PlayTrack: An Indoor Location Based Player Tracking System.

Warwick Bayne, David Beath, John Harborne and Mark Vause
School Of Information Technology
Otago Polytechnic
Forth Street Campus, Dunedin
0800-762-786
autonomousmovement.op@gmail.com

ABSTRACT
Indoor location tracking of sports players is a problem that has yet to be satisfactorily solved. The aim of PlayTrack is to take the benefits of outdoor location tracking and bring it to the indoor area. This is done by using wireless radio frequency devices and networking technology to allow for accurate real-time location tracking, and to then provide this data to the coaches in an easy to use application. In this paper we highlight the design context that we were operating in, the decisions that led us toward the system we implemented, and an overview of the final system design.

1. INTRODUCTION
This project has successfully demonstrated that a trilateration based signal strength network can position multiple tags within an indoor area with an accuracy of 5m. This paper first describes the background of this topic then describes the artifacts of testing.

2. DESIGN CONTEXT
The goal of this work is to track the players of Ice Hockey and collect data on their movement, orientation and distance travelled throughout a game and allow the coach to review the data and then improve their player's tactics. Player tracking means creating statistics about sports players as they play a game in real-time. The options we considered include video, ultrasound or signal strength, or a hybrid approach, and these are all considered in the following sections.

2.1 Sports Analytics and How They Can Be Improved With Technology.
Many systems that are currently used by professional teams use a video replay of each "play" or event that happens during a game. With multiple cameras arrayed around a field, the analysis team can then show the players an event from different angles. This helps them to analyze the decisions made by each player. With this analysis the player can then improve the way they make future decisions. An example of this type of system is the TryMaker Pro system created by Versuco Sports (Verusco, 2012). This system uses any available video feed of a game and segments it into parts that are categorised and fed into a database. Once this is done the staff are able to play back the footage frame by frame and view all of the angles of the action to gain a better understanding of what happened (Verusco, 2012).

The results gained from the analysis of this can be then be used by coaches and players to predict the outcome of future events. Ganeshapillai & Guttag were able to analyse previous performances of baseball pitchers. This analysis is then used to help the batters who are opposing them. With this knowledge the batter can then anticipate how the next pitch will happen. Another example of decision making assistance is with rebounds in NBA basketball. Over past seasons and games, various details of how a shot was taken are collected and analysed. Examples of the information collected include location of shot, location of rebound, where the ball was before the shot was taken. This information is then combined to work out the best location to take the shot for the attacking team. This also tells the defensive team were to defend, where a rebound is likely to go and where the attacking team will be positioning their players.

Analysis of sport is highly dependent on the sport that is being analysed. As each sport is different there is always a different area to concentrate on. There are four main types of sports: period based, set point, time and performance based (Tjondronegoro & Chen, 2008). Tjondronegoro & Chen (Tjondronegoro & Chen, 2008) have said that hockey is a period based sport.

The main sport we are looking at for our project is Ice Hockey. Ice Hockey consists of two linesmen and a referee as well as six players on two teams skating around a rink trying to score goals with a rubber puck. The game is fast paced with players attaining speeds of 32 KM/H (Humprey, 2012) and the puck can travel up to 160 km ("Science of Hockey: Shooting the Puck") . Each game is made up of three periods of 20 minutes each. Statistics are mainly kept about goalies. Useful statistics that are kept by teams include the amount of saves, save to goals against ratio. These help the coaches in selecting goalies.

Athletes that play in other positions have different roles. There are two forwards and two backs. Making up the rest of the team is a centre and a goalie. The defense's main job is to stop the opposing team scoring goals. They do this by trying to gain the puck from the opposition and "checking" (K. Wood, personal communication, March 15, 2012) the other team. Checking is when another player intentionally hits another player to distract and impede the other teams progress (K. Wood, personal communication, March 15, 2012).
The offensive team's main job is to score goals. They do this by stealing the puck from the other team and dodging the other team's defensive side (K. Wood, personal communication, March 15, 2012).
The main areas of analytics that Ice Hockey coaches look at are the areas of tactics and the efficiency of how each player skates

across the ice (K. Wood, personal communication, March 15, 2012). Skating across the ice has been analyzed to be made up of two main areas two foot gliding and struggling for the puck (Bracko, 2004).

Previous efforts with computer vision in Ice Hockey have looked at identifying when shots have been scored from broadcast footage of a game (Wang & Zhang, 2009). This research has used a Hidden Markov Model to empirically learn what a shot looks like. This model is then used to identify which shots have been scored. once a shot has been identified a section of the video around the event so it can be quickly analyzed after the game (Wang & Zhang, 2009).

User enjoyment and perspectives are key to the success of a system in this area. This is because “many individuals have become more comfortable with biometric measurements; many still retain negative perceptions of the technology.”(Pons & Polak, 2008). Accordingly systems must be “easy to use and “transparent” to end users”(Pons & Polak, 2008).

There are many different methods to use cameras to track people and many different issues that need to be solved with camera tracking technology. This problem has been addressed before and there are some very good open source libraries that are of use.

There are two main open source library's that are used in computer vision. These libraries will be useful to be able to find and track players when they move around the hockey rink. The first of these is OpenCV which runs in C++ and the second is Aforge.net which runs in C#.

OpenCV (Open source Computer Vision) is an open source library that can be used to take images from a camera and find shapes or colours by filtering the image as shown in (Sinha, 2010). This can be done in a number of ways including thresholding and filtering. To be able to find an object of a particular colour spectrum, image thresholding is used where a colour scheme is isolated, for example red, after this the other colours are removed from the image. This leaves an image that highlights the red scale. This image is then run through a binary filter which turns it black and white. The white areas show where all the red is in an image. After the black and white image is created, a filter is then applied to select the appropriately shaped object and the coordinates are stored for future use.

Aforge.net is another open source image processing library which works more on image processing then actual object finding. Aforge.net has a very good ability where it can take a single colour image (gray-scale) and find the blobs (the biggest objects of the same color). Which leads to the problem of how do to find objects of multiple colors. To do this, more than one filter is used on the source image and then the results are matched. Unfortunately Aforge.net is not a very efficient library which takes a quite noticeable time to process the images per camera cycle.

There are three main issues with finding a person on a camera. The first is how to tell what is a person and what isn't. The second is how to find a person after they have left the frame and then reentered the field of view. Another issue with finding people is identifying which people have moved to a different camera.

One of the hardest parts of finding people in images is teaching the computer what is a person and what isn't and this can be done is several ways. the first is to teach the computer what the space looks like when there are no people in it and get it to tracks parts of images that don't match as shown in this article (AForge.Net Framework Features- Motion Detection). This can lead to issues with the space changing i.e. someone putting a box in the hallway or moving the camera. There are other ways to dynamical find people (Gray & Tao, 2008), by seeing what has changed from the last group of images using positioning histograms and various shape patterns to detect body shapes and movement. Then a database is used to remember the shape of that person to speedup detection next time they enter the field of view.
Tracking people in different areas at different times is another issue faced by computer vision. The ability to pick someone up when they leave the view of one camera and enter the view of another camera is a difficult problem as a most systems think they are different people. have presented a case where surveillance in an airport but similar methods can be will be viable. An ice hockey rink, where the field is quite large, we will need multiple cameras to cover it all. Douglas Gray and Hai Tao (Gray & Tao, 2008) describe ways of using people which in low population areas, with 3 -5 people at a time using basic calculations based on location and velocity however when there are more people, the risk of losing track of them intensifies.

2.2 Embedded Tracking Systems And Their Use In The Sport Industry

The last few years have seen the emergence of positioning or location aware technologies being used in the sport industry to allow the tracking of player’s locations and assessing the movements they make at a later date. The positioning techniques can be used in two main ways, the first method is self-positioning.

Self-Positioning is used by the target device using signals that are transmitted by the gateways or antennas which can either be terrestrial (employed on the land mass itself) or by satellite and then uses the information gathered from the signals to calculate its position by making measurements from the distributed transmitters and determines it position from this information. Self-positioning receivers then know where they are and applications that are running can use this information to make position based decisions which makes it useful for vehicle navigation. GPS and A-GPS use self-positioning techniques that incorporate a worldwide navigation system that consists of 24 satellites spaced equally in six orbital planes (Zeimpekis et al., 2002).

The GPS receivers can then process the signals from the satellites to compute its position in 3D by use of latitude, longitude and altitude with an accuracy of 10 meters or less. The main downside of this technology is that the receiver needs clear line of sight

with the satellites which restricts the use to outdoor applications (Zeimpekis et al., 2002)
.
A-GPS is a similar technology but it is assisted by third party vendors such as phone companies by the use of cell towers to triangulate the position of a device by using a group of the towers to assist the device in use of the satellites and position finding, this method can be very accurate ranging from 1 meter up to 10 meters.

Remote positioning is a technique where a mobile terminal can be located by measuring signals travelling to and from a set of given receivers (Zeimpekis et al., 2002). The receivers which can be in a single position or a more distributed system containing multiple receivers which then measure a signal originating from, or reflecting off the object to be positioned.

A common use of remote positioning is Cell-ID or Cell of Origin (COO), this method relies on the fact that by using mobile networks you can determine the location of a mobile device by finding out which of the cell sites are currently being used by the target device. (Zeimpekis et al., 2002)
Another type of remote positioning that isn't as widely used as Cell-ID is called Direction or Angle of Arrival (AOA). This method uses a directional antenna beam from space which is moved until maximum signal strength or coherent phase is detected this measurement then produces a straight line focus, a second AOA measurement will yield a second straight line, by intersecting the two lines will give a position fix for this system however this technique on its own is inefficient at providing an accurate reading(Zeimpekis et al., 2002).

To overcome this there is the Time Delay which by using the knowledge that electromagnetic waves travel at a constant speed in free space the distance between the two points can be measured by calculating the time delay of a radio signal that it sent and the receiving of the signal, this method is commonly used in satellite systems.

There are two types of Time Delay methods that can be used with Time Delay, Time of Arrival (TOA) and Differential Time of Arrival (TDOA). Time of Arrival derives the distance of a device by measuring the time taken for a signal to make a round trip form satellite-device-satellite which by halving this result it can determine the distance the device is away from the satellite. Differential Time of Arrival uses multiple precisely synchronized transmitters which are then compared to a local base time from the signal origin to measure difference in time as it passes each of the transmitters, this can then be evaluated to give the position of the device. (Zeimpekis et al., 2002)

Existing systems are very widespread and being used readily in the world; some have been adapted for sport industry use.

SPORTVU (SportVU) is a company that has released a product called STATS which is a player tracking system being used in football, basketball and American football. Its purpose is to monitor the players and provide feedback about the distance they have travelled during a game or their location on the field during a match and many more other in game specific information and then display it graphically or presented as a stat profile for the game or individual player. This system is usable across a wide platform of devices including being broadcasted over TV, iPhone, and across the web.

Trimble has an Asset Tracking system used for larger scale tracking of Positions across multiple industries, for example tracking the position of fleets in a shipping company, shipping container tracking, hazardous materials tracking or even in the agriculture industry for tracking machinery on large scale farms. These devices once attached to the asset that is to be tracked will allow for the company to monitor the movements of single or multiple assets so that the company can assess the movements and evaluate cost and effectiveness of the company (Trimble.com).

GPSPORTS (Globally Positioning Sport) is a company which offers the GPSports SPI Pro X II which is bundled with SPI Real Time (RT) which allows the coach to track in Real Time the Distance, Speed, Heart Rate, Impact/Body Load on the players body (Muscoskeletal stress). This system is based on a coach's laptop on the side of the field and receives data from the players so the coach can make decisions based on performance. This product is in use across multiple industries of sport including Football, AFL, Rugby League, Rugby Union, Hockey, American Football and Olympic Sports (GPSports).

2.3 Design Context Conclusion

The area of sports analysis is vast with many different options to collect data about players. Our project is looking at specifically looking at tracking Ice hockey players as they move across the ice.

Throughout this literature review we have looked at several different areas, including how embedded RFID's (radio frequency ID) work, what kind of information people recording sports stats need and how users can find the positioning of people in a room with cameras.

3. OVERARCHING DESIGN DECISION

3.1 What location positioning method to use?

There are a number of ways to locate an object in 2D and 3D space which we evaluated. The positioning method we used depends both on how suitable the technique is for our particular situation, but also the hardware and technology we had access to. Our very first plan was to use a combination of Radio Frequency based positioning with Visual Tracking from multiple cameras. The Visual Tracking part was quickly dropped due to complexity and cost reasons.

We eventually settled upon trilateration using Received Signal Strength Indication (RSSI) to calculate the distance. RSSI was chosen because while it is subject to interference from other wireless devices, and can be noisy regardless, it's drawbacks were

still easier to deal with than the other methods we evaluated. It also had the advantage of being cheap and relatively easy to implement in comparison with the other methods. It was quickly obvious from our research that it is considered the most promising method for indoor location tracking.

Trilateration uses the measured distance from three or more points in order to locate an object, which is the same method used in GPS (Kaminsky, 2007). We considered using a Time-Of-Flight (TOF) measurement to calculate the distance, like in GPS, but using GPS was ruled out due to us wanting to use our system indoors. TOF was ruled out due to the distances involved being too small, as wireless signals travel at the speed of light, which is approximately 1 meter per nanosecond, thus requiring Gigahertz clocks in order to have a hope of measuring accurately.

Triangulation was ruled out as it uses measurement of angles to locate position. This is unsuitable for fast-paced measurements of multiple objects, and would require much more complicated hardware.

We also briefly considered trying using ultrasonic sensors for distance measurement, but this was ruled out due to the relatively slow propagation of sound waves, the even greater problems with interference by other objects than wireless, and an even greater lack of control over interference within the frequency range.

The following sections detail our design decisions and implementation.

4. OVERVIEW

We have developed a trilateration based networked indoor player location tracking system for use by teams and sports organizations that play games indoors and cannot therefore use GPS based tracking as outdoor sports can. By using trilateration based tracking and networked equipment we have developed a way of tracking players in real-time which can then provide feedback and statistics to the coach so that he or she can assess the situation at any point of the game. They can also review a part of the game instantly of at a later time with the team.

5. PLATFORM/TECHNOLOGY JUSTIFICATION

We chose to use Microsoft Visual Studio 2010 as the platform to develop our UI using C# as we have experience using this thus it was the logical choice for developing the user side of our application. For the back-end of the application we used Python and the Twisted module, which is an event driven networking engine for Python to allow our hardware to communicate across a network. Hardware chosen for the monitoring of the players are Arduinos with Xbee wireless devices attached to them as the mobile tags. for the stationary base stations we have used Arduino based EtherTens with PoE regulators and custom interface boards for the Xbee's. So that we can have a large distribution of hardware around a stadium we use Ethernet and a PoE Switch to power these devices, which allows for power and communication all in one.

5.1 What hardware to use?

The decision of what hardware to use both influenced and was influenced by which positioning method we used. From the beginning we had it in mind to use Arduinos as the hardware base, due to prior familiarity with them, their low cost, easy to use, extensible, open-source nature, and large community of users. This no-doubt had a huge influence on our decision process; however, it was quickly becoming apparent that RSSI was the best distance measurement tool for our purposes. The Arduino was capable of using the wireless Xbee module using the ZigBee wireless protocol, which has built in support for measuring RSSI. This is the go-to hardware combination among the Arduino community for wireless applications, and there were a number of papers positively evaluating the Xbee's usefulness in wireless positioning systems. Thus we settled upon this combination for our project.

5.2 How to deal with multiple tags?

With our system, each tag has to be able to constantly send signals to all the base stations, and each base station has to be able to receive signals from multiple tags. The default setting for Xbee devices is that they send and receive to a specified address. This would obviously be very difficult to deal with, as it would require programming all the addresses of the base stations into each tag, which would then require reprogramming if a device failed and thus an address changed. In order to fix this we set each tag to broadcast mode, so that they would indiscriminately broadcast to any listening devices. The Xbee devices all listen on the same Channel with the same Network ID, and have basic encryption built in, so this was not considered a security problem. Upon receiving a packet, the base station then reads the remote address and RSSI of the packet, along with the ID of the base station, and sends this data to the computer.

5.3 What software languages to use?

We originally decided to use C# as our main programming language as C# is a good all round language that facilitates rapid development. Later on in our project we moved the calculation engine across to using Python. The reason behind the move was because we were using the C# application to aggregate the data from all of the base stations around the rink. This was working very inefficiently and was not responsive enough for any real world applications. Once we moved the calculations to Python we decided to use the Twisted framework for communicating with the base stations over HTTP. We made this decision based on advice from lecturers and from senior Python developers.

5.4 How to deal with reflected signals?

Duplication of signals and varying RSSI values due to reflected signals was an issue that was brought up by everyone that we talked to about our project, as well as being aware of it ourselves. We determined this to not be a problem, as the ZigBee protocol account for duplicated packets. It was also obvious during testing in an enclosed area that this was not an issue, as the tags are set to

broadcast every 10 milliseconds, while the base stations are set to receive data as fast as they are physically able. We saw that we were receiving a packet consistently every 14 milliseconds, with a consistent RSSI value when the tag was not moving, which given the timing would indicate that reflections are not a problem.

5.5 How to deal with wireless spectrum interference?

The RF spectrum and available channels for the ZigBee protocol (802.15.4) and Wi-Fi (802.11b/g) overlap. Interference from Wi-Fi devices was thus a problem that we needed to address. Therefore we set the channel that we used to overlap as little as possible with the Wi-Fi spectrum. During testing in the incredibly wireless noisy environment at Polytechnic, we did not detect a noticeable interference with the signal as opposed to outdoor testing, even when using Wi-Fi devices directly next to the tags or base stations. If it did become an issue with large scale testing, then it would be possible to implement channel hopping to mitigate interference, but this would have to be balanced against the added processing required of the hardware. At this time, we do not believe this to be needed.

5.6 Where to do the data smoothing, hardware or software?

In dealing with a signal from each tag roughly every 15 milliseconds, with 6 players per side during an Ice Hockey game, there is a considerable amount of data that needs to be dealt with. RSSI being relatively noisy, we knew that the data was going to have to be smoothed considerably before being used in the trilateration calculation. In evaluating how to get the data from the base stations to the PC, if we were going to use wireless then that would be a considerable amount of data that we would have to send. In that case we would need to smooth the data on the hardware end, so that we could send a single distance value for each tag for the calculation. On the other hand, there would be a considerable amount of smoothing to be done all at once on the PC end if we sent it the raw data.

We decided against performing the smoothing on the hardware end because each Arduino only has a 16 MHz processor. Any PC would have a GHz plus processor, at least 10 times the processing power in raw clock speed of the 8 plus Arduino base stations that would be used. The PC would be much more able to perform any required calculations, and we wanted to keep the Arduino's processor free so that there would be as little delay as possible in receiving signals. At the PC end, while there is a lot of data, we thought it very unlikely that it would be too taxing for even low-end PCs, and if it was then we could always use a dedicated server with multithreading for calculations. This wasn't needed, but it's nice to have options.

5.7 How to get the data to the PC, wireless or wired?

Using a wireless connection would have the advantage of not having to string cables around the edge of the rink, making for easy setup. A wired connection would otherwise be much faster for data transfer, would not have the problem of signal interference, and would free up the base station radios to focus on receiving signals instead of having to transmit as well. The main disadvantage was that we would have to string cables around the edge of the rink.

We settled on a wired connection mainly for the lack of signal interference issues and not having to deal with transmitting from the base stations. In deciding what kind of cables to use, it was obvious that the default USB that Arduino's use doesn't have the range we need. Arduino comes in an Ethernet capable version, so we decided to use that, especially when we realised that we could use a Power over Ethernet system, and remove the requirement for the base stations to have separate power cabling.

5.8 Which end of the Ethernet connection was going to be the server?

Originally we thought of the PC end of the hardware to PC connection as the server, and so we attempted to set the base stations up as clients, and have them push their data to the server. This ran into problems when the base stations refused to push the data any faster than once a second, far too slowly for our purposes. We then realised that it actually made more sense for the base stations to act as servers, as they would always be pushing data, and never to pull data from the PC. One of our lecturers also suggested that we use the Twisted Python networking module on the PC end, which supports connecting to multiple clients, and has much more robust networking features than are easily available for C#. We found this to be a much faster and more robust solution, as the base stations would not have to worry about trying to connect to a server in addition to their other duties, and Twisted has procedures for cleanly dealing with network disconnects, and also passing data through to the calculation engine.

5.9 What format should the data be passed through as?

The formatting of data changed a number of time through the development process as other architecture decisions were decided. There are 4 data interchanges that needed to be dealt with, though that number also changed during development. Those interchanges are: tag to base station, base station to aggregator/calculation engine, calculation engine to UI, and the base station configuration settings from the UI to the aggregator/calculation engine.

The tag to base station was fairly simple to solve, we needed the ID of the tag, and the signal strength of that packet. Both of those can be worked out by the base station that receives the packet, so we didn't need to actually send data in the packet. We considered sending an incrementing count in each packet, so that we could compare the RSSI of the same packet received by each base station, but this wasn't need as we decided that it would be better to find the average RSSI of a number of packets received over a short amount of time.

Data formatting for the base station to aggregator/calculation engine interchange was also straightforward. The RSSI data was of course the whole point and could be sent as an integer, while we would also need the tag ID, which is an 8 byte HEX value, to be sent as a string. We decided to add a single byte integer to

represent the base station ID as well so that we didn't have to dynamically find the ID of the base station on the aggregator end. Dynamic assignation of base station ID was impossible, unlike with the tags, as we needed to know which ID was assigned to a specific location for the calculations. The base stations were also going to need manually programmed unique MAC and IP addresses anyway, so adding a single integer ID made things easier.

The sole job of the aggregator is to connect to the base stations and receive data, passing that data on to the calculation engine, so the data format did not need to change on the way through. The calculation engine takes this raw data, turns the RSSI into distance values, smooths the data to find the most likely distance from each base station, and then performs the trilateration calculation to find the X and Y position values for each tag. These position values are sent to the UI as integers, along with the unchanged string of the tag ID. At this point a timestamp is also appended. The easiest way to deal with the timestamp was as a string with each value in the string separated by colons. The timestamp format is Year, Month, Day, Hour (24), Minute, Second, Microsecond.

In order for the calculation engine to trilaterate the positions of the tags, the locations of each base station must be known. The aggregator also needs to know the IP addresses of each base station in order to connect to them. Due to us deciding that the easiest way to pass the configuration along was as an XML file, the X and Y position values, ID, and IP address of each base station are recorded as strings, which are then converted to integers by the configuration parser.

5.10 How do we pass the base station configuration to the calculation engine?

The base station configuration is stored in XML. We chose XML because it is an easy to read format and both C# and Python have nice libraries to parse the data. For C# we used XDocument and Python has a built in XML parser. So that both programs talked to each other nicely, we agreed on the format before we coded the adaptor classes. This prevented miscommunication between group members and greatly sped up the development process as the UI and Python coders did not have to wait on each other's efforts to be completely finished before starting their own work.

5.11 Where should we timestamp the data?

We decided to timestamp the data once the position has been calculated as this provides the user with the best approximation of when the position has been calculated. Appending a timestamp to each point of RSSI data as it was received by the base station was considered for comparison and calculation purposes, but we rejected that as too computationally intensive on the part of the base stations. We would also have had to synchronise the clocks of the base stations, which would be too much work for not enough gain, possibly introduce errors, and in the end was not otherwise required. Timestamps have been added to the raw data as it is received by the aggregator, but is only used for later manual analytics, and is not required by the calculation engine.

5.12 What data smoothing method to use?

While it definitely shows a good trend in signal strength vs distance, RSSI data can be quite noisy even in the best conditions. We always knew that we were going to need a good data smoothing algorithm in order to reach an acceptable range of accuracy; however, the particular form that algorithm would take depended heavily on what the data looked like. Throughout the year we have been reading research papers on RSSI smoothing techniques, and we have a number of avenues that look promising. The problem we face is that what works for static or slowly moving tags is very different from what works with tags that are quickly moving and changing direction.

We started out with a basic average for our signal strength data, which was predictably nowhere near accurate enough, but gave us an idea of what sort of smoothing we would need. From this we determined that a weighted average based on the most commonly received signal strengths would be required. Testing for the number of most common data values, and the particular weighting attached to them, is still ongoing as we collate more testing data and improve accuracy.

5.13 Should the calculation be done in the same program as the viewer?

Originally we had a defined boundary between the hardware and the software portions of the project. Naturally the calculation engine was programmed by the same team members that programmed the UI. While we had intentions to keep the two separate, in the course of attempting to get something working as a minimum viable product the two were intertwined, as it vastly simplified data transfer and debugging.

When we switched to using Ethernet to pass the data from the base stations to the PC, we found that the networking modules for C# are difficult to work with and not as robust as we'd like. One of our lecturers suggested using Twisted Python as the networking module, so we switched to using that as our aggregator. In making the switch we discovered how easy it was to work with Python, and that there were a number of Python modules we could use to improve the calculation engine as well. This also had the benefit of allowing our user interface programmers to focus solely on the user interface, while at this point the hardware was finished, so our hardware programmers switched to working on improving the calculation engine.

Moving to two different programs to handle the different tasks fulfilled our initial intentions of keeping the user interface separate from the calculations. This improves the robustness of the project, as if something goes wrong with the UI, the aggregator and calculator will continue receiving data and calculating positions.

5.14 How often should the calculations be performed?

How often the tag positions are calculated depends largely on how often you want the positions updated. We designed the system so that we could run calculations as often as every 100 milliseconds. This gives about 6 data points per tag per base station, not enough

for proper smoothing of data, and requires the use data points that have also been used in previous calculations, which isn't a bad thing. Currently for testing purposes we are running a new calculation every 500 milliseconds, which gives a good amount of data for smoothing purposes, without overlap with older data.

5.15 How should we display the data to the user?

We decided to display the data to the user by creating a user interface that takes a player's position and displays it as a dot on the screen. We decided to display the data this way as it provides the users of our application an easy way to communicate coaching tips to players. This display method also allows the coach to display to the team in the after match analysis of each game. With this method the coach is able to project it onto a screen so all of the player can view the data at the same time.

5.16 How should we store configuration and location data in the long term?

The only way to properly store data long term is with a database. Having classroom experience with SQL databases and C# to SQL interfacing, this was the database system that we decided to use. The database design went through quite a number of changes over the year due to changing designs in other areas, but actual implementation of the database could safely be left until the rest of the system had been established. For testing purposes we output all data to CSV and text files, as both these formats are easy to implement.

6. DEVELOPMENT PROCESS

We have developed the system by using an Agile process. This consisted of three iterations that allowed us to develop a greater understanding of the problem. During each development iteration we set about breaking down each task into smaller chunks. Each of these chucks was made into mile stones and the group split up these tasks into manageable pieces. This methodology worked well as it allowed the group to take on roles they were good at. Each group member's specific skill also helped expand the other members of the group as all of the parts had to intercommunicate and group members would often explain and show how each part of the code worked to another group member.

As a part of the coding process we also used peer programming to solve any major issues that we faced. We also used Mercurial for our revision control. This enabled us to share code and keep a track of what changes other group members have made to the project. It also came in handy when things didn't work out as planned, as it allowed us to back out of branches. It also allowed us to work on the project at the same time without the fear of clashing with each other.

Figure 1. Full Scale Testing

7. ARTIFACT DESCRIPTION

7.1 Interaction Design

We started out with a multi-windowed design. Although it was good at breaking up the different parts of our interface, during one of our prototyping sessions it it was brought to our attention that most people don't really like lots of different windows popping up. So after this we worked on a different user interface design that used a tab control instead of multiple windows.

7.2 Software/System Architecture

The System Architecture of this project is made up of four main parts. The tags attached to each player that broadcast signals, the base stations that receive these signals, the aggregator that gets the data from the base stations and then calculates the position of each player, and the user interface which displays the final result. These parts all work together to provide the user with the location data of players who are skating around the rink. To track a player, a tag with an Xbee is attached to their helmet**.** Around the rink are arrayed multiple base stations that receive signals broadcast from the tags. From these received signals, the strengths of the signals can be retrieved. This is then retrieved from each base station via Ethernet by a Twisted Python client. This client aggregates the data from all of the base stations and then works out a weighted average for each base station for the 10 most common signal strengths received from each station. After this it converts the

signal strength into distance and then calculates the player's position based on trilateration.

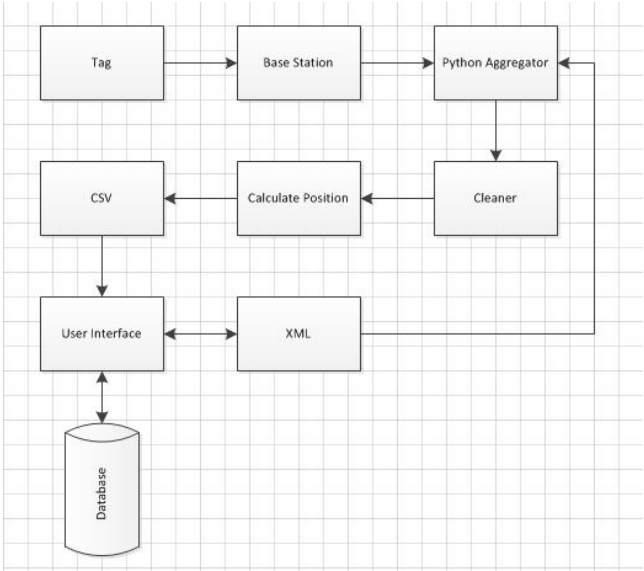

Figure 2. System Data Flow

Once the position of the player has been calculated, it is stored for the user interface to display nicely to the coaching staff and players. The user interface displays a map of the hockey rink and shows the location of the players'. It can just represent each player as a single dot, or it can represent the player with a trail of diminishing dots that show its last 5 positions. Another option for displaying the player data is to use a line that follows the player around. See **Figure 3.**

7.2 Functional Requirements versus Completed System

As we have been developing PlayTrack we have been continually referring back to our functional requirements. These requirements have changed as the project has changed scope. We have cut back in scope in some area of the functional requirements as we have encountered some technical challenges that have made us have to focus purely on the core functionality of the tracking technology itself and leave the statistical analysis work as part of future work on the project.

8. TECHNICAL HIGHLIGHTS

1. Real-time and re-playable playback:

   The ability to pause a real time game gives the coach the ability to see the location of all of the members of his team. This can also be used by the coach after a game when watching the a replay to point out key plays. This can also be used to point out to players areas where they can improve and what they can do to improve their game.

   Watching the recorded play-through of a game is a matter of reloading the data from the database and playing it back.

2. Portable:

   This system is designed to be easily portable and requires a laptop and an Internet connection. This makes it very easy to set up as all you have to do is place the base stations around the hockey rink. This allows the couch to take the system to away games with relative ease.

3. Adaptable to different Sports:

   Although this system is designed for ice hockey can be changed to work with other games with relative ease. The base stations can be easily moved and the settings changed to work for any indoor games IE: basketball, netball and soccer. This is because configurations can be changed to accommodate more than one type of playing area.

4. Data Aggregator and Calculation Engine:

We created a network data aggregator that is capable of scaling to record data from a huge number of base stations. This allows us to use as many or as few base stations as required without any extra setup. The calculation engine is similarly scalable, and written so that changing the algorithms used is a matter of simply writing and inserting new methods.

5. Hardware Platform:

   We created our own hardware platform based on off the shelf open-source hardware. It is easily extensible and scalable, making it adaptable to a number of different sports.

9. TESTING

We tested our system through a series of testing sessions at a variety of venues. These testing sessions involved the group setting up the equipment as if they were deploying it in a live situation and then placing a tag in a measured location. Once this had been done, we would collect data from the code and analyse whether our work had improved on the accuracy of the tracking. This testing would then feed back into our development process.

10. DEPLOYMENT EVIDENCE

Over the course of developing the system we have tested our software in large scale environments four times. Over the course of these testing sessions accuracy (mean squared error) went from 7239 to 96.52. This means that we have greatly improved on the accuracy of our calculations and can better predict where the player is situated on the playing area.

11. FUTURE WORK

Future work can be done on this project in the areas of smoothing algorithms and data visualization\ This would include doing a lot

more testing so we can refine our trilateration algorithm as well as working on the user interface code. The trilateration algorithm could be improved by creating more a effective smoothing algorithm as this would remove the less accurate data and give us much better information. Further analysis of player movement would allow the design and implementation of filtering algorithms that use past and predicted future positions to influence the current calculated position.

The user interface could be improved in a number of ways. The first of those would be to improve the way the play screen is displayed. It would look much nicer if it was displayed with the used of C# XNA display rather than using a windows Forms picture box. The next improvement would be to add a series of analytical features which could include: real time distance traveled, acceleration, heat-maps , full database support and comparison with other games that have been recorded.

12. CONCLUSION

This report has described our efforts at creating a system that can track the locations of players as they more around an indoor sports location. After reviewing the available location position solutions, we decided to use a trilateration based approach, utilising Received Signal Strength Indication data from wireless devices to find distances. We then created a hardware platform utilising Arduinos to control XBee Radio Frequency devices, which communicated over Ethernet with a Python based data aggregator and calculation engine. We also developed a User Interface by which a coach can watch and analyse the locations of the players in real-time, with later playback capability. In testing, we found that this system provided a robust and easy to use platform for which to track player locations. While we weren't able to achieve the degree of positioning accuracy that we would like, we have shown that it is possible to create a system. Further work to refine the data smoothing algorithms and implement further filtering techniques would continue to improve performance and accuracy.

13. ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed. Thanks to then lecturers and advisors at Otago Polytechnic. Special thanks to our stakeholders from industry.

14. REFERENCES

AForge.Net Framework Features- Motion Detection.). Retrieved March 15, 2012, from http://www.aforgenet.com/framework/features/motion_detection_2.0.htm

Bracko, M. R. (2004). Biomechanics powers ice hockey performance. *Biomechanics*, 47-53.

Bradski, G.). OpenCV, from http://opencv.willowgarage.com/wiki/

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV*: O'Reilly Media, Inc.

El-Deeb, A., Foina, A. G., Badia, R., & Ramirez-Fernandez, F. J. (2010). *Player tracker - a tool to analyse sport players using RFID.* Paper presented at the 8th IEEE International Conference on Pervasive Computing and Communications PERCOM, Mannheim.

Ganeshapillai, G., & Guttag, J. (2012). *Predicting the next pitch.* Paper presented at the MIT Sloan Sports Analytics Conference, Boston, MA, USA. http://www.theebgar.com/wp-content/uploads/2011/04/Predict-the-next-pitch.pdf

Goldsberry, K. (2012). *CourtVision: new visual and spatial analytics for the NBA*. Paper presented at the MIT Sloan Sports Analytics Conference, Boston, MA, USA. http://www.sloansportsconference.com/wp-content/uploads/2012/02/Goldsberry_Sloan_Submission.pdf

GPSports.). Retrieved March 15, 2012, from http://gpsports.com

Gray, D., & Tao, H. (2008). *Viewpoint invariant pedestrian recognition with an ensemble of localized features.* Paper presented at the ECCV '08 Proceedings of the 10th European Conference on Computer Vision: Part I.

Humprey, T.). Science of Hockey: Starting and Stopping on the Ice Retrieved March 16, 2012, from http://www.exploratorium.edu/hockey/skating2.html

Kirillov, A.). AForge.NET, from http://www.aforgenet.com/

Maheswaran, R., Chang, Y.-H., Henehan, A., & Danesis, S. (2012). *Deconstructing the rebound with optical tracking data*. Paper presented at the MIT Sloan Sports Analytics Conference, Boston, MA, USA. http://www.sloansportsconference.com/wp-content/uploads/2012/02/108-sloan-sports-2012-maheswaran-chang_updated.pdf

Pons, A. P., & Polak, P. (2008). Understanding user perspectives on biometric technology. *Commun. ACM, 51*(9), 115-118. doi: 10.1145/1378727.1389971

Science of Hockey: Shooting the Puck.). from http://www.exploratorium.edu/hockey/skating1.html

Sinha, U. (2010, July 19). Tracking colored objects in OpenCV Retrieved March 16, 2012, from http://www.aishack.in/2010/07/tracking-colored-objects-in-opencv/

SportVU.). Retrieved March 16, 2012, from http://www.sportvu.com/

Thomas, A. C. (2007). Inter-arrival Times of Goals in Ice Hockey. *Journal of Quantitative Analysis in Sports, 3*(2).

Tjondronegoro, D., Chen, Y.-P. P., & Joly, A. (2008). A scalable and extensible segment-event-object-based sports video retrieval system. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), 4*(2). doi: 10.1145/1352012.1352017

Trimble.com.). TM3000 Asset Tracking Device, from http://www.trimble.com/integrated-product/TM300-Asset-Tracking-Device.aspx?dtID=overview&

Verusco Sports.). Retrieved March 15, 2012, from http://www.verusco.com/video-analysis-software-rugby-union.html

Wang, X., & Zhang, X.-P. (2009). *Ice hockey shot event modeling with mixture hidden Markov model*. Paper presented at the Proceedings of the 1st ACM international workshop on Events in multimedia, Beijing, China. https://dl.acm.org/ft_gateway.cfm?id=1631031&type=pdf&CFID=90185993&CFTOKEN=56011261

Zeimpekis, V., Giaglis, G. M., & Lekakos, G. (2002). A taxonomy of indoor and outdoor positioning techniques

for mobile location services. *ACM SIGecom Exchanges - Mobile commerce*, *3*(4). doi: 10.1145/844351.844355