction from the available data and in this case also data fusion, the following research questions are formulated, which each attempts to answer a specific part of the problem. By combining these individual solutions a full race tracker could be achieved.

1. How can data be reliably transferred by roving nodes over a distance of two kilometers using easy available and cheap ZigBee radio modules? What is the performance of many-to-one mesh networking using roving XBees?

2. Which algorithm is simple enough to run on easy available and cheap hardware and is still reliable in the detection of rowing strokes in accelerometer data? Can this algorithm be extended in a way that accelerometer orientation will not influence the algorithm's outcome?

3. How can the relative distance from the starting position be calculated from GPS recordings? What is the accuracy of this data? What are the improvements when using a cheap static reference receiver to reduce omnipresent noise?

4. How can the accuracy of the distance-travelled data be improved by fusing GPS and accelerometer data streams?

The first research question stipulates the problem with the length of the track and the necessary limited size of the onboard devices and thereby the communication module. While wireless communication ranges of over two kilometers can be obtained using large antennas, this is not an option for the live tracking devices because of the size requirements. The communication modules need to be cheap and easily available but still have a high reliability for its transmissions over this long range. The ZigBee standard looks promising since it makes use of mesh networking techniques: a network in which longer distances can be overcome by passing data through intermediate devices. Therefore there is no need for a high-power module which could reach all of the devices by itself. The fact that the boats are moving, needs some further attention since this could influence the way the mesh networking protocol works and establishes its data routes. The XBee is an easily available specific ZigBee implementation for a relative low price. It has the capabilities of using many-to-one mesh routing for wireless sensor networks in which there is a data collector. It needs to be assessed if these modules perform well and reliably in rowing conditions. First the range of a normal routing network has to be obtained as a base line measurement. Then movement changes need to be made to one of the nodes in order to test the ability of re-establishing routes to the data collector. It needs to be examined if packets will be lost during movement and what the distance ranges are. Tests could be done similar to [2] in which the performance of ZigBee PRO modules is tested for point-to-point and multi-hop transmissions.

The stroke rate shows the number of strokes executed per minute by a crew. This is an important measurement in rowing since it gives information about the speed a crew is rowing, but also if the crew is accelerating, decelerating or keeping a similar speed. Since it would be valuable to show the stroke rate in a live tracking application, it is investigated if this number could be detected from the acceleration of the boat. During the rowing stroke specific body movements are made by all the crew members to drive the boat forward. These movements result in a characteristic pattern of acceleration. This pattern can be used to detect the rowing stroke in the accelerometer data. Specifically, peaks can be detected in the acceleration which correlate to a certain part of the rowing stroke, by which the stroke rate can be calculated. In [3] a number of simple peak detection algorithms for time-series are described. Although these algorithms are simple enough to be programmed onto cheap microprocessors, they work on full data sets and not in online detection. Therefore a peak detection algorithm should be designed which performs well on rowing accelerometer data with the limited processing resources in mind.

To take the stroke detection a step further, it could be investigated if it is possible to design a simple algorithm which preprocesses the accelerometer data in such a way that orientation of the onboard device is not affecting the filter's output. Artificial intelligence (hereafter AI) feature extraction algorithms can be tested for the extraction of orientation of the sensor. Using the extracted orientation, the data needs to be rotated and simplified to acquire the moving axis with only accelerometer data, which then could be fed into the peak detection algorithm. However since most AI feature extraction methods use vast amounts of data and perform many calculations, these methods might not be suitable out-of-the-box for the application on a microprocessor. The possibilities of these algorithms are explored and an algorithm should be designed specifically for the use in rowing with limited computational power.

To answer research question three, the knowledge about the curvature of the Earth needs to be taken into account. Post-processing algorithms need to be found for relative distance calculation between recorded latitude and longitude points. There are two well known formulas that are both used in navigation to calculate the distance between two points given their latitudes and longitudes [4]. The Haversine formula calculates the distance assuming a spherical Earth and its counterpart, Vincenty's indirect formula, has the assumption that the Earth is an oblate spheroid, which more closely matches the Earth's real shape. The implementation and pros and cons of both algorithms need to be investigated. A comparison can be made between conventional timing results and timing results acquired by using GPS data and the calculated distance, for recorded two kilometer races.

As a data fusion improvement, measurements could be made with an additional GPS receiver at a static point. Since the GPS error is influenced by several atmospheric and satellite specific effects, the measured error on a static point can be used to correct the measurements of a moving GPS receiver. Measurements over a certain long time interval from this static point can be used to get the averaged actual latitude and longitude of the receiver. Later, the offsets of the static receiver can be applied in post-processing mode to the data of the roving receiver, achieving a system which is similar to differential GPS (hereafter D-GPS) [5]. The accuracy of this data fusion technique has to be assessed to determine how much improvement this D-GPS like technique entails.

Finally a fusion algorithm needs to be found which integrates the GPS and accelerometer data streams into a single more accurate distance-travelled measurement. A Kalman filter [6] is proposed for estimating the current boat's state using information of both sensors. The accelerometer data can be used to estimate the current speed of the boat, and by integration also the travelled distance. However, the noise on this data would yield a growing error over time, since only relative measurements are made. The GPS data can be used to correct the estimation since it makes absolute measurements for both speed and position. Due to the complementary error behaviour of the sensors, the estimated state should be more accurate. To test the improvement over GPS-only results, again a comparison can be made between conventional timing results and timing results acquired by using the Kalman filter state estimate, for recorded two kilometer races.

This thesis starts by giving an introduction into the background and related work in chapter 2. In chapter 3 the assessment of the different hardware components is done while also some functional descriptions and theories behind these components are given. The way raw accelerometer data is transformed into orientation invariant measurements and how strokes are detected from these measurements is explained in chapter 4. Chapter 5 elaborates on the boat's position tracking and proposes two different sensor fusion methods to obtain higher accuracy. Finally, in chapter 6 a discussion is presented on the work done in this thesis and some future work is suggested.

2 Background and Related Work

This chapter gives background information and an overview of related work on specific parts of the use of sensors, communication details and rowing live tracking. First in section 2.1 a wide view is given on the application of different tracking techniques in different types of sports. Then in section 2.2 previous work on GPS localization, tracking and sensor fusion is discussed. Section 2.3 elaborates on previous work in stroke rate detection. The work on communication networks with moving nodes is discussed in section 2.4 and currently available commercial products are presented in section 2.5.

2.1 Background

During the last years there has been a growing interest in automatic systems for detecting, tracking and identifying participants in different types of sports. The goals of these solutions differ: some of them are used for improving the participant's performance, some of them are used for obtaining metrics and analysis for coaches, some are used for timing races and others are used for statistical match analysis. The main two categories of technologies used for automatic detection and tracking are intrusive solutions and non-intrusive solutions. With intrusive solutions, special sensors or tags need to be placed on the sportsman. In contrary with non-intrusive solutions, no additional objects are needed in the sport environment.

During indoor sports, non-intrusive systems can often be used. Using a single or several normal or infrared cameras, the field of play can be recorded and analyzed using image processing techniques. The advantage of these types of systems is that they are completely unobtrusive to the sportsmen. Systems like these are used for the tracking of football players on the field, speed of a tennis ball served, basketball statistics and the number of rounds skated by ice skaters. Since many indoor sports only require a limited space, this technique can easily be applied. For outdoor sports that generally cover larger areas, other detection and tracking systems are used. Most of the outdoor detection and tracking systems make use of an intrusive sensor or tag. The sensor or tag has to be placed on the sportsman or on the vehicle used, and therefore the sportsman is aware of the tracking solution being present and sometimes the sportsman is even obstructed a little by the tracking solution.

The intrusive systems can be divided into two categories: passive tags and active sensors. Passive tags are generally cheap and can be used when there is the possibility to place a gate with sensors, through which all competing participants have to go. RFID is used for the detection and when tracking needs to happen at multiple points along the track, multiple sensor-gates are placed. This type of participant detection is often used in cycling and running. The accuracy of these solutions is very high: RFID sensors have a limited detection range and when a tag is detected, it is sure that the tag was within range of the sensor and therefore its location is known. Active sensors make use of battery powered components to transmit data to a main processing unit. In outdoor usage these sensors often contain a GPS for positional data. Applications can again be seen in cycling (for constant tracking) but also in skiing and sailing. The accuracy of GPS is not very high, so in general this data can not be used for sports timing. It can however be used for visualization and possible data logging purposes.

Since recently people also started using their smartphones for training and tracking sports. Most smartphones contain a GPS sensor and an accelerometer and are therefore well suited for simple tracking. Various apps are available to assist training and compete in running and cycling. Although the usage of smartphones for personal use can be beneficial, it has not yet been applied for large sport events.

2.2 Localization & Sensor Fusion

Exact localization is the most important criteria which a rowing live tracker should satisfy. The timing resolution used in finish photos is one hundreds of a second, although rowing umpires normally adhere the bow ball rule: a small, soft ball no smaller than 4 cm diameter securely attached to a boat's bow can be used to distinguish the winner [1]. At a top speed of 6.3 m/s (world record) this would result in a resolution of about 6 milliseconds.

The travelled distance from the start needs to be extracted from GPS data acquired from sensors on the boats. This can be done using mathematical formulas but since no constraints are placed on the placement of the sensors on the boat, an algorithm needs to be designed to cope with this differentiation in placement on the boats. A mere distance calculation algorithm will not suffice, but start-timing data have to be incorporated into the calculation to compare data against the detected location during the start. It might even be possible to do false-start detection using this data.

Non augmented GPS data can reach a absolute point positioning resolution of 6.8 meters [7]. This error is too large for a proper rowing race tracking system and therefore it needs to be investigated if the accuracy could be improved by combining location and speed data from a GPS sensor with data from an accelerometer. Double integration over time for the acceleration data yields the travelled distance [8]. However the data are suspect to noise and the accuracy deteriorates with time due to the integration. This drift accumulates and for long duration tracking this solution would prove inaccurate. Nevertheless the accelerometer data could be used for short term distance measurements and could possibly be used in excluding high offset GPS data. In this thesis it is examined if the fusion of data streams will result in more accurate location measurements.

The position tracking problem can be dealt with by using the initially known location (the start) and an algorithm to account for the travelled pathway to determine the new location. Local data can in this case be used to determine the boat's location. Since boats never move backward during a race, this a-priori knowledge could be used as input for the tracking task, since erroneous readings can be excluded if they would imply a backward movement. A Kalman Filter can be used to solve the tracking problem [6]. Such a filter takes Gaussian noise assumptions and propagates the belief about position and speed through time and incorporates information from measurements. The belief is a probability density over all possible locations. The location that has the highest probability is the location at which the boat is most likely to be. Since the Kalman Filter can make use of noisy relative and absolute measurements, both accelerometer and GPS data can be incorporated to improve the estimated location state and thereby the distance-travelled parameter.

This combination of both relative and absolute position measurements guarantees a higher accuracy on both sides. Combining the data into a single location measurement, while including a-priori knowledge, could be beneficial for the position determination in rowing.

Rule based sensor fusion between GPS, odometer and accelerometer data has previously been used in overcoming GPS-dark areas in localization tasks in which there was a need for robust and constant localization [9]. The results show that short duration GPS outages can be overcome by using this technique.

In [10] the possibility of fault detection using sensor fusion is shown. This paper describes a method in which data of multiple similar sensors are combined and the combined output is verified against the output of the single sensors. When one of the sensors is deviating more than normal from the combined signal, a possible fault is marked.

Sensor fusion touches the foundations of artificial intelligence: how to combine data from multiple sensors into a fully contextual understanding of a situation. By fusing data of multiple low-cost and simple sensors one could achieve a performance which is higher than the sum of the single sensor performances. In this case one could achieve a system that uses sensor fusion and conveniently integrates data provided by the GPS and accelerometer to estimate the correct distance travelled by the system. This fusion is useful because of the ability to acquire increased performance using low-cost sensors.

Fault detection can robustly be done with the fusion of data from multiple sensors. In situations where artificial intelligence is applied, like autonomous robotics, it is not unlikely that some of the sensors fail sooner or later. By giving the agent the possibility of doing such fault detections one could achieve an integrated system that checks its own inner workings. When a fault is detected the agent could take actions and alert the user of the faulty sensor. In the meanwhile the agent could restrain from using that sensor's data.

2.3 Stroke Rate

The stroke rate during rowing is of significant importance. In [11] it is explained that the stroke rate has a high correlation with the boat's efficiency. In normal race situations a power 10 push, to overtake another crew, is accompanied with a higher stroke rate. This number therefore gives an indication of what a crew is up to.

The accelerometer data contains the acceleration of the boat and can therefore be analyzed to output the stroke rate to the viewers. If this analysis is performed within the onboard devices, not all sensed data have to be sent over the network. This will save energy resources since the amount of communication can be reduced. Work has been done on the detection of rowing strokes in accelerometer data [12], but this generally has been done on an offline collection of data. Offline data is easier to analyze since peaks can be found more easily, since there is a possibility of looking ahead. Online detection of strokes generally happens with specific sensors attached to the oars or seats [12], although accelerometer based detection has been done in commercial products [13] and open-source software [14]. These solutions generally are quite computationally expensive and require fast hardware to run on. The developed rowing tracking system will use off the shelf, cheap and easy to acquire hardware and therefore can not rely on high performance hardware. A simple online data analysis algorithm needs to be designed to detect strokes using the cheap hardware. This algorithm needs to be robust in the sense that a different orientation of the sensor should still yield the same results.

Pattern recognition has to be done on the accelerometer data to detect the correct strokes. As the orientation of the sensor should not matter, first feature extraction has to be performed to obtain the correct (virtual) axis on which the strokes can most easily be detected. Then an algorithm has to be designed which does online recognition of rowing strokes. The total detection algorithm therefore has to be intelligent in extracting and analyzing the accelerometer data in order to make the onboard devices as unobtrusive for the athletes as possible.

2.4 Communication

In [2] mesh networking with moving nodes is discussed. This paper gives a clear overview of the functionality and inner workings of these types of networks. Since the nodes/boats in our rowing race tracking system will generally move, the choice for a ZigBee mesh network is made since it will help in the ad-hoc generation of message paths to the data collector. This way many-to-one communication is relatively easy to setup. This setup can be seen as a Wireless Sensor Network [15] in which the boats have the data gathering sensors and the data processing collector is positioned at the finish-tower. Research on many-to-one multi-hop routing with moving nodes could not be found. Since the interest lies in the performance of ZigBee networks in rowing boats over long distances using many-to-one routing, this type of routing is researched upon.

In artificial intelligence autonomous robots or unmanned aerial vehicles sometimes have the task of roving around and gathering data. Using the proposed many-to-one mesh networking setup one could retrieve the data from these robots to a central database without spending significant amounts of money and power on transceivers. Especially due to the mesh properties of the network it is easy to gather data even from very big swarms. As long as every single node is in range of any other node all data can be gathered. This way one could even choose to equip one 'leader' of the swarm with bigger range communication equipment like a cellular antenna and make him the data collector which relays data to a base station.

2.5 Commercial Products

The commercial SpeedCoach GPS presented by Nielsen Kellerman is an in-boat performance measurement device that displays and logs distance & speed using a 5 Hz GPS receiver and stroke rates using an accelerometer. In-home developed algorithms are used to detect stroke rate and speed. The device is sold for $ 399 and does not seem to be using any fusion techniques on the sensor data.

Swiss Timing is another commercial service provider that live-tracks rowing races with GPS devices. They are detecting speed and stroke rates, but often these measurements are off and do not line up with the intermediate timing measurements. The sensors and algorithms used in these devices are not known, since this information is kept confidential. They manage to track multiple races after each other but only show updates of the measurements approximately every 25-50 meters or 10 seconds, which makes the tracking feel sluggish. Presumably averaging algorithms are used to get higher accuracy and stroke rates might be detected from the GPS data.

Talos Rowing is an open-source Android application that shows speed, stroke rate, distance and some other metrics about the rowing technique based on the mobile device's GPS and accelerometer sensors. It can be used to view the different parameters live in the boat, but the data can also be recorded for later offline processing.

Rowing in Motion is an iOS application similar to Talos Rowing, but with the focus on analysis of the rowing technique. The boat's acceleration is displayed together with an average stroke acceleration pattern. The initial orientation of the device is used for calibrating the 0–90 degrees vertical tilt. The smartphone does have to be aligned horizontally, not allowing to tilt sideways, to be able to detect the boat's movement direction.

All these solutions are shortly mentioned in table 1, which also states the price, if applicable.

Table 1: Products for Rowing

Name	Features	Price
SpeedCoach GPS	*commercial product* displays distance, speed & stroke rate logs 200-data points, with fixed 100 meter interval GPS & accelerometer data	$ 399
Swiss Timing	*regatta timing service provider* live tracks distance, speed & stroke rate every 25 meters logs speed & stroke rate every 50 meters GPS & possibly accelerometer data	unknown
Talos Rowing	*Android app* displays and logs distance, speed & stroke rate GPS & accelerometer data	open-source
Rowing in Motion	*iOS app* displays and logs boat acceleration & stroke rate logs GPS position GPS & accelerometer data	€ 90

3 Hardware

The success of the usage of sensors in rowing is mainly dependent on the price of the individual onboard devices. At least 80 devices will be needed to track a full rowing regatta. Therefore the price per device needs to be as low as possible. The aim is to use easily available consumer electronics in order to keep the price low.

The selection of hardware is mainly based on the price of the components. Furthermore the ease of use is taken into account and therefore only completely functional modules are selected. These modules consist of the actual sensor or processing chip and the supporting circuitry in order to enable communication and a reliable noise free power supply. By using complete modules only jumper wires need to be attached to the correct pins to get a working system. It can even be chosen to put the full system on a breadboard for easy development and testing.

This chapter is divided into sections which describe the selection and available function of each of the hardware components of the onboard device. The main processing controller board is discussed in section 3.1. In section 3.2 the accelerometer details and its communication work-flow is presented. The GPS module and the way GPS measurements are made are examined in section 3.3. Additional data storage and the process of logging rowing data is outlined in section 3.4. Section 3.5 discusses the communication module and the network routing and communication that is involved, and finally in section 3.6 the power supply and the power consumption of all components is shortly presented.

3.1 Controller Board

Probably the most important part in every embedded system is the microcontroller. This component handles the communication between the different modules, processes incoming sensor data and acts accordingly. The microcontroller has to be programmed to be able to communicate on the different device busses, but also needs a general program which decides when to perform what actions.

The Arduino platform [16] has been selected for the development of the onboard devices. Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. Preassembled boards with many in- and outputs can be purchased for a low price (< € 20,-) and many accompanying and third party software libraries are available for functions like software serial [17], I^{2}C [18] and SPI communication [19], GPS NMEA data parsing [20] and SD card interfacing [21].

The Arduino Fio, as shown in figure 1, has been chosen as the appropriate controller board for this scenario since it is specifically intended for wireless applications. It contains an XBee socket, for wireless communication, and a connection for a lithium polymer battery, which can be charged over USB using the included charge circuit. The ATmega328P microprocessor on the Arduino Fio runs at 8 MHz and has 2 KB of SRAM. 30 KB is available for programming and because of its memory limitations additional external memory is needed for data storage in the case of data logging. The microprocessor software is written in the Arduino programming language, which can be edited in the Arduino integrated development environment, and is heavily based on C or C++. When

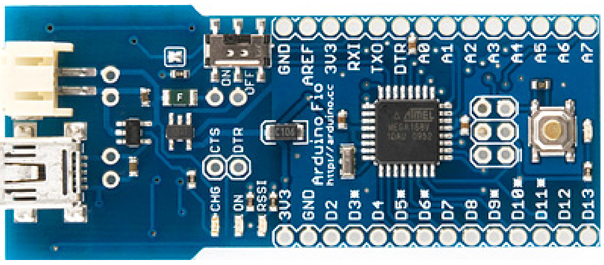


Figure 1: Arduino Fio Controller Board

uploading the program to the board, the software is compiled using AVR Libc and uploaded using avrdude.

3.2 Accelerometer

To measure the boat's acceleration an accelerometer is used. Most cheap accelerometers available today are capacitive micromachined accelerometers that use the spring mass principle. This principle states that due to the moment of inertia a mass can be used to convert acceleration to spring displacement. A small mass is attached to the end of a spring and when the whole body is moved, the spring is stretched from its equilibrium position for a certain distance. This distance can be measured and from this it can be calculated what acceleration was applied to the sensor. In capacitive micromachined accelerometers this happens using an elastically suspended plate [22] which follows the acceleration with a delay. The distance between the fixed and elastic parts of the accelerometer changes due to the acceleration and this difference can be detected and used to calculate the acceleration.

The Freescale MMA7361 and the Freescale MMA8452Q accelerometers were compared because of their low price and availability on a breakout board. Both accelerometers are available for under € 10 and have a selectable measurement range (±1.5g/±6g and ±2g/±4g/±8g respectively) which is applicable for rowing (mostly below 1g, with outliers around 1.5g). The MMA7361 is an analog device, having the necessity to sample the acceleration at regular intervals by the microprocessor. The quality of the microprocessors ADC (10-bit for the Arduino Fio) determines the resolution of the measurements. This device is very easy to use, since no additional communication protocols are required. The MMA8452Q, as shown in figure 2, is a digital accelerometer capable of communicating over I^{2}C. It samples the acceleration at an internal rate of 1600 Hz and provides a selectable output data range with averaged values. The internal 12-bit ADC takes care of the digitalization of the signal and interrupts can be used to signal when new data is ready. The latter accelerometer is chosen based on the internal processing abilities and the relaxed timing conditions for the microprocessor. Even when setting the output data rate to 1 Hz, the full acceleration during that second is taken into account. This accelerometer also has a higher resolution while consuming less power.

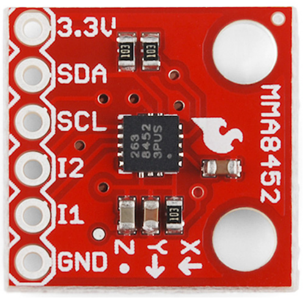


Figure 2: MMA8452Q Accelerometer

3.2.1 Accelerometer Measurements

When the MMA8452Q accelerometer's full-scale is set to 2g and resolution to 12-bits, the measurement range is -2g to +1.999g, and each count corresponds to 1g/1024=1 mg. Absolute peaks during rowing rarely exceed 1.5g so this range closely fits the purpose.

To improve the output of the accelerometer, the high resolution output mode is used which reduces noise. Oversampling is done at a ratio of 32, in which case the internal 1600 Hz rate is used to sample the acceleration and the output is averaged to give 50 Hz updates.

Communication with the accelerometer happens over I^2C. I^2C is a digital communication protocol invented by Philips using a multi-master serial single-ended bus. A lot of low-speed peripherals use this protocol to communicate. The advantage of this protocol is that it only needs two wires to communicate, but therefore communication can only be half-duplex.

In the rowing tracking application it is critical to handle the data of the accelerometer in time. When other parts of the software spend as long as one accelerometer reading (0.02 sec at 50 Hz), the samples are discarded and overwritten. Although it can be verified if a sample is overwritten, it is not possible to know how many samples are overwritten. This puts extra difficulty in the usage of accelerometer data, since missing samples will result in skewed timing and crippled acceleration sums.

The MMA8452Q stores its 12-bit accelerometer data in six registers: an LSB and MSB register for all of the three axes. An additional register is used to signal data readiness and data overwrites. These registers are placed conveniently so that a sequential read of seven registers can read all this data at once.

The Wire library [18] used in Arduino for I^2C communication runs at 100 KHz by default. By changing the speed in the source-code of the library to 400 KHz, a speed improvement on seven sequential register reads could by achieved of almost 100% (1400 microsecond vs. 715 microseconds for a single sample read).

The usage of the Wire library for I^2C communication in an interrupt service routine (hereafter ISR) is difficult because of the blocking nature of the I^2C library. One could opt for bit-banging the I^2C communication to be able to use this within an ISR, but for this project a lower output data rate (hereafter ODR) is used. The lower ODR would not negatively affect the accuracy since the oversampling capabilities of this chip are being used.

3.2.2 Static Error

During initial testing it was found that the accelerometer, while laying still, gave readings which account for more than 1g of gravity. Since the accelerometer was put in the middle of the breadboard, many wires pass the accelerometer closely and could cause electrostatic interference. A test was done to determine if the error on the measurement was caused by the interference or if there was another source for the error. The accelerometer was put at a distance apart from all the other components and jumper wires, using long and straight wires directly to the accelerometer's breakout board. Data was logged for at least ten minutes.

During the recorded period the standard deviation of the measurements was very low and there was almost no deviation from the average, giving no sign for electrostatic charges or interference. When such interference is present, high peak noise on the readings is expected. This specific accelerometer gives an averaged >1g reading over the ten minute period. To see if this specific accelerometer was off, another test was done with another accelerometer of the same brand and type. This second accelerometer gave almost the same results, although the averaged readings were just below 1g. It can be concluded that the error could be caused by an incorrect calibration of the accelerometer. This offset has to be taken into account when using the accelerometer data. A long-term average over every axis can be used, which can be subtracted from the measurements to overcome this offset.

3.3 GPS

For absolute positioning the Global Positioning System (hereafter GPS) is used. This system consists of space-based satellites that provide location and time information to receivers anywhere where there is a line of sight to four or more GPS satellites. The EM-406a and the Parallax PMB-648 GPS receiver modules are compared for use as they are both available for just under € 40. Both modules use the SiRF StarIII chipset and a built-in patch antenna and therefore have very similar specifications. They both have a built-in rechargeable battery for memory and RTC backup so that a quick cold start time can be achieved. Communication with the modules can be done over both TTL and RS-232 serial with NMEA commands. The EM-406a is a bit more expensive, but provides a PPS timing signal and has a slightly lower power consumption. The PMB-648 however operates on 3.3–5V, while the EM-406a operates on 4.5–6.5V, which makes the PMB-648 easier to use with the 3.3V Arduino Fio. Because of the similar performance, the lower price and the applicable voltage levels, the PMB-648, as shown in figure 3, is selected for the onboard device.

3.3.1 GPS Measurements

The Global Positioning System is a space-based satellite navigation system that has about 30 active satellites orbit the earth. Each satellite is constantly sending out signals telling the sending satellite's ID, the position of the satellite and the time the data were sent. A GPS sensor receives these transmissions from multiple satellites which are at that point visible to the receiver. For each of these received signals the sensor calculates the difference in time of the reception of the signal and the time the satellite sent the signal. Using the speed of light,

Figure 3: PMB-648 GPS Module

the distance to each of the transmitting satellites can then be calculated. To determine the actual sensor's location, the distances are used to draw a sphere around each satellite. The receiver should be located on the surface of each of these spheres, so the point where all the calculated spheres intersect, is the exact location of the sensor. Due to various error sources the location can usually be determined to within 15 meters, and with more sophisticated GPS receivers and during low noise situations to about 6.8 meters [7].

The PMB-648 GPS module is configurable to output several types of data and has many settings. It uses the marine NMEA-0183 protocol to communicate over a RS-232 interface. At the initialization of the controller board program the GPS module is set to 38400 baud to speed up communication, which is by default at 4800 baud. Most commercial GPS sensors have a feature called 'static navigation'. This feature is designed for motor vehicles and freezes the position and fixes the speed to zero at very low speeds. Hereby it cancels the drift at low speeds which are a result from the natural inaccuracy of GPS. Since for the rowing tracking application the output needs to be as exact and raw as possible, static navigation is disabled. The GPS module defaults in outputting many NMEA messages which not only give information about the time, position, speed and course of the sensor, but also provide information about which satellites are in view, what their received signal strengths are and many other types of information. For the tracking application only the RMC NMEA message is enabled which contains position, velocity and time information and no other superfluous data.

The Arduino Fio has only one hardware serial port, which is always in use by the XBee communication module. Since the GPS module also communicates over a serial line, a software serial library is used. Such a library operates on any of the digital pins of the Arduino and replicates serial functionality. Instead of the natively supported SoftwareSerial library [17] the AltSoftSerial library [23] is used. This library improves the performance by using a hardware timer to overcome blocking transactions, a type of transaction which is in use by the native SoftwareSerial library. Performance improvement is mostly seen on low baud rates, but is not negatively impacting at higher data rates.

3.4 Data Storage

For the purpose of data logging additional storage memory must be used, since the microcontroller only has 2 KB of SRAM, 1 KB of EEPROM and 30 KB

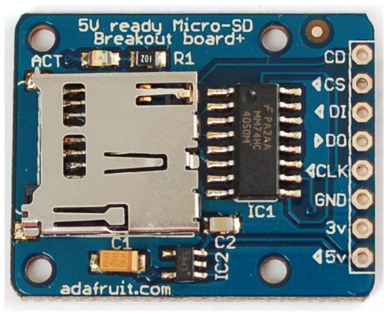


Figure 4: MicroSD Breakout Board

of flash memory, which is far too little for logging purposes. Storage memory can be added in the way of EEPROM, RAM or flash memory. The latter is non-volatile memory and easily available in the form of SD cards. Cards are available at every electronics shop and pack gigabytes of data storage for a low price. Every modern day SD card can communicate in a low power SPI mode, which is easy for the microprocessor to use. In this case the Adafruit MicroSD card breakout board+, as shown in figure 4, is used as an SD card socket with SPI connections breakout. The additional features of the breakout board are not used.

3.4.1 Data Logging

Communication with the SD card happens over the Serial Peripheral Interface bus (hereafter SPI). SPI is a synchronous serial data link between two or more devices that operates in full duplex mode. The master on the bus (in this case the microcontroller) initiates the data frame and possibly uses a slave select line in the case that more than two devices operate on the same bus. For data logging purposes a single file on the SD card is made. Every time new data arrives this data is appended to the log file. However any bytes written to a file on the SD are not physically saved until the flush command is given. A flush causes the current data block to be written, the directory block to be read, updated and written again. Since this many steps need to be made, a flush command is slow. When the microprocessor was constantly writing the ‘number of milliseconds since the board began running the current program’ at its maximum speed, and directly flushing after every write, only 89 writes per second could be achieved. When other tasks need to be performed by the processor as well, this speed could even drop further. Another test was performed to see how the performance would benefit of postponing the data flush. Tests were done in which a flush was performed every 5, 10, 30 and 60 seconds. Writing speeds of 2500 Hz were now easily obtained.

For logging purposes the received data from the GPS sensor and the accelerometer are written to the SD card as soon as new measurements become available. The highest possible ODR for the accelerometer is determined by testing the amount of data overwrites that happen in the accelerometer. A data overwrite occurs when the measured data by the accelerometer is not read by the microcontroller before a new measurement is made. Since only one set of registers is reserved for a single data sample, the accelerometer overwrites old data as soon as it is measured. The highest possible ODR therefore depends on the available processor time for reading the accelerometer data. Any blocking methods, such as the SD flush, could cause the processor to stall and miss acceleration data. It was found that when flushing the SD card data every 30 seconds, a 50 Hz accelerometer ODR could be reliably written to the tested SD card. However, even in this case overwrites rarely happen (0.01% of the time). It could not be determined what the cause for these sporadic overwrites were, but it is thought that the wear levelling algorithms present on most SD cards could have caused these unpredictable effects.

During the last national regatta of 2013 several races were measured using the onboard data logging device. Three of the same onboard devices were built, which have a microcontroller board, a GPS module, an accelerometer, an SD card and a battery for power. The device was put in a waterproof enclosure and duct taped at a convenient free space on the boat's deck. A total of 14 races in different experience classes were recorded: men's coxless four, men's coxed four, men's eight, men's lightweight single scull, men's lightweight coxed four, woman's single scull, woman's coxless pair, woman's coxless four, woman's lightweight coxless pair, woman's lightweight double scull. The aim was to get as much variety as possible, to acquire a dataset that represents the full scale of rowing. Data was recorded from the moment the enclosure was closed till the moment the device was taken from the boat and opened again. This raw data would therefore match data acquired in real-time when the onboard device would have been sending the data to the shore wirelessly.

Unfortunately one of the onboard devices, which had been used to record four individual races, had problems with data logging. When analyzing the obtained data, it was seen that many accelerometer data overwrites had happened. This specific device contained a rather old SD card, which apparently had slower writing speeds than the other cards used. Because of the slower writing speed, the processor could not maintain the fast interval for reading the accelerometer data. Since the overwrites happened randomly through the collected data, and the actual data could not been retrieved, the accelerometer datasets measured with this onboard device had to be discarded. The device did not experience problems with recording the GPS data, so for GPS measurement analysis these datasets could be used.

3.5 Communication

To achieve a useful rowing tracker, the sensed data from the boat need to be sent to the data collector on the shore in real-time. There are not many wireless communication solutions available that are able to directly bridge the 2000 meter that rowing tracks are long. One of these solutions makes use of the cellular network. Due to the rise of the smartphone, Internet is available almost everywhere in the form of GPRS, 3G and since recently also 4G. In embedded systems a GSM module can be used to utilize these networks. Simple GPRS modules are available for prices ranging from € 50 to € 115. Although these modules would all have a direct communication line with the data collector through the Internet, the downside is that these modules usually consume a lot of battery power and therefore need an additional battery and thus weight more. Furthermore GPRS modules do not natively support any higher level communication and security methods, since they operate as simple modems. Network protocols like TCP/IP need to be implemented and since the modules are connected to the Internet, they are easily accessible for outsiders, so additional security measurements should be taken. The physical sizes of the GPRS modules found are very similar: all modules are about three times the size of the Arduino Fio controller board, making them rather big. Finally, for each of these modules a SIM card with a data subscription from a local telecom provider is needed. Using these GSM modules would not only mean a high startup cost, but also results in a large onboard device and difficulties for the implementation and security.

Since the usage of GPRS modules contradicts the goals set for the onboard device, other communication solutions are considered. ZigBee is the specification of a set of high level wireless communication protocols that uses small, low-power digital radios to achieve a personal area network. In several industrial and research projects, ZigBee modules have been used because of their ease of use, price and transmission range [24, 25, 26]. ZigBee is based on the IEEE 802.15.4 standard and adds the network layer and application layer, and most important the ZigBee device objects. The latter are responsible for the keeping of device network roles, requests to join a network, device discovery and security. In terms of networking capabilities, ZigBee can deal with several types of topologies. A point-to-point network can be used in the most simple case, but star and mesh topologies can be used when many nodes need to be connected. In a mesh network each node is not only responsible for the sending and collection of its own data, but the node also serves as a relay for data of other nodes. By collaboration between the nodes, data can propagate through the network by hopping from node to node until the final destination is reached. Route discovery is normally done as soon as a data packet is put in the transmission queue. If the route is not already known, a route request is propagated through the network and using a smart algorithm the shortest and cheapest (in the sense of route quality) route is found [27]. Since this type of routing reconfigures around broken or blocked paths, it is also called *self-healing*. Mesh networks are very suitable for dynamic networks spanning a large area, since nodes that are too far away for direct communication, can use multi-hopping for their transmissions. Since most of the times the crews will be rowing, the boats will continuously change their position. Due to the self-healing properties of a mesh network, moving nodes will generally not be a problem.

A ZigBee network can consist of coordinators, routers and end-devices. A coordinator forms the root of the network and always has to be accessible. The coordinator manages access to the network and has a repository for security keys, and is a specialized type of router. Both the coordinator and routers can form connections in a mesh topology and accept connections from other nodes in order to create a network. Router nodes can also inject data packets into the network and are therefore a specialized end-device. End-devices are the simplest type of nodes and can only directly communicate with their parent node, a router. They can communicate with any other node on the network using multi-hop, but can also go into sleep mode to preserve power and thereby achieving a longer battery life. Because of the possibility of end-devices going into sleep, ZigBee radios are often used in Wireless Sensor Networks (hereafter WSN). A WSN is a network of data sensing nodes which collect all data at a certain node which stores and processes the data from all the nodes [15]. The rowing tracking application can be seen as a WSN since all boats have their own

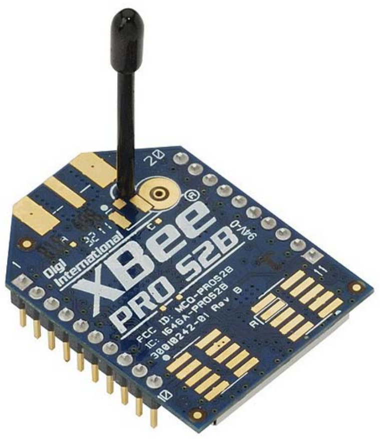


Figure 5: XBee ZB PRO 63 mW RF Module

sensors, collect data and send the data ashore to be processed and displayed.

In June 2005, the ZigBee specification has been released and products are widely available on the market ever since. Many of the available solutions are for domestic usage and therefore have a limited range to about 100 meters. Probably the most well known implementation of ZigBee has been in the XBee ZB modules. XBee ZB modules are easily available ZigBee-compliant chips that are manufactured by Digi International. There are two models: a low cost 1 mW XBee, available at € 20 and a high power 63 mW XBee-PRO, available at € 35. The 1 mW XBee is intended for home usage and has a small range of 100 meters due to its low power. The XBee-PRO in contrast has an advertised range of about 3.2 kilometers and is therefore very suitable in the case of rowing tracking. Specifically the XBee-PRO ZB PRO S2B module, as shown in figure 5, is selected because of the high transmission range and the ZigBee mesh networking support.

The XBee-PRO ZB S2B module provides mesh networking according to the ZigBee standards, with additional buffer methods and fault tolerance. Security can be enabled and the unit tightly integrates with the Arduino system, making it easy to operate. The module can operate either in AT command mode, in which the module acts as a physical serial communication line, or in API mode, which provides more control over messaging, network diagnostics and the possibility to extract the sender's ID. The advertised outdoor line-of-sight range is 3.2 kilometers and data throughput can be achieved up to 1 Mbps. Even when not having intermediate nodes available this range should well suit its purpose. The indoor/urban range is advertised as only 90 meters, which indicates that obstructions limit the range extensively.

Some simple tests were done to determine the real range of the XBee module. A microprocessor was programmed to sent the current GPS reading via an XBee radio to a static receiver XBee every second. The receiver records the transmitted location of the beacon device to later plot distances. The beacon was then moved away from the receiver by walking till the receiver lost the signal for more than ten seconds. It was found that in a line of sight, a distance of more than 1500 meters could be achieved, although not always reliable. Obstructions clearly block the signal and to achieve a more reliable communication network over the same range shorter distance intermediate routers should be used.

To test the mesh network capabilities along the 2 kilometer track, the same beacon test was performed again but this time with router nodes located every 500 meters between the start and the finish-tower, similar to [2] in which ZigBee PRO performance is tested for point-to-point and multi-hop transmissions. During this test it was possible to transmit data along the full length of the track, although sometimes packets were lost and sometimes multiple packets were delayed and received later all at once. Both these issues could be attributed to the loss of a route to the receiver node. The beacon node starts within range of the coordinator and therefore has a high chance of creating a direct route. The further the beacon gets, the poorer the signal strength becomes. At some point the beacon moves out of the coordinators range and has to discover a new route to the receiver, via a router node. Since in normal network discovery, link status messages are only sent 3–4 times a minute [27], there is a possible time span of 20 seconds in which there is no valid route to the receiver. Since the XBee module has a serial buffer and a retry scheme, in which the module re-transmits data up to four times until an acknowledgment is received, a small amount of time without a valid route can be overcome, but when the buffer is full, data is lost. The latter case accounts for the missing packets during the test and the former case where there is still data in the buffer and some data is being re-transmitted, accounts for the case where multiple packets are delayed and later are all received at once.

The solution to the issues of passive route discovery (only perform route discovery when there is no response to a link status message) was found in many-to-one routing, which applies active route discovery for a route to a data collector. In networks where many devices must send data to a central collector, the normal routing scheme requires significant overhead: every device which wants to send data has to perform its own route discovery. Instead in many-to-one routing, a single many-to-one broadcast transmission is sent from the data collector to establish reverse routes on all devices [27]. When a device wants to send data to a data collector and finds a many-to-one route in its routing table, it will not perform route discovery but starts transmitting along the known route. The many-to-one route request is sent periodically by the data collector to update and refresh the reverse routes in the network, based on the current link signal strength. This way, when a roving node moves away from the coordinator in the direction of a router, the link signal strength to the coordinator will at some point be less than the signal strength to the router, while still in range of both devices. When a many-to-one route request is then sent, the reverse route is updated to make use of the router instead of keeping the direct route to the coordinator. Therefore many-to-one routing will make sure that a valid route to the data collector is available at all times. When performing the beacon test again, with many-to-one routing turned on, no data was lost when moving along the track with the same speed as a rowing boat at average racing speed.

In wireless applications security is always an important consideration. Unauthorized access to the network could lead to injection of malicious data or worse, data manipulation or loss. The ZigBee specifications also consider secure communications. The security architecture is implemented into different layers of the communication protocol in order to ensure message integrity, confidentiality, and authentication. Security can be applied to the network layer and to the application support sublayer (hereafter APS) [27]. In the network layer packets are encrypted and authenticated using 128-bit AES encryption with the network key and devices can only communicate in a network when they share the same key. A frame counter is used against replay attacks. On the APS layer, security can be used to encrypt data between source and destination. This provides end-to-end security, which for the application of a rowing tracker is not necessary, since the whole network is considered secure. A link key is preconfigured into all XBee devices so that the network key does not have to be exchanged unencrypted. Furthermore the coordinator is configured to only allow nodes to join to the network that share the same link key.

3.6 Power

To enable all the components to work well, a reliable power source needs to be used. Various rechargeable batteries have been compared, but the cheapest available battery had the correct plug and performed well with the Arduino Fio. The battery used is a brand-less polymer lithium ion battery with a capacity of 1400 mAh at 3.7V. The battery shown in figure 6, has overcharge and short circuit protection and a cut-off voltage of 3.0V. According to the Arduino Fio specifications the input voltage cannot be lower than 3.35V although in practice no problems have been experienced on low power levels. The Arduino Fio provides a built in charge circuit for charging the battery over USB.

The power consumption and the price per module are shown in table 2. In the worst case scenario the onboard device can run for almost four hours on one battery load. This probably is a little more, since the XBee will not constantly be transmitting and the SD card will not be constantly writing. This duration is more then enough for the usage during rowing regattas, since boats will be on the water for less than two hours consecutively.

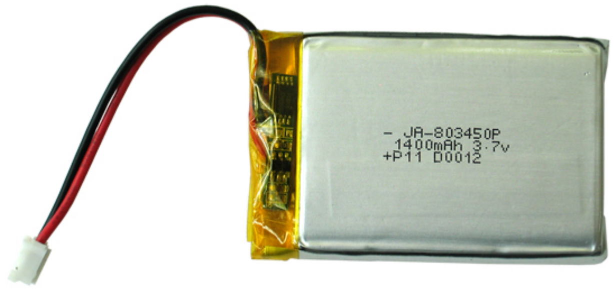


Figure 6: 1400 mAh 3.7V polymer lithium ion battery

Table 2: Power consumption & price per module

Module	Power consumption	Price
Arduino Fio	50 mA	€ 16.50
Accelerometer	0.165 mA	€ 7.90
GPS	100 mA	€ 27.96
SD card idle SD card transfer	1 mA 100 mA	€ 10.56
XBee idle XBee receive XBee transmit	15 mA 47 mA 205 mA	€ 33.00

4 Stroke Rate Tracking

In rowing the number of strokes per minute always has been an important parameter. From this number it can be inferred if a crew is accelerating or if they are tired and just keep the same pace. Because this number gives so much information, the inclusion of the stroke rate in the live tracker would have a great advantage.

This chapter explains how the stroke rate is extracted from the acceleration data. Section 4.1 explains how the anatomy of a rowing stroke results in a typical acceleration pattern. Related work on stroke detection is treated in section 4.2. A solution to the problem of the unknown orientation of the accelerometer is given in section 4.3 and finally two actual stroke detection algorithms and their results are discussed in section 4.4.

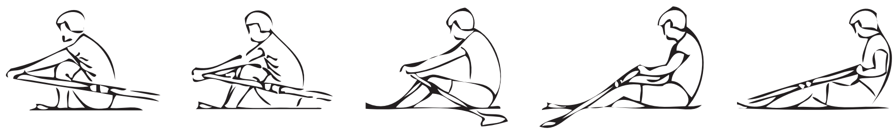


Figure 7: Stages of the Rowing Movement

4.1 Anatomy of a Rowing Stroke

The rowing stroke by itself is a repeated movement in which legs, back and arms work together to move the blade of the oar in the water and propel the boat forward. It is typically divided into two parts: the drive and the recovery. The drive happens immediately after the catch and consist of the leg push, then a back swing and finally the arm movement as shown in figure 7 from left to right. After the finish of the drive and the blade removal from the water it is time for the recovery. The recovery is similar to the drive, but then backwards: first move the arms away, then bend the back and finally pull-in the legs. This results in a typical boat acceleration as depicted in figure 8 [28].

The graph starts with a part of the recovery: the pull-in of the legs. The movement of the body weight leads to a deceleration till the rower reaches the front position. The catch is made which slightly decelerates more till the legs are pushed down which results in the acceleration becoming positive, depicted in figure 8 just after the dotted line marked 'catch'. Swinging the back still results in some acceleration but it is not possible to keep the high level of the leg push so it results in a small dip. Finally the arm pull is made which further accelerates the boat till the final swing and the finish. At this point the boat's mass takes over and accelerates further during the recovery with the arms and back. Then the cycle is started over again.

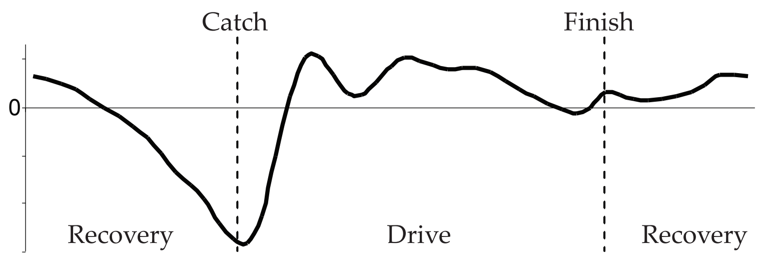


Figure 8: Typical Pattern of Boat Acceleration during the Stroke Cycle [28]

4.2 Related Work

The Talos Rowing Android application [14] algorithm uses the acceleration along two axes of an Android mobile phone to determine the amount of strokes per minute. Because of this, it is only possible to use this app if the orientation is known (rower facing or cox facing, portrait mode or landscape mode) and the tilt of the phone is between 90 and 180 degrees. Measurements are taken with a speed of around 50 Hz (depending on the phone's accelerometer and capabilities). A low-pass filter is used to compensate for the force of gravity. By subtracting the long-term average over one axis from the measured value, only the dynamic component of the acceleration is taken into account. The formula of Pythagoras is applied to the dynamic components along the two axes to obtain the magnitude of acceleration.

Stroke detection is done using a model of half a sine wave. To detect a stroke the algorithm uses a number of checks. First the negative acceleration during the catch has to pass a certain threshold to exclude noise. Second the negative acceleration has to have at least half the amplitude of a low-pass filtered average over the last negative peaks. This dynamic threshold makes sure that other decelerating factors, like the dip just before the finish are not recognized as a stroke. When the negative acceleration becomes a positive acceleration again, and the peak passed all checks, a stroke is detected and the time is recorded. Two sequential time stamps are then used to determine the stroke rate.

4.3 Orientation Invariant Measurements

Since the onboard device has to be as unobtrusive as possible, the decision was made that there should be no constraints on the placement of the onboard device. This yields that the accelerometer data cannot be directly used for analysis. First the axis parallel to the boat's movement needs to be inferred from the accelerometer data.

4.3.1 In Theory: Orientation Invariant Measurements

In [29] a method is proposed which uses gravity to estimate the accelerometer orientation. Accelerometers measure gravitational ('static') acceleration as well as 'dynamic' accelerations caused by the propulsion of the boat to which the sensor is attached. Gravity pulls downward along a single (virtual) axis of the accelerometer and this results in an acceleration output in the opposite direction along that same axis. Since the sensor is fixed to the boat, the sensor's orientation is linked to the orientation of the boat. The force of gravity can therefore be used to estimate the vertical component. The dynamic part of the acceleration is zero on average, so over a chosen sampling interval all readings along an axis are averaged. This average is an estimate of the vertical acceleration corresponding to gravity and is called **v**. The dynamic part of the acceleration **d**, caused by the propulsion of the boat, is then calculated by subtracting **v** from the raw measurements **a**:

$$\mathbf{d} = \mathbf{a} - \mathbf{v}.$$

Using vector dot products, the projection of **d** upon the vertical axis is computed:

$$\mathbf{p} = \left(\frac{\mathbf{d} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}}\right)\mathbf{v}.$$

Next, this dynamic vertical component **p** is subtracted from the dynamic acceleration **d** which results in the dynamic horizontal acceleration **h**:

$$\mathbf{h} = \mathbf{d} - \mathbf{p}.$$

Principal component analysis (PCA) is done on the acquired horizontal plane as in [30], to infer the axis of movement along the axis of which the acceleration variation is greatest, usually the rowing direction. It is then still unknown what the forward direction of movement is since during rowing the average acceleration will almost always be 0. The knowledge about the anatomy of the rowing stroke is used to find the forward direction. In [28] it can be seen that the negative peak during the catch is almost always greater in magnitude than the positive peaks during the drive. Therefore the minimum and the maximum in the data is recorded over a chosen sampling interval, typically a couple of minutes. If the magnitude of the maximum is greater than the magnitude of the minimum, the direction of the movement along the axis is backwards.

The achieved vector represents the boat's acceleration along the axis of movement. This vector can be used for further analysis and stroke detection.

4.3.2 In Practice: Orientation Invariant Measurements

Unfortunately the usage of vector dot products, matrix multiplication, principal component analysis and large amounts of data is not possible on the chosen microprocessor due to the limitations in memory and processing power. A method needed to be found which achieves the same goal while also outputting roughly the same resulting data. Inspiration was found in [14] which uses only a couple of variables and simple calculations to obtain an acceleration magnitude from two axes data.

Using a low-pass filter, the long-term average $\bar{\mathbf{a}}$ over each of the three axes is subtracted from the measurements $\mathbf{a} = (a_x, a_y, a_z)$ in order to obtain the dynamic component of the acceleration $\mathbf{d} = (d_x, d_y, d_z)$:

$$\mathbf{d} = \mathbf{a} - \bar{\mathbf{a}}.$$

The axis with the highest amplitude during a certain sampling interval $a_{\max}$ is chosen to assess the direction of acceleration. Since the only propulsion of the boat is mainly done in the direction of the boat's movement, Pythagoras' theorem can be applied on all the three axes of **d** to obtain the magnitude of the acceleration vector $\vec{a}$:

$$\vec{a} = \sqrt{d_x^2 + d_y^2 + d_z^2}.$$

The direction of acceleration in $a_{\max}$ is then determined and applied to the acceleration vector $\vec{a}$

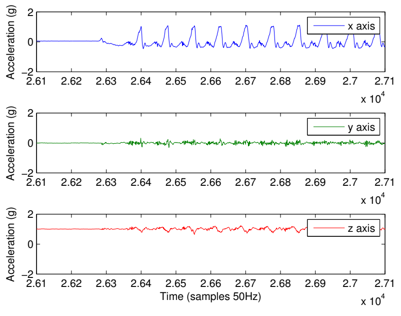
$$\vec{a} = \begin{cases} \vec{a}, & \text{if } a_{\max} \geq 0 \\ -\vec{a}, & \text{otherwise} \end{cases}$$

to turn the acceleration in the correct relative direction. The acquired value now corresponds to the acceleration of the boat along the axis of movement although the absolute direction is still unknown because the sum of accelerations will be zero on average. Again the knowledge of the rowing stroke is used to find this direction. A certain sampling interval is used to find the minimum and the maximum of the acceleration and the lowest magnitude is used as being the absolute direction of movement.

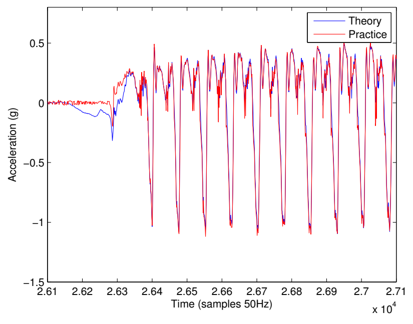
4.3.3 Results

To verify our practical method for obtaining orientation invariant measurements, the output data of the theoretical method is compared to that of the practical method. First the raw data of one of the races was analyzed using both methods. In figure 9a a fragment of the raw accelerometer data is shown. First look at the z axis; this axis has a constant offset of about 1g and almost no variance. This can be explained as a result of the gravity force pulling downwards. Second, the data on the y axis of the sensor has an average of zero and again almost no variance. This axis was sideways to the boats movement and therefore has almost no acceleration whatsoever. Then lastly, the x axis of the sensor has a high variance and shows the typical pattern of a rowing stroke, although it is displayed upside down. From this it can be concluded that the x axis of the sensor was almost completely aligned with the direction of movement of the boat, and that the sensor was not tilted.

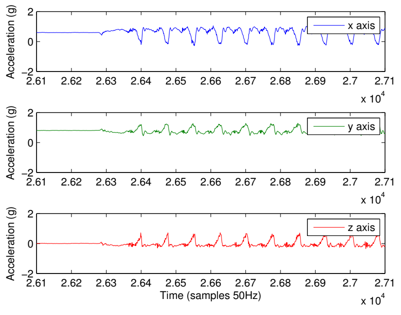
Figure 9b shows the output of the theoretical and practical orientation invariant measurement filters. The stroke pattern is clearly visible in both the outputs. Apart from the initial stroke, the outputs of both filters are very alike. It can be seen that all negative peaks caused by the catch are at exactly the same places. During the drive, the practical filter results in some noise, especially at the point where the back swing starts (dip around 0g). This is probably caused by sideway movement which is incorporated by the practical filter as part of the acceleration. These movements along the up and down and sideways axis of the boat result in fluctuations of the resulting acceleration value. However, in rowing these movements tend to be small and average out over time. This would mean that the resulting magnitude of the acceleration obtained by the practical filter will be a little higher than the magnitude obtained using the theoretical based filter.



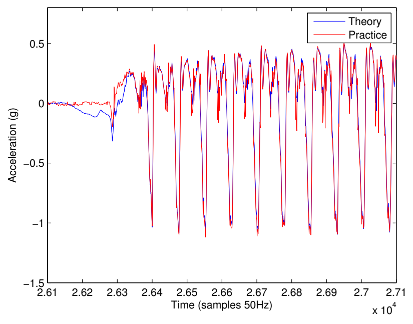
(a) Fragment of Raw data



(b) Theory vs Practice on Raw data



(c) Fragment of Rotated data



(d) Theory vs Practice on Rotated data

Figure 9: Orientation Invariant Output

Since the raw data of the accelerometer was very nicely aligned with the rowing direction of the boat, a second evaluation has been performed on a transformed dataset. The raw data has been multiplied with a random rotation matrix to mimic a random oriented placement of the sensor on the boat. The rotated data is shown in figure 9c. In this figure it is visible that the accelerometer sensors along both the x and y axis measure parts of the static component caused by gravity, indicated by the offset on these axes. All three x, y and z axes measure parts of the dynamic acceleration caused by the rowers' propulsion, visible as negative peaks on the x axis and positive peaks on the y and z axis.

In figure 9d the output is shown of the theoretical and practical orientation invariant measurement filters applied to the rotated dataset. Again the stroke pattern is clearly visible in both the outputs. The output of the filters on the rotated data is by eye not distinguishable from the output of the filters on the raw data. The peaks are again at exactly the same place and also the noise during the drive is present. From this evaluation it can be concluded that both the theoretical and practical filters will perform correct in real-life situations. For easy stroke detection the negative peak at the catch should easily be recognized in the output data and this is the case for both filters.

4.4 Stroke Detection

To be able to show the stroke rate of a crew, it is necessary first to detect a single stroke. According to [28] the negative peak of the catch has the highest correlation with the stroke rate and can therefore be perfectly used in stroke detection. The interval between two strokes can then be used to determine the amount of strokes per minute.

4.4.1 Simple Peak Detection with Static Threshold

First simple peak detection has been applied to the accelerometer's orientation invariant measurements. The negative peak of the catch is searched for below a certain threshold. When the signal becomes higher than zero again, the detected peak is stored. The time between the last two peaks detected is used to calculate the stroke rate.

As seen in the top graph of figure 10, the peaks are detected well. However, this computational inexpensive algorithm sometimes misses a starting or faulty stroke. Besides the threshold needs to be adjusted for each type of boat, since the negative peak at the catch in for example a single skiff is less deep than in an eight. This makes the algorithm harder to use in a real-life situation.

The peaks found by this algorithm are the absolute peaks in the signal. Since the deceleration caused by the rowing stroke is a combination of leg, back and arm movement, the absolute peak is not always exactly at the same position during the stroke. Due to small variations these peaks shift a little in time. This difference in time is then propagated to the calculation of the amount of strokes per minute (hereafter SPM). As can be seen in figure 11 the outcome of this algorithm is very erratic. When a peak is slightly shifted, two SPM readings are effected, namely the one given with that peak and the one after, since both use the same time stamp, which has the shifted peak. The output therefore wags around the actual value and is not very consistent.

4.4.2 Averaged Peak Detection with Dynamic Threshold

To overcome the shortcomings of the simple peak detection algorithm the specific problems in this algorithm were targeted. First the static threshold is made dynamic. During a certain sampling interval the minimum acceleration is recorded. Five of these minima are stored. When a new minimum is processed, the lowest value (the highest negative peak) of these five minima is multiplied by 0.6 and set as the new threshold. The threshold also has a set minimum so that during rest, the threshold will not become too small and cause noise to trigger a false stroke detection. The threshold is initialized at a value of -0.25. Specific initialization is needed since the algorithm needs to detect peaks to start updating its threshold. The value is deliberately chosen close to zero, so that for all types of boats at peaks are detected. During the time of rowing up to the start, the algorithm will adjust the dynamic threshold so that it will match the crews minimum acceleration.

The second improvement is made on the absolute peak detection. After careful inspection of the acceleration data it was found that the peak itself is not very time constant, but the threshold passing points are. When only taking into account the moment in time when the acceleration drops below the threshold, a change of the dynamic threshold could cause a shift in time again.

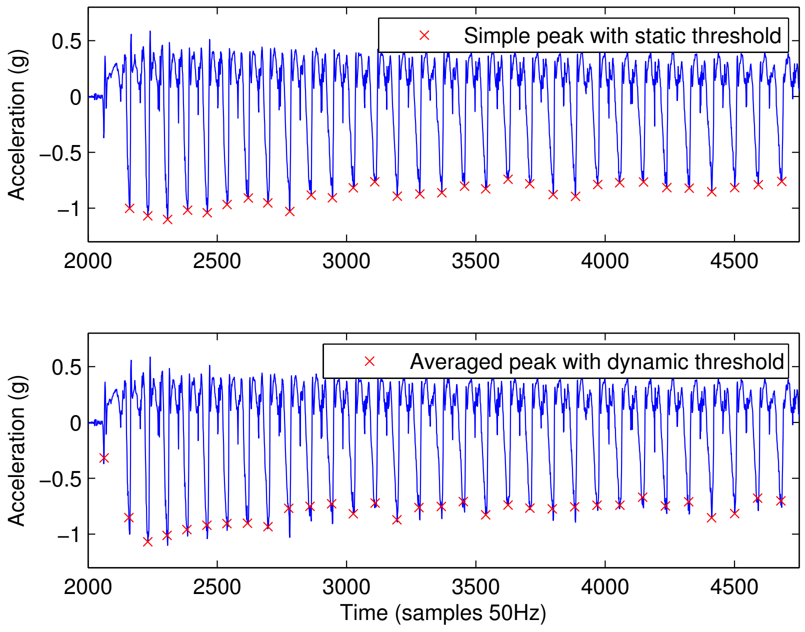


Figure 10: Peak Detection Methods

Therefore the averaged time between dropping below the threshold and rising above the threshold is taken as the peak of the signal. In the bottom graph of figure 10 the peak positions are visualized. It might not be visible to the naked eye, but the peaks are spaced more equal in time which is confirmed in figure 11 by the less erratic movement of the output of this algorithm.

4.4.3 Results

A validation of the SPM rates has been done using an onboard commercial device from Nielsen Kellerman, the StrokeCoach [13]. This device uses input from a three-axis accelerometer to display stroke rates in the range 10-115 SPM with a $1/2$ SPM resolution. Furthermore it stores the stroke rate and count automatically every 10 seconds.

In figure 11 the results of the simple peak detection with static threshold, the averaged peak detection with dynamic threshold and the StrokeCoach SPM data are displayed. The averaged peak detection algorithm closely matches the StrokeCoach data during the first 50 seconds. Then the StrokeCoach data starts to fluctuate quite a bit, but the averaged peak algorithm matches the StrokeCoach data always within 2 SPM.

Unfortunately it turns out that the data from the StrokeCoach is not always as accurate as is advertised. After questioning various coaches and rowers, it seems that the data always has the tendency to fluctuate around the actual value with about 2 SPM variation. To verify this, a simple combined movement test has been done. In figure 12 three StrokeCoaches with Surge Rate Technology are attached to a single piece of cardboard and moved all at once. It can be

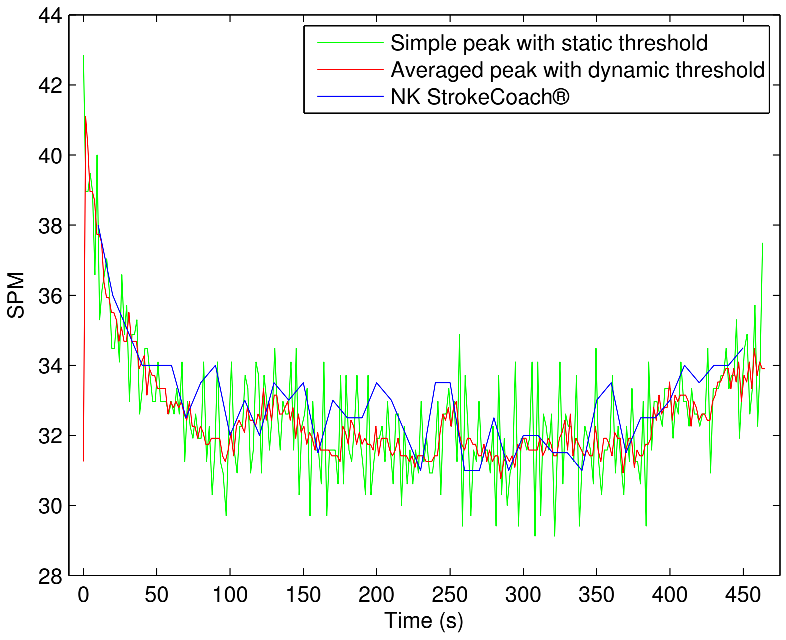


Figure 11: Comparison of Peak Detecting Algorithms on Accelerometer Race Data against StrokeCoach SPM data

seen that all three devices give a different SPM reading varying up to 1 SPM between devices.

Since the StrokeCoach data was not having enough accuracy, another source for validation needed to be found. A video recording was made of a training session while the onboard device was storing its accelerometer data for post-processing. The video-footage was aligned in time using filmed footage of a GPS clock on a mobile phone and the GPS recordings of the onboard device. The footage was manually annotated for strokes. The footage was taken at 24 frames per second so every stroke was measured with a 0.04 second accuracy. Figure 13 shows the calculated SPM values from the video footage in red and the result of the dynamic threshold detection with averaged peak in blue.

Two segments of race rate rowing are included in the graph, as well as a segment of low tempo rowing. Unfortunately the video footage was interrupted after the power 20 push at around 90 seconds. At this very same moment, the switch between race tempo and low tempo rowing, the dynamic threshold detection with averaged peak algorithm misses two strokes. This is caused by the fact that during race tempo the negative peak at the catch is deeper than during low tempo rowing. The dynamic threshold needs some time to adjust to this and thereby misses these two first strokes of the low tempo. Since the tempo thereby drops below 10 SPM, a 0 SPM value is given as explained in section 4.5.

It can be seen that during these 150 seconds of data, the detection algorithm shows a close match to the video footage data. The data in the footage actually

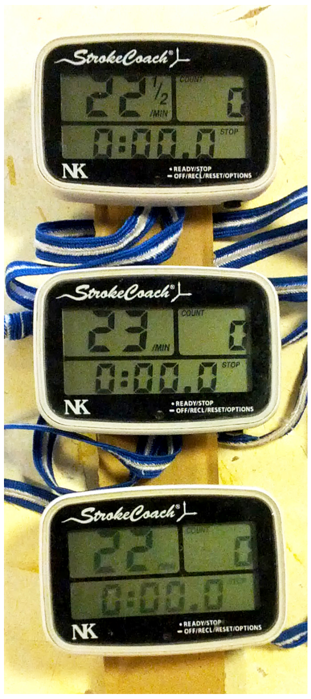


Figure 12: Three StrokeCoaches with Surge Rate Technology displaying different SPM while moved together

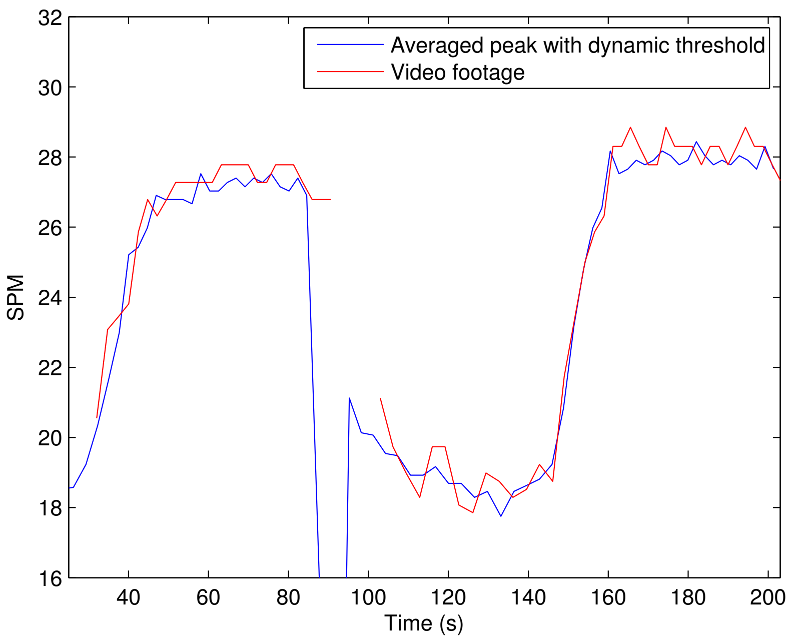


Figure 13: SPM Extracted from Video Footage Compared to the Averaged Peak Detection with Dynamic Threshold Algorithm

have a higher variance during the low rate rowing. It can thereby be concluded that the orientation invariant filter and the stroke detection algorithm perform at least as good as the state of the art commercial solutions and might even perform better.

4.5 Tracking

Every time a new stroke is detected, the microcontroller sends a message through the XBee network to the shore's data collector with the peak's device time in milliseconds. The server can use two subsequent messages to calculate the current SPM and display this on the live tracking application. When the detected interval between two peaks is larger than 6 seconds, the value is ignored since in that case the stroke rate is lower than 10 SPM. Furthermore, every time a peak message arrives, a 6 second timer is set for that specific node which results in a 0 SPM rate when no new message arrives before the timer expires. This is needed since when no strokes are made, no peaks are detected and therefore no new SPM rate is calculated. In that case, when a crew is resting, the tracker would keep showing their last stroke rate.

5 Position Tracking

For live tracking purposes the position data is the most important parameter. If this value is available for all crews it can be used to show the crew's current ranking, the distance between crews and the distance to the finish line. This gives the opportunity to fully assess the current race situation and make well-founded predictions of the final score.

In this chapter some methods are proposed to obtain an accurate along-track position of a boat. Section 5.1 explains the algorithms used to calculate along-track positions and the along-track distance travelled by a boat. Furthermore baseline measurement results for rowing races are obtained from GPS data only and compared to the conventional timed results. Section 5.2 proposes two methods to improve the along-track distance measurements. First the use of a cheap static reference GPS receiver for the reduction of omnipresent noise is examined in section 5.2.1, and second the implementation and optimization of a Kalman filter is proposed in section 5.2.2 which combines information from the GPS and accelerometer to improve the measurement accuracy.

5.1 Along-track Distance

Side by side rowing regattas are usually held over a distance of 2000 meters. The six competing crews all start in their own lane next to each other. During the race however, it is not necessary to stay in your own lane. As long as a crew is not bothering other crews, it is perfectly within regulations to zigzag across the track.

For the live tracking purpose it is not necessary to show the absolute location of a boat on the track. Actually this could be confusing to the viewers, since multiple crews of the same club could be racing against each other and when some of them might swap lanes, people might interchange the crews. Another possibility is to display the distance travelled by a crew, since the start of the race. Unfortunately this is also not a usable parameter since the possible zigzagging could cause a higher value than the distance moved along the track since the start of the race.

The along-track distance is the only possible parameter to use in the case of rowing tracking. The absolute positional data therefore needs to be projected onto an along-track distance measurement scale. Several mathematical formulas are used to transform the data and get a position along the rowing track. The along-track position is then recorded every second. The position at the full second before the start signal is given, is used as an offset to correct all further measurements and get the relative displacement along the track.

In section 5.1.1 an evaluation is made of two possible formulas that calculate distances along the globe. Implementation of the translation method from absolute positions to relative distance is described in section 5.1.2 and the results of a comparison between timed race results and GPS measured race results is made in section 5.1.3.

5.1.1 Haversine and Vincenty

As Pythagoras found out in the 6th century BC, the Earth is not a flat disk, but actually has a spherical shape. When calculating distances between two locations this needs to be taken into account. A line along a sphere has a longer length than a straight line between two points crossing the sphere.

For great-circle distance calculations between two points on a sphere, the Haversine formula is used [31]. This formula is an equation used in navigation which calculates the great-circle distances between two points on a sphere from their latitudes and longitudes. It is a simple formula that can be applied in a fixed number of calculation steps and could easily be performed on the selected microcontroller. However, the Haversine formula is only an approximation when applied to the Earth. The Earth's radius varies from 6356.752 km at the poles to 6378.137 km at the equator [32]. Furthermore, the radius of curvature of a north-south line on the Earth's surface is greater at the poles than at the equator. Therefore the Haversine formula introduces small errors because of the spherical geometry and can not give more accurate results than 0.5%. On a 2000 meter rowing track this could result in errors up to 10 meters.

Since a 10 meter accuracy is way too low for proper rowing tracking, other formulas for geographical distance are considered. In 1975, Thaddeus Vincenty developed two iterative methods to calculate the distance between two points on the surface of a spheroid. These methods take into account that the Earth is a oblate spheroid and are therefore more accurate than Haversine's formula. The inverse Vincenty method computes the distance between two points given their latitude and longitude. This formula has widely been used in geodesy because of its accuracy to within 0.5 mm on the Earth ellipsoid WGS-84, which is in use by the Global Positioning System. When comparing Haversine's and Vincenty's formulas on the 2000 meter distance from the start (52.323567, 4.82255) to the finish (52.327808, 4.85106) of rowing track the Bosbaan in Amsterdam, the formulas have an outcome of 1994 meters and 2000 meters respectively. For the application of rowing tracking, the high accuracy of Vincenty's formula would be perfect. However, because of the number of iterations needed to achieve this high accuracy, the computation might not be feasible to implement on the microcontroller.

5.1.2 Implementation

Because of the heavy recursive computations used in Vincenty's formula, it is decided to offload the computations to the receiving server onshore. The GPS measurements are encapsulated in the NMEA-0183 protocol. These messages are time stamped by their nature and it is therefore not a problem that the communication overhead introduces an extra delay in data delivery as long as the data remains FIFO. The offloading also has the advantage that the onboard devices do not need to know on which rowing track they are being used, since this could be setup at the main processing server.

On the server a state for each of the nodes is kept identified by the source address of the data. When a data packet arrives at the server, the GPS measurement time, latitude and longitude are extracted from the data packet and stored in the database. The extracted data is also directly used for computation of the along-track distance from the sending boat. When a race is started, the state of the competing boats are updated with the starting time. Since GPS data is only outputted once every second, the location data from the full second before the start time is used as the boat's initial position. For distance calculations it should not matter that this is not the location at the exact starting

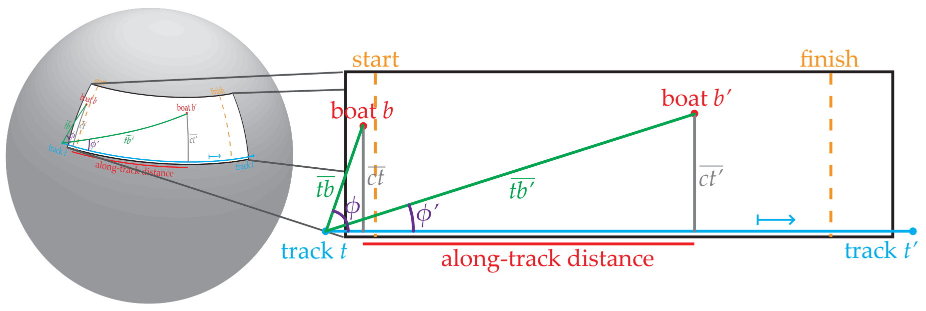


Figure 14: Along-track distance determination using Vincenty's formula

time, since boats are kept at the same spot during the starting procedure. This procedure involves a roll call, and usually takes about ten seconds, which is lengthy enough for a single GPS measurement.

Before the regatta, the orientation of the track is programmed into the server application, by entering a start point t and finish point t', identifying the great-circle path $\longmapsto$ with the orientation of the track. This data is used to calculate the along-track position.

The along-track position is a parameter which indicates in meters how far on the track a point is positioned. The along-track position of a given point is determined using the following method as depicted in figure 14. First the bearing from the start point of the track t to the end point of the track t' is calculated using Vincenty's formula. Then the Vincenty's distance $\overline{tb}$ between the start of the track t and the point b is calculated. This calculation also gives the bearing ϕ from the start point t to the measured point b. The distance $\overline{ct}$ of the point b to the great-circle path $\longmapsto$ is then determined using the cross-track distance formula which needs ϕ, $\overline{tb}$ and the bearing from t to t'. Finally the distance $\overline{tb}$, the distance $\overline{ct}$ and the bearing ϕ are used to determine the along-track position.

When both starting time and the GPS data from the full second before the starting time are present in the node's state, the along-track position during that second is calculated and stored. When new GPS readings are received, the along-track position for these points are calculated as well. The along-track distance can then be calculated by subtracting the along-track position at the start time from the current along-track position. This obtained distance can then be used in the live tracker.

5.1.3 Results

This section displays the results of the Vincenty's along-track distance algorithm using GPS data only to determine finish timing results for 2 kilometer races. Evaluation was done based on race times measured by the conventional timing systems of Time-Team. A very accurate clock is used to time stamp the start during the starting procedure. When boats pass the finish line an umpire presses the finish button giving another time stamp. The delta time is used as the race result-time for a boat.

A GPS clock is synchronized to the conventional timing system clock to be able to cross-reference time stamps. To evaluate the along-track travelled distance, the start time stamp is used and the associated along-track position is used. This is applied as an offset to all subsequent measurements and it can then be calculated what time it took according to the GPS data to travel an exact 2000 meters. Since GPS measurements are only taken every full second, interpolation between the last measurement before the 2000 meter travelled distance and the one after, is used to get the exact sub-second time. From this race-time the sub-second part from the start time stamp is subtracted, since it is clear that no movement was made during that period.

Table 3: GPS Along-track Distance Results of 14 Independent Races

Conventional Timed	GPS Timed	Difference
06:13,74	06:12,91	-00:00,83
06:13,74	06:13,54	-00:00,20
07:00,00	07:01,12	00:01,12
07:17,09	07:17,34	00:00,25
07:27,19	07:28,02	00:00,83
07:28,41	07:29,01	00:00,60
07:30,86	07:30,91	00:00,05
07:35,66	07:36,04	00:00,38
07:38,84	07:39,29	00:00,45
07:53,84	07:54,48	00:00,64
08:25,91	08:26,13	00:00,22
08:28,68	08:28,21	-00:00,47
08:36,52	08:37,35	00:00,83
08:38,13	08:37,57	-00:00,56

A total of fourteen boats were tracked during the last national regatta of 2013 and the algorithm results are compared to the conventional timing system results and shown in table 3. It can be seen that GPS timed race results are very close to the conventional timed results. All differences are below one second, bearing one. The average error is only 00:00,53 seconds and has a standard deviation of 00:00,30 seconds. It can be concluded that the GPS along-track distance is very close to the actual time spent rowing over the 2000 meters. These results are far better than what was expected since non-augmented GPS point positioning has an average error of at least 6.8 meter [7]. While having this error during two individual measurements (start and finish) the possible error is twice the normal error and therefore 13.6 meter. At an average boat speed of five meters per second this would result in a 2.72 second error.

The high performance of the GPS timing could be explained by the fact that not absolute point positioning is used but a derived along-track position is used, which could minimize the error. The error on an absolute point could be directed in 360 degrees. The along-track position only depends on the position along the track and thus along one (virtual) axis. Only when the error is in completely the forward or backward direction of the track, it fully effects the along-track position. Say for example that the error is perpendicular to the track, the along-track position would not be affected by the error. In the case of a 45 degree error only about 70% of the error would make it into the along-track position. Actually in 358 out of 360 cases the absolute positioning error is minimized when using the along-track position. On average this minimization would account for 36% of error reduction, namely 1 minus the absolute cosine integrated over 0–2π. When projecting this error reduction onto the expected error in seconds, this would result in a 1.73 second error at an average boat speed of five meters per second. This value more closely resembles the error seen in the GPS timed results.

Furthermore, some error sources of GPS positioning like satellite ephemeris error, satellite clock bias and atmospheric effects could well give the same or at least similar errors during the complete race. Although in the normal case this would result in an error because one is interested in absolute point positioning, in the case of the rowing tracking this error would be cancelled out because the same error is used in two measurements and is therefore cancelled in the difference calculation.

Finally it can be concluded that the results obtained by using GPS timed races are usable for live tracking rowing races. Although the error is still half a second on average, this is far better than what was expected. This can be explained by the usage of derived positional parameters instead of absolute point positioning.

5.2 Sensor Fusion

For rowing, race timing is done with a precision of one hundreds of a second. For very equally skilled rowing teams, the difference on the finish line is sometimes merely a couple of hundreds, but most often the difference is bigger than half a second and can easily be seen by eye.

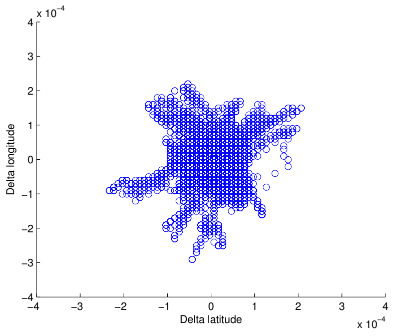
The GPS timing results found in section 5.1 already have this sub-second precision but in order to be able to show even the smallest differences between crews, an additional step is needed. Two methods are researched upon to improve precision. The first is the use of an additional GPS receiver at a static point, which is explained in section 5.2.1. The second is the combination of the accelerometer data for short term and the GPS data for long term precision, using a Kalman filter, as is explained in section 5.2.2.

5.2.1 Simple Differential GPS

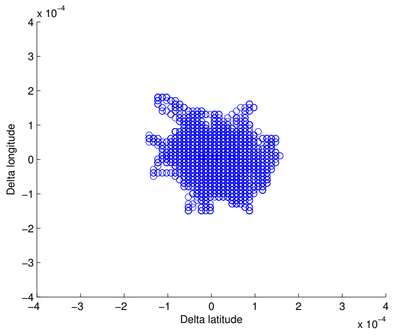
The accuracy of GPS single point positioning has always been limited by many errors, including satellite ephemeris error, satellite clock bias, atmospheric effects, receiver noise, multi-path and before May 2, 2000, selective availability [7]. A general method to correct for these errors is differential GPS (hereafter D-GPS). In D-GPS a fixed ground-based reference station is used to receive the difference between the positions indicated by GPS and the known fixed position. These differences are then applied to the roving GPS receiver's data in order to correct for the reception noise. These implementations can enhance the GPS point positioning accuracy from the 15-meter nominal GPS accuracy to about 10 cm in case of the best implementations.

Normally reference stations broadcast the corrections that need to be applied to the roving GPS receivers data. This implies that a GPS receiver needs an additional antenna to receive the broadcasted data. Since one of the requirements of our onboard devices is to keep them as cheap as possible, the addition of components needs to be avoided. Besides, reception of publicly available D-GPS signals is often only available along the coastline and harder in domestic areas. The further away the reference station is, the less useful the corrections are, since the atmospheric effects are very local.

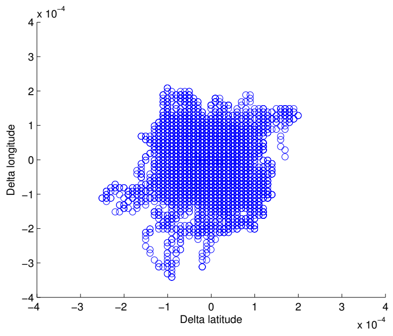
It is therefore opted to use a local GPS receiver to obtain correction data. A test is done in which two GPS sensors have obtained twelve hours of data after an initial hour of warmup time, in order to smooth out its location details. The first half of the data is averaged to obtain the actual location of both receivers. Then the offset to this actual location was calculated for every sample in the second half of the data. In figure 15a and 15b the delta offset from the actual location is plotted. It is clear that GPS detected locations wander around the actual location. This is expected, but it is furthermore expected that both receivers wander in sync, due to the same atmospherical conditions. A D-GPS calculation is then applied by subtracting the offsets of the first sensor from the values of the second sensor. In figure 15c the resulting deltas are shown. It is clear that the output values still wander around the actual GPS location.



(a) Delta GPS Sensor 1



(b) Delta GPS Sensor 2



(c) Delta D-GPS

Figure 15: Delta GPS Readings

Table 4: Standard Deviation Delta GPS Readings

	Latitude	Longitude
Sensor 1	4.075×10^{-5}	5.219×10^{-5}
Sensor 2	3.761×10^{-5}	4.210×10^{-5}
D-GPS	5.250×10^{-5}	6.688×10^{-5}

When comparing the standard deviations shown in table 4, it can be seen that the variance in the D-GPS corrected measurements actually is higher than those of the single sensor readings. This means that it is not possible to use consumer electronics GPS sensors in a D-GPS application. Upon further research it was found that the used satellites in the calculation of the location determine the noise. Since typical receivers switch among satellites fairly frequently based on proprietary criteria that vary from one manufacturer to another, one model of unit to another, and probably even one software release to another, it is not known which satellites are used. If two receivers are only 15 centimeters apart they might already use different satellites in their calculations and therefore have different noise values. This is further explained by figure 16 in which both delta latitude and delta longitude of the two sensors is compared over time. The values of two receivers never seem to synchronize and usage of simple D-GPS using cheap consumer electronics therefore has to be excluded as a possibility in the rowing tracking application.

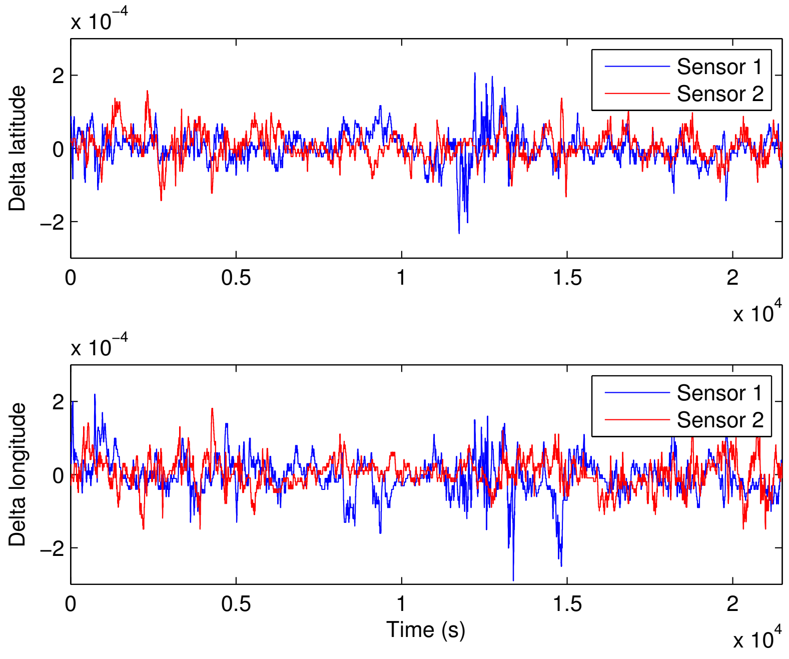


Figure 16: Comparison of delta GPS values over time

5.2.2 Kalman Filter

The output of the GPS receiver is a position relative to the WGS-84 coordinate system. Noise upon the reception of the GPS signals and other noise factors like clock offset cause the output of the GPS to fluctuate around the actual position. The accelerometer measures the acceleration of the boat over three axes and using the orientation invariant filter as proposed in section 4.3 a vector is obtained of the acceleration along the rowing direction of the boat. This acceleration vector can be applied to the speed measured by the GPS receiver to obtain the actual speed of the boat. By integration the speed value can furthermore be translated into a distance. The accelerometer however is sensitive to sideways and up and down acceleration which is taken into account when determining the acceleration magnitude, introducing an error. Fortunately this error is zero on average so it will only affect temporary measurements and not the overall integration.

GPS has a bounded error since it is measuring absolute positions and the noise factors can be determined. The maximum location accuracy that can be determined using high scale GPS receivers is 6.8 meters according to [7]. The accelerometer on the other hand, has a very high resolution and can thereby measure very small changes in speed. However, when integrating the accelerometer data to obtain a distance, the noise causes the error to grow in time without any bounds. Due to the complementary error behaviour a combination of the data of these sensors should give a higher accuracy output. Sensor fusion is applied to combine the measurements made by both sensors into an output state which describes the boat's along-track position and velocity.

The term *fusion* refers to the process of combining the two datasets to produce a better output. The accelerometer is sampled at 50 Hz while GPS measurements are received at 1 Hz. The fusion algorithm needs to take this multi-rate input into account. To overcome this problem the GPS data could be extrapolated at 100 Hz using previous GPS readings. Another possibility is to apply the accelerometer data to the current speed and use the outcome value whenever a new GPS reading is received. This latter option is chosen due to its simplicity and no need for higher data rates than 1 Hz. This lower rate also simplifies network communication and decreases the load on the processor.

In general there are two types of architectures of data fusion: tightly coupled and loosely coupled systems [33]. In a tightly coupled integration a specially designed Kalman filter, that models the GPS geometry and the accelerometer error, could be used to directly correct the accelerometer error with the data from the GPS measurements. This architecture uses the often only internally available pseudorange measurements which represent a distance estimate from the GPS receiver to one satellite before being error corrected. Pseudoranges are almost always also preprocessed by a Kalman filter in the GPS receiver to output a geographic position and velocity. A loosely coupled integration technique uses these internally preprocessed pseudorange measurements together with integrated accelerometer measurements to estimate the actual position and velocity. In this case two independent solutions for position and velocity are combined into an output. This latter form of coupling is used in the case of the rowing tracker since cheaply available GPS sensors only output derived position and velocity and not its internal pseudoranges. Pseudorange reporting GPS sensors are available, but for at least five times the price of the GPS sensor used here.

A Kalman filter is a recursive data processing algorithm that can estimate the variables of a system. The algorithm uses series of measurements taken over time, containing noise and measurement inaccuracies to produce a statistically optimal estimate of the underlying system state. It can be theoretically shown that the filter minimizes the variance of the estimation error. Because of the low number of variables and simple calculations, Kalman filters are often implemented in embedded control systems, where an accurate estimate of the process variables is important. In this case a Linear Kalman filter (hereafter LKF) is used to produce the optimal state estimate of the boat. This state estimate contains the along-track position and the speed of the boat, given the set of GPS and accelerometer measurements and the relationships between the measurements and the state vector. The state of the system is described by vector x consisting of two variables: directional speed in meters per second $\vec{v}$ and the along-track position in meters $\overline{tb}$.

The Kalman filter works on a prediction-correction basis [6]. First a *prior belief* in a certain state estimate is computed by making a prediction based on the dynamics of the system. Using measurements of the system, the prediction is corrected and a *posterior belief* is calculated. The prior belief is the probability of being at state x_k given all the measurements z up to step k. The posterior belief is the probability of being at state x_k given all the measurements z up to and including step k. For an extensive explanation about Kalman filters [6] can be consulted.

Every time step the LKF calculates a new prior belief for the state of the system, which can be seen as a prediction. Due to noise and measurement inaccuracies, there may be a prediction error, causing the predicted state to be different from the true state. New measurements give information about the true state of the system and are used by the LKF to correct the predicted state. For the prediction the LKF uses a system model which describes how the true state of the system evolves over time. Every time step the system uses propagation to predict the new state of the system, by projecting forward the most recent belief in the prediction step:

$$\hat{x}_k^- = A\hat{x}_{k-1}^+$$

$$P_k^- = AP_{k-1}^+A^T + Q_{k-1}.$$

Matrix A is an $n \times n$ matrix which describes the propagation of the state at time step $k-1$ to the current state at time step k. P_k^- is the prior error covariance at time step k, which together with the prior state estimate $\hat{x}_k^-$ characterizes the Gaussian prior belief. Q_k contains the covariance values of the process noise.

During the correction step, measurements are used to update the belief with information gained from these measurements:

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$P_k^+ = (I - K_kH)P_k^-$$

in which

$$K_k = P_k^- H^T(HP_k^- H^T + R_k)^{-1}$$

The first equation describes how the measurement data z_k is used to correct the prior state estimate $\hat{x}_k^-$. Matrix H relates the current state x_k to the measurement values, so that the difference between the prior state estimate and the measurement data can be calculated. K_k is called the *Kalman Gain* and determines the factor to which the difference between measurement z_k and the measurement prediction $H\hat{x}_k^-$ is taken into account in the posterior state estimate $\hat{x}_k^+$. The posterior error covariance P_k^+ is calculated from the prior error covariance P_k^- by applying the same Kalman Gain K_k. R_k contains the covariance values of the noise in the measurements.

It is assumed in an LKF that all errors are zero-mean random noise which is Gaussian distributed. On average this type of noise is zero, but at certain times it can fluctuate around this average.

For the rowing tracking purpose the along-track position of the boat and the speed are of interest. The GPS output data is first processed according to the method described in section 5.1 to obtain the along-track position $\overline{tb}$. Furthermore the velocity and course data from the GPS measurements are projected on the track to obtain the speed in the direction of the track:

$$\vec{v} = \cos(\phi - \tau)v,$$

in which ϕ describes the course in radians given by the GPS sensor, τ is the direction of the track in radians, v is the velocity given by the GPS module and $\vec{v}$ is the calculated speed in the direction of the track.

Initialization of the LKF is done with data from the first received GPS reading. The calculated values of $\overline{tb}$ and $\vec{v}$ are set in the posterior state estimate $\hat{x}_0^+$ and the initial uncertainty P_0^+ is set to 0.5 for both along-track position and speed.

Every time a new accelerometer measurement is made, this data is incorporated into the prior state estimate $\hat{x}_k^-$. The acceleration data is put through the invariant orientation filter as described in section 4.3, which results in the acceleration in the direction of the boat's movement $\vec{a}$. The prior directional speed is updated using the measured acceleration data using the uniform acceleration formula:

$$\vec{v}_k^- = \vec{v}_{k-1}^- + \vec{a}_k t,$$

where t is the time that the acceleration is applied, in this case 0.02 second, since measurements are taken at 50 Hz. The prior along-track position is estimated using the prior directional speed and the formula for displacement:

$$\overline{tb}_k^- = \overline{tb}_{k-1}^- + \vec{v}_k^- t,$$

in which t is again 0.02 seconds. The prior state estimate thereby uses the current accelerometer data to make a prediction of the state of the system, instead of only using past data.

Since the accelerometer data has some noise in the measurements and the measurements are relative to each other and have to be applied recursively, the error of the prediction will grow over time. The GPS module outputs absolute data that can be used to correct this error. When a new GPS measurement arrives, the Kalman Gain is determined and the LKF corrective step is applied. The posterior state estimate $\hat{x}_k^+$ is calculated in the correction step. This state estimate now holds information from both the accelerometer, since the prior state estimate is taken into account, and information from the GPS, since this is used to correct the prior state estimate. Just after the calculation of the posterior state estimate, the new prior state estimate $\hat{x}_k^-$ is set:

$$\hat{x}_k^- = \hat{x}_{k-1}^+.$$

The same value is used for the new prior state estimate in order to use the posterior state estimate and apply the recursive accelerometer update onto the latest best known state estimate.

The LKF is a practical implementation, since in the recursive steps, all the measured data is incorporated into the state estimate. Nevertheless only a small number of variables need to be kept, which makes it a memory efficient algorithm. The primary interest now lies with the parameters used that affect the performance of the filter. Q_k contains the process noise covariance at time step k and R_k the measurement noise covariance at time step k. Since it is assumed that the noises are independent, only the main diagonal of the covariance matrices Q_k and R_k contain the variance in the state and measurement vector variables respectively, and the off-diagonal elements are zero. A common problem when designing Kalman filters is the lack of a description of the noises that affect the states and the measurements [34]. Available recorded data can then be used to estimate the parameters.

In general there are two types of methods that can be used to construct a Kalman filter using test data if the covariance matrices Q and R are unknown [34]. The first type of methods parameterizes the matrices Q and R themselves and estimates their values using test data. The second type of methods uses the fact that if corrections are made with a regular interval the Kalman Gain converges to a single value. These methods therefore estimate the Kalman Gain K directly from the data. Since Q and R only directly affect K, when the latter is parameterized, less variables have to be estimated. This disadvantage of this method is that no information is gained about the covariance in the noise of the states and measurements.

Because the LKF is processing data for two variables, namely the along-track position $\overline{tb}$ and the velocity $\vec{v}$, K has two values $K_{\overline{tb}}$ and $K_{\vec{v}}$ respectively. Because of the way K is calculated, its values always have to be between 0 and 1. The values of K are optimized using a variant of the 2-fold cross-validation technique. In normal 2-fold cross-validation all data points are split into two random sets d_0 and d_1, both of equal size. Training is done on d_0 and the trained algorithm is then tested on d_1, which is repeated the other way around so that both sets are used once for training and once for testing. This is used in machine learning to overcome the problem of over-fitting. It has the advantage that machine learning can be used when little data is available since both sets are large and the full set is used for both training and validation. In this case the algorithm is not really 'trained' but merely optimized to use the correct parameters. If this optimization is done on the full dataset at once, there is a high chance of over-fitting, in which case the parameters are set exactly so that the outcome is optimal for this dataset, but might perform very bad on other datasets. Therefore the optimization is done on two randomly splitted subsets. If the optimization outcome is the same for both sets, this would yield an unbiased optimal result which has a high chance of the same performance on other datasets.

To optimize the values of K the error landscape is plotted for both subsets with both parameters in K varying between 0–1 with an interval of 0.1. The value zero actually is chosen to be 0.01, since a value of zero would result in no propagation of the corrective updating measurement from the GPS. The error landscapes are shown in figure 17, where the x axis represents the value of $K_{\overline{tb}}$, the y axis the value of $K_{\vec{v}}$ and the z axis shows the average error in seconds of the Kalman filter timing results on the two subsets of five independent races. It can be seen that the average error is between 0.35 and 0.7 seconds for almost any K. Only around the edges where either $K_{\overline{tb}}$ or $K_{\vec{v}}$ is 0.01 the error varies a lot. When

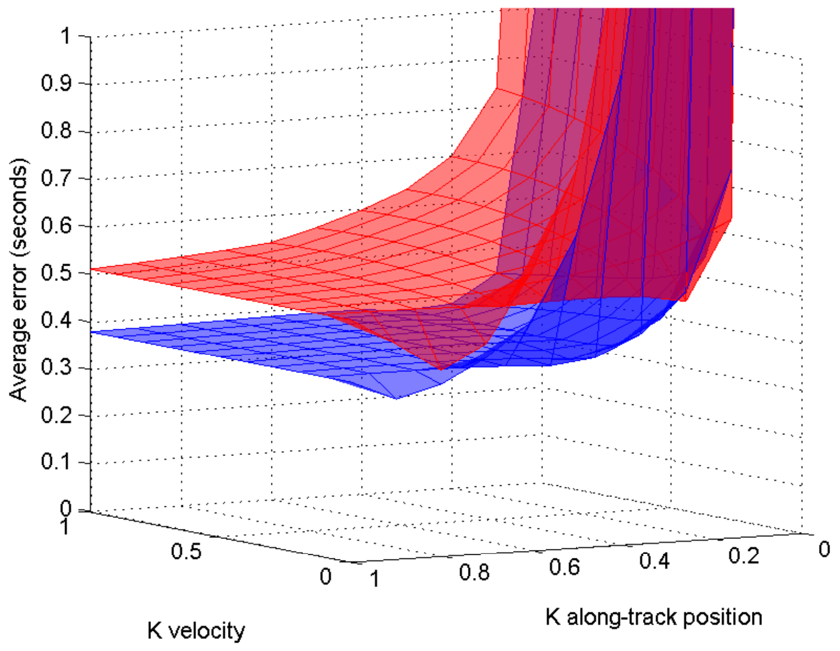


Figure 17: Two Error Landscapes of Data Subsets, each with Five Independent Races, with Varying K

$K_{\overline{tb}}$ is 0.01 the LKF almost only uses the along-track distance as determined by the accelerometer. Since noise on the accelerometer measurements causes a large error after being integrated twice, the error on the value of $\overline{tb}$ grows, which results in a big error when only the accelerometer data is used. On the contrary it seems that the accelerometer data is able to correctly predict the velocity, since when varying $K_{\vec{v}}$ the sum of error does not vary much. In both landscapes there is a small valley around $K_{\overline{tb}} = 0.8$ (red subset), $K_{\overline{tb}} = 0.9$ (blue subset) and $K_{\vec{v}} = 0.01$. With these values, the along-track position as determined by the GPS sensor is taken into account with a factor of 0.8 or 0.9 respectively, while the along-track position as determined by the velocity with incorporated accelerometer information is taken into account with a factor of 0.2 or 0.1 respectively. The velocity is almost only determined by the accelerometer data and the initial GPS reading. With these values of K an average error can be obtained which is smaller than the average error when only using GPS data ($K_{\overline{tb}} = 1$). Since one optimal parameter setting of K is the optimization goal, the optimal values for K for the two subsets are averaged and taken as the optimal value: $K_{\overline{tb}} = 0.85$ and $K_{\vec{v}} = 0.01$.

The results of the LKF predicting times for ten independent races with $K_{\overline{tb}} = 0.85$ and $K_{\vec{v}} = 0.01$ are shown in table 5. Four of the races which were used for GPS results had to be excluded from the LKF results, since accelerometer measurements were not correctly logged as explained in section 3.4.1. The evaluation is done in the same manner as evaluation was done for GPS-only timing results as described in section 5.1.3. It can be seen that all LKF results lie within a one second difference from the conventional timed results. The average error is 00:00,38 seconds and the standard deviation is 00:00,31 seconds. There is no apparent indication that the LKF timed results are either always too slow or always too fast.

An improvement over the GPS timed results from section 5.1.3 is made and shown in the last column of table 5. A total improvement of 00:00,63 seconds is achieved over ten races using the LKF. In only three out of ten cases the LKF performs worse, and in seven cases the LKF performs better than the GPS-only solution. The average improvement is 00:00,06 seconds which in absolute sense is not much, but yields a 14% improvement over the average error of 00:00,45 seconds which was obtained using GPS only on the same datasets. These results show that an implementation of an LKF for tracking boat's position using the combined information from GPS and accelerometer is beneficial.

Table 5: LKF Distance Results of 10 Independent Races

Conventional Timed	LKF Timed	Difference	Improvement over GPS Timed
06:13,74	06:13,38	-00:00,36	-00:00,16
07:00,00	07:00,96	00:00,96	00:00,15
07:17,09	07:17,02	-00:00,07	00:00,17
07:28,41	07:28,88	00:00,47	00:00,13
07:30,86	07:30,84	-00:00,02	00:00,04
07:35,66	07:35,84	00:00,18	00:00,20
07:38,84	07:39,01	00:00,17	00:00,27
07:53,84	07:54,40	00:00,56	00:00,07
08:25,91	08:26,15	00:00,24	-00:00,02
08:38,13	08:37,34	-00:00,79	-00:00,23

6 Discussion & Future Work

In this thesis different methods are described which can be used to implement a rowing race tracker using GPS and accelerometer data. The data obtained by these sensors can be used to extract different features of rowing including as shown, stroke rate, along-track position and speed. In the introduction the following research questions were asked which would lead to a rowing tracking solution:

1. How can data be reliably transferred by roving nodes over a distance of two kilometers using easy available and cheap ZigBee radio modules? What is the performance of many-to-one mesh networking using roving XBees?

2. Which algorithm is simple enough to run on easy available and cheap hardware and is still reliable in the detection of rowing strokes in accelerometer data? Can this algorithm be extended in a way that accelerometer orientation will not influence the algorithm's outcome?

3. How can the relative distance from the starting position be calculated from GPS recordings? What is the accuracy of this data? What are the improvements when using a cheap static reference receiver to reduce omnipresent noise?

4. How can the accuracy of the distance-travelled data be improved by fusing GPS and accelerometer data streams?

The tests that were performed and the results found in section 3.5 show that reliable long-distance communication can be achieved using XBee communication modules. Although direct transmission of data using these modules can overcome distances of 1.5 kilometers, small obstructions can cause the connection to drop. A reliable communication was obtained over 500 meters even while sometimes having an obstruction. While it was not sure if the transmission power is high enough to pass the obstructions, or the retry and acknowledgment structure of the ZigBee protocol makes sure that data arrives, no packet loss was seen when using these distances. In future work this could be examined by looking at the ZigBee transmit status packet which contains data about the number of application layer transmission retries that took place.

Many-to-one routing is a very practical solution in wireless sensor networks which need a single central data collector. Instead of spending significant overhead on route discovery, a single many-to-one broadcast transmission is sent from the data collector to all nodes, creating reverse routes on all devices. By setting the time interval for these many-to-one broadcast transmissions to a value smaller than the time needed to pass by a new router node along the track, the transmission route is updated as nodes are moving, which makes sure that a route to the data collector is always available. Although the tests prove that this feature works with a single node passing multiple routers, it could be investigated what the performance is when using 80–100 nodes which are needed to live track a full regatta. It is thought that especially in the case of this many end-devices the performance increase of many-to-one routing over normal mesh routing could be observed.

Two different stroke detection algorithms have been compared in section 4.4. It is shown that the deceleration peak during the catch can be used to detect a single stroke. The exact pattern of acceleration differs a little from stroke to stroke and therefore causes this peak to shift a bit in time. When using simple maximum peak detection this shift causes variations in the calculated stroke per minute rate. To overcome this, a method is proposed which averages the times at which the acceleration passed the threshold values. By doing so a more smoothed value for the SPM is achieved.

The difference in acceleration power between for example a single scull and an eight results in problems when using a static threshold. In this case a relative high threshold causes peaks to be missed for a single scull, but a relative low threshold causes noise in the eight to result in false peak detection. A dynamic threshold is proposed to solve this problem. It is shown that stroke rates can be detected with the same accuracy as when manually counting strokes. It even appears that the proposed algorithm performs better than a commercially available stroke counting solution which is available for $ 399, at least three times the price of the solution used here.

The invariant orientation filter as proposed in section 4.3 makes sure that the data used for stroke detection is not affected by the orientation of the accelerometer. Instead of using a computationally complex algorithm, a simple solution is presented which achieves similar performance on randomly rotated datasets. By using this computational simple solution, the full process of stroke detection from rotated accelerometer data can be performed on cheap and easily available microprocessors. This results in an inexpensive onboard device that could be used for live rowing tracking.

In section 5.1 a method is presented which can be used to calculate the along-track travelled distance since the start of a race, by using GPS coordinates. It becomes clear that for these calculations the curvature of the Earth need to be carefully taken into account, since simple calculations based on spheres lead to errors up to 6 meters on 2000 meter rowing tracks. The accuracy of the along-track distance data is examined in section 5.1.3 by comparing the results to a conventional timing system in use during national regattas. The obtained results prove to be very accurate, which can somewhat be attributed to the use of the relative along-track position measurement which minimizes the absolute error.

The use of a cheap static reference receiver to reduce reception noise is investigated in section 5.2.1. Unfortunately the output limitations of the cheap GPS receivers raise problems in the extraction of the noise. Since the exact measurements of the distances to each satellite (pseudonumbers) are only available internally in the receiver, the deviation from the known position of the static receiver for each of these satellites cannot be calculated. Furthermore every receiver could potentially be using a different set of satellites to determine its position, and thereby would receive a different error in the positional output. For future work it could be investigated if it is possible to have cheap receivers output these pseudonumbers. If this is possible, it is certainly a promising method of improvement since the raw GPS output is already capable of predicting finish times with only a 00:00,53 second error. The question then is if the higher price for a pseudonumber outputting GPS receivers is worth the small potential gain.

The Kalman filter used for fusing GPS and accelerometer information into an accurate state estimate is described in section 5.2.2. The two estimated parameters are the boat's along-track position and its speed. Every time step both these parameters are estimated using the data from the accelerometer. The accelerometer measures the acceleration of the boat. By using the uniform acceleration formula the acceleration can be used to calculate the current boat speed from the last known speed and the measured acceleration. Then by integrating the speed over time the distance travelled since the last measurement is obtained. Both these values are then good estimations of the current boat's state. However, since these estimations are only based on relative measurements, existing noise on the measurements will cause the error on these estimations to grow over time. The GPS measurements, in contrary, contain an absolute speed and position. Although there is also an error in the GPS measurements, this error does not grow over time. Due to the complementary error behaviour of the two types of information, it is possible to integrate this in a more accurate state estimate. The Kalman filter is tuned using the available data and an improvement of 14% over the GPS-only results is obtained.

Although an improvement of 14% seems good, the absolute improvement on a single race is on average only 00:00,06 seconds. Since GPS calculations are offloaded to the central server because of the computational expensive Vincenty's formula, the Kalman filter also has to be calculated on the server. The application of a Kalman filter thus yields that the accelerometer data has to be sent over the network to the data collector. Because accelerometer measurement are taken at 50Hz, this would result in a major increase of network traffic. Even if the measurements are stored in a local buffer and sent to the data collector together with the GPS measurements, this would result in at least twice the number of data packets, since 50 2-byte values are stored, and the packets can only contain up to about 80 bytes of data (variable because of the exact route). It could be investigated if it is possible to either perform Vincenty's formula and the Kalman filter on the microprocessor, or if there is a way of compressing the data so that the computations can be completely offloaded. The latter seems most feasible and in a simple application the accelerometer measurements could be averaged over time and sent as a single value together with the GPS data. However for the use in a live rowing race tracker the improvement of 00:00,06 seconds would not be worth the extra computations for the Kalman filter and the extra network load which also reduces battery life. Even when using Kalman filter fusion there is still a 00:00,31 second standard deviation and the improvement would be dwarfed by this variation.

6.1 Hardware

During the final stage of the research, some potential improving hardware parts were brought to attention. First on September 27 (three days *after* the colloquium), a Kickstarter campaign funded the creation of the Flutter wireless ARM development board. The Flutter is a board that combines a microcontroller and a communication module into a single device. It is claimed to have over 1 kilometer range and provides 256-bit AES encryption for security. Flutter will be published as open source hardware and software, and would therefore be ideal for the creation of a customized PCB containing an additional accelerometer and a GPS module. A basic version of Flutter will be available for \$ 20 and the pro version for \$ 30. While it can not be deduced conclusively from the present data if the basic or pro version is needed, a saving of over € 30 can easily be achieved, since the Arduino Fio and XBee could be discarded.

Furthermore another GPS module was found which states to have a higher sensitivity and more location updates per second for roughly the same price. This PA6H (MTK3339) GPS module could be tested to see if higher accuracy results are feasible. Nevertheless this module is smaller in size and has a lower power consumption, so even while performing the same as the current module this module would improve the onboard device.

Lastly the accelerometer could be improved upon by using a different chip. Instead of using the MMA8452 the microprocessor could be relieved in its time critical task of handling the accelerometer data, by using for example the MMA8450 (12-bit) or the MMA8451 (14-bit), which have an embedded FIFO buffer of 32 samples. The number of I^2C transactions could be minimized and this would allow the microprocessor to spend 31 sample-durations (at 50 Hz: 0.62 second) on other tasks while only reading the data once every 32 samples. The freed processor time could for example be used to perform Vincenty's formula and the Kalman filter calculations on the microcontroller. Furthermore both these accelerometers have an embedded orientation detection feature which could be used for the orientation invariant measurements. The problem is that both the MMA8450 and the MMA8451 are not available on a breakout board. When designing a customized PCB this would not matter anymore as the accelerometer could be placed on the main PCB.

6.2 False Start Detection

An additional feature that could be implemented using the data already available is false start detection. According to Rule 74 in the FISA's Rules of Racing 2013 [35], a false start is made when a crew's rowers begin rowing before the start command is given. Since rowing will cause an acceleration of the boat, the moment when the rowing starts should be easily extractable from the accelerometer's data. The last received GPS measurement can be used to determine the time in seconds precision of the start of rowing. The number of accelerometer measurements since this last received GPS measurement can be counted and multiplied with the sampling interval (0.02 seconds at 50 Hz) to get the sub-second precision. Then the time of the start of rowing can be compared to the time stamp made by the start command. When the rowing started before the time stamp of the start command, a false start can be signalled.

The accelerometer invariant orientation measurements could be used to perform the false start detection on. A simple threshold on the magnitude of the measurement could be used to determine if a boat's crew is rowing or not. Every time a crew starts rowing a time stamp signal can be sent to the data collector which determines if the start command for the race of the specific crew is given just before or after that start of rowing. If this is so, the signal's time stamp can be used to determine a possible false start. It furthermore needs to be taken into account that during every stroke the magnitude of the invariant orientation measurements passes zero. When simply signalling 'the start of rowing' every time this threshold is passed, many false detected rowing starts are signalled, namely during each stroke. This could be overcome by using a well chosen duration during which the signal has to be below the threshold, to change the boat's state from rowing to resting, after which a new start of rowing can be signalled again.

References

[1] Koninklijke Nederlandse Roeibond. 2013. *"Reglement voor Roeiwedstrijden."* Retrieved from http://www.knrb.nl/content.php/nl/621

[2] Hamilton, C. Sampath, V. 2010. *"Performance of ZigBee PRO Mesh Networks with Moving Nodes."* Unpublished manuscript, Texas A&M University, Texas, United States.

[3] Palshikar, G. 2009. *"Simple Algorithms for Peak Detection in Time-Series."* In Proc. 1st Int. Conf. Advanced Data Analysis, Business Analytics and Intelligence.

[4] Rizos, C. 2003. *"Trends in GPS Technology & Applications."* In 2nd International LBS Workshop.

[5] Kee, C., Parkinson, B., Axelrad, P. 1991. *"Wide area differential GPS."* Navigation, 38(2), 123-146.

[6] Negenborn R. 2003. *"Robot localization and kalman filters."*, Master's thesis, Inst. Inf. Comput. Sci., Utrecht Univ., Utrecht, The Netherlands.

[7] Satirapod, C., Rizos, C., Wang, J. 2001. *"GPS single point positioning with SA off: How accurate can we get?"* Survey Review, 36(282), 255-262.

[8] Liu, H., Pang, G. 2001. *"Accelerometer for mobile robot positioning."* Industry Applications, IEEE Transactions on, 37(3), 812-819.

[9] Mazl, R., Preucil, L. 2003. *"Sensor data fusion for inertial navigation of trains in GPS-dark areas."* In Intelligent Vehicles Symposium, 2003. Proceedings. IEEE (pp. 345-350). IEEE.

[10] Carminati, M., Ferrari, G., Grassetti, R., Sampietro, M. 2012. *"Real-Time Data Fusion and MEMS Sensors Fault Detection in an Aircraft Emergency Attitude Unit Based on Kalman Filtering."* Sensors Journal, IEEE, 12(10), 2984-2992.

[11] Kleshnev, V. 1999. *"Propulsive efficiency of rowing."* In Proceedings of the XVII International Symposium on Biomechanics in Sports (pp. 224-228).

[12] Tessendorf, B., Gravenhorst, F., Arnrich, B., Troster, G. 2011. *"An IMU-based sensor network to continuously monitor rowing technique on the water."* In Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2011 Seventh International Conference on (pp. 253-258). IEEE.

[13] Nielsen Kellerman. 2009. *"Surge Rate Technology."* Retrieved from http://nkhome.com/news/2009/11/surge-rate-technology/

[14] Tshalif. 2011. *"Talos Rowing"* [android software]. Available from http://nargila.org/trac/robostroke/wiki

[15] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., Cayirci, E. 2002. *"Wireless sensor networks: A survey."* Computer networks, 38(4), 393-422.

[16] Arduino. *"Arduino."* Retrieved from http://www.arduino.cc

[17] Arduino. *"SoftwareSerial Library"* Retrieved from http://arduino.cc/en/Reference/SoftwareSerial

[18] Arduino. *"Wire library."* Retrieved from http://arduino.cc/en/Reference/Wire

[19] Arduino. *"SPI library."* Retrieved from http://arduino.cc/en/Reference/SPI

[20] Hart, M. 2013. *"TinyGPS - A Compact Arduino GPS/NMEA Parser."* Retrieved from http://arduiniana.org/libraries/tinygps/

[21] Arduino. *"SD library."* Retrieved from http://arduino.cc/en/Reference/SD

[22] Gerlach-Meyer, U. E. 1991. *"Micromachined capacitive accelerometer."* Sensors and Actuators A: Physical, 27(1), 555-558.

[23] PJRC. *"AltSoftSerial Library"* Retrieved from http://www.pjrc.com/teensy/td_libs_AltSoftSerial.html

[24] Jang, W. S., Skibniewski, M. J. 2008. *"A wireless network system for automated tracking of construction materials on project sites."* Journal of civil engineering and management, 14(1), 11-19.

[25] Kadar, P. 2009. *"ZigBee controls the household appliances."* International Conference on Intelligent Engineering Systems, 2009, pp. 189-192.

[26] Ruiz-Garcia, L., Barreiro, P., Robla, J. I. 2008. *"Performance of ZigBee-based wireless sensor nodes for real-time monitoring of fruit logistics."* Journal of Food Engineering, 87(3), 405-415.

[27] Digi International. 2012. *"XBee/XBee-PRO ZB RF Modules."* User Manual.

[28] Kleshnev, V. 2010. *"Boat acceleration, temporal structure of the stroke cycle, and effectiveness in rowing."* Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology, 224(1), 63-74.

[29] Mizell, D. 2003. *"Using Gravity to Estimate Accelerometer Orientation."*. In Proceedings of the Seventh IEEE International Symposium on Wearable Computers (ISWC'03) (Vol. 1530, No. 0811/03, pp. 17-00).

[30] Kunze, K., Lukowicz, P., Partridge, K., Begole, B. 2009. *"Which Way Am I Facing: Inferring Horizontal Device Orientation from an Accelerometer Signal"* In Wearable Computers, 2009. ISWC'09. International Symposium on (pp. 149-150). IEEE.

[31] Veness, C. 2010. *"Calculate distance and bearing between two latitude/longitude points using Haversine formula in javascript."* Retrieved from http://www.movable-type.co.uk/scripts/latlong.html

[32] Boyle, M. J. 1987. *"Department of Defense World Geodetic System 1984-It's definition and relationship with local geodetic systems."*, DMA Technical Report 83502.2. Washington, DC

[33] Mayhew, D. M. 1999. *"Multi-rate sensor fusion for GPS navigation using Kalman Filtering."* PhD Thesis, Department of Electrical Engineering, Virginia Polytechnic Institute and State University.

[34] Bos, R., Bombois, X., Van den Hof, P. M. J. 2005. *"Designing a Kalman filter when no noise covariance information is available."* In Proceedings of the 16th IFAC World Congress, Prague, Czech Republic.

[35] Fédération Internationale des Sociétés d'Aviron. 2013. *"Rules of Racing."* Retrieved from `http://www.worldrowing.com/fisa/resources/rule-books`