Faculty of Mathematics and Natural Science
Department of Computer Science
University of Groningen

Using Calibration Pinpoints for locating devices indoor

Master of Science Thesis

By:
Dennis Kanon
S1673491
University of Groningen

Author:	Dennis Kanon
Supervisor:	Prof.dr. M. Aiello
Second Reader:	Dr. M. Wilkinson
Date:	August 30, 2010

Abstract

In the field of Smart Homes a fundamental issue is that of the localization of people. Some technologies that work well outdoors, such as GPS, do not function as well inside a house. The problem of indoor localization is far from being solved, as a number of issues have to be addressed such as costs, acceptability of the solution by the user, robustness of the solution, amount of structural changes to the buildings that have to occur, and privacy.

The issue is often addressed recurring to high frequency radio waves. Techniques like Time of Arrival, Angle of Arrival and Signal Strength can be used to provide an estimate of the position of a device. Since the earlier two have the need of highly specialized hardware then the latter, we opt to investigate a solution that adopts the use of a Signal Strength Identifier. The Signal Strength Identifier, or SSID for short, is a value that indicates the strength of a received WIFI signal.

This approach has to handle the problem that the design of the building (such as walls, pipes, etc.) interferes with the line of sight of the signal and causes problems. For instance, the degradation of the signal is not regular. To tackle this problem, we propose the use of Calibration Pin Points. These Pin Points provide each room with one or more Points of measured SSID values.

From these values features can be extracted, which in turn can be used to either locate a device in relation to such a Pin Point. In this paper we define the method to set Pin Points and use them for indoor localization, we also provide an evaluation of the method in a building of the University of Groningen. The approach is based on the fact that the person to be localized carries a device to receive the earlier mentioned signal strengths.

In our evaluation, we resort to a simple implementation that can easily be ported to Hardware that is already in many users possessions. The increase of Smart Phones like the iPhone, Android Phones, Windows Mobile Phones, Symbian S60 Phones and the likes can be used to help tackle this problem. Many of these devices are capable of receiving these Signal Strengths and sending these values to a server. The actual localisation is Server side as to prevent load on the Smart Phone.

As a result the paper proposes a system that is capable of locating this device with a high enough precision to be used in a Smart House. The solution is cost effective with the use of ordinary access points and routers and can be implemented in any house without any construction necessary. Provided of course the house owners places the earlier mentioned routers and carry their phones with them.

Table of Content

1 Introduction

1.1 General Introduction

As smart devices become more and more prominent in our life, they become more and more powerful and connected to various wireless networks. It would be a shame to waste its potential, especially in the field of smart homes. Many of these devices include a wireless networking abilities adhering to one or more of the IEEE 802.11 standards [3].

The use of the IEEE 802.11 standard and its signal strength is often documented as a good way to find one's distance to an access point with relative ease in a perfect environment [1,2]. The distance from three of these access points can then be used to locate a single spot in this environment with a precision of a meter or less. This is more then enough for an indoor location system to find such a smart device's position.

Of course an indoor structure is not a perfect system and walls obstructing the line between an access point and a device will affect the signal. Often this cannot be predicted and sometimes a signal can even be seemingly erratic in its behaviour.

1.2 Research Questions

For the research going into the earlier described problem, we have to condense the solution into one or more research questions. The first four questions are discussed in this paper, while the other questions are a discussed in the paper of my research partner Simon P. Takens. His research focussed more on the actual localisation rather then the negation of interference:

- Is it possible by storing calibration values to solve the problem of a buildings layout and interference?
- To what level is it possible to negate this interference, by using these stored values and the Features extracted from them?
- What other features can one add besides the stored data, to further negate this interference?
- To what extend can these stored data, Features and other features be used to find a device with stored data location precision.
- Is it possible to use this negation to find a device in a building with a higher precision than stored data location precision?
- What different techniques exist and how can they be implemented with the earlier stored values in mind?
- What precision is possible given our current work?

The questions originally sprang by the idea of using a timed recording of values in a room for the localisation of a device in a room. Of course the precision of these recorded values needs to be tested to see to what level they can be used to redetect such a device. From this came the idea to use this stored data as a "finger print" for that room a Pin Point so to speak. They would be function as calibration points to use inside a building. These Pin Points would function as constants one knows and can use for localisation and tracking of a device.

Of course these constants can then be used to try and negate the influence the building has on the degradation of the signal strength. This however gave way to the research of my research

partner. He tried to find methods using these constants to find a device in between these Pin Points. This in turn would make the earlier mentioned stored data, extracted Features and other features constants to work with, in an otherwise unpredictable environment, here the Pin Points would basically form a signal degradation map of the building. The Pin Points can then help make sense of the seemingly erratic influence of the building, by providing more information about the degradation. This however is beyond the scope of this thesis and is discussed in the thesis of my research partner.

1.3 Dividing work and research

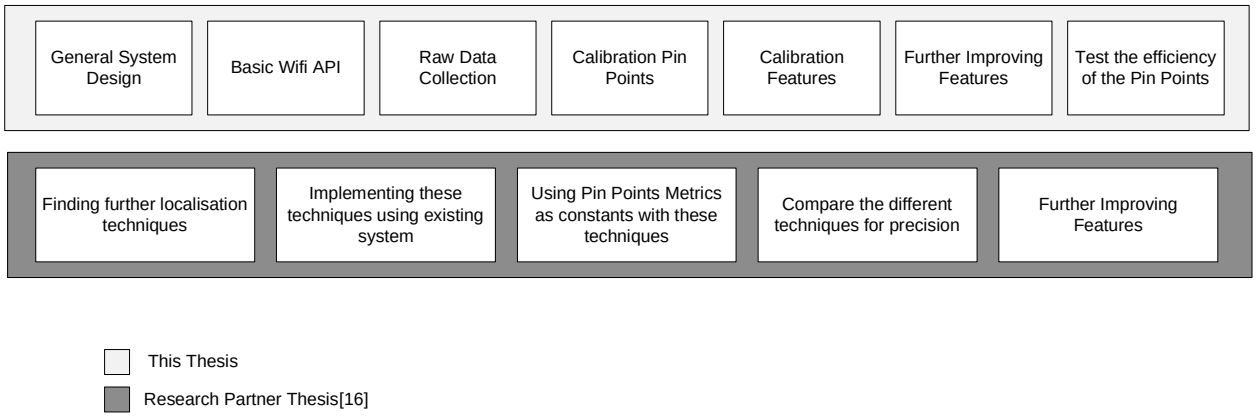


Figure 1: Division of research divided per Thesis using colouration

Figure 1 shows the general division in research. This paper and my research focused on the design of the general system, the WiFi API and the generation and testing of the Pin Points. This split was made on the basis of workload and research required for each of these options. By then the WiFi Api and general system design was already mostly done, and the focus shifted on researching topics related to the generation and testing of these Calibration Pin Points, The Calibration Features and other improving features.

From here the implementation needed to be tested with the help of a predetermined test set-up. It also was needed to establish what values and results needed to be collected. Afterwards based on the collected results, there should be some discussion and interpretation that will then be detailed in this thesis.

1.4 Methodology

The theory behind the system is that each room inside a building has a unique influence on the signal. This not only depends on the room's distance from the transmitter, but also on multicast, line of sight and the construction materials used. Furthermore, the location of neighbouring rooms can also affect the signal strength. This usually is a big problem when one wants to locate a device via the received signal strength indication. Received Signal Strength Indication, abbreviated as RSSI, does however tend to stay at a constant level when one stays at the particular position. Another minor influence is the presence of people, it is recommended to make the said Pin Points during days that the building is populated to record the best values.

Using the information provided in the section above it was a natural progression to make identifying fingerprints per room, the so-called Calibration Pin Points. The original idea was to use these Pin Points as localisation to find out if a person was in a room or not. This would create a map of constant Pin Point values collected from 3 different Access Points taken over time. From these values certain characteristics can be extracted, for instance the average value of the RSSI per Signal. Then as the device detects its own RSSI the comparison to this value, or Feature, can then be used to provide us with a positive identification or not of the location of the device with the precision of a Pin Point.

This in turn would create a map of the area for the system by the use of previously collected information. It might even be possible to use these values to estimate a position in between these Pin Points as they provide us with some insight of that particular areas degradation per room. This however is beyond the scope of this paper and is discussed in the paper of my research partner [16].

However the extraction of Features can also provide the system with added precision. Since the Feature described above is just one average value it might not be very precise. Other Features like a most frequent and median value might be more precise. However, the research also included the testing of various Features and Feature designs. These are single value Features; other Features like a Feature that uses more then one value or even a Feature that would require more then one measurement from the device are also discussed.

A Feature in essence is the extracted information from the stored raw data values, which can be tested with the on the fly gathered results from a device to either give a positive identification or a comparison.

To solve this problem we thought of the following. Measuring the signal strength over time at certain locations from more then one access point we try to capture a “finger print” of that location. These “finger prints” or Pin Points as we call them we believe are unique in its behaviour of the strength of the signal. So in short the pinpoint is nothing more then a measure over time for a specific location in which all the raw data of these access points is stored.

From this raw information we need to extract information of measurement, since just a raw collection of signal strengths on given times is a lot of information to compare. We call this extracted information a “Feature”. These Features can measure multiple aspects of a signal pinpoint, like a pattern over time, averages and mean and compare these to a measured signal of a smart device.

1.5 Test Objectives

In this research we want to test to what extend one can use the Pin Points as a constant factor in a building. This testing can be done by trying to detect how well a device can be tracked on the earlier mentioned Pin Points, the number of false positives, how well these values hold themselves over a long period of time and how well the values hold when moving from room to room.

This will also sufficiently test the Pin Point’s functionality as a map of constants for the area, showing the degradation of the signals over the area. Since if the system is properly able to

find a device it will mean that the values gathered are useable for the other research discussed in the paper of my research partner [16].

Furthermore the testing should include the different features added to improve the results, running each test with and without the features does this. But also running a test that has all of the features enabled. This should show the benefit of the features and it's effect on the gathered results. To prove how much the loss of a transmitter affects the system, there also should be a test that uses less then the required amount of transmitters that can then be compared to the results of the others tests.

1.6 Summary

In short this paper focuses on the related work, approach, implementation, experimentation, results and the discussion of afore mentioned research. First the identification of related research and work is essential to full understand and tackle the problem. It also creates a basis for the reader to get a more in-depth idea of the problem. Furthermore some of the inspiration was taken from the research done by others and thus it is important to state their work and progress in the field of indoor localisation.

After this the paper focuses on the Approach, which basically discusses the approach to the problem using algorithms, pseudo code and diagrams. It also tries to explain what the different known problems are and how it is tried to counter these. Basically it discusses the problems, solutions and the system in a theoretic and/or schematic way.

After approach follows a section about the implementation, this is mostly the general design of the software using during the testing. It describes what language was used, developing environment and uses UML to show the general architecture of the system. It serves to give the reader an understanding of the se-tup, architecture and development tools used while testing and developing the system.

Next is the discussion of the experimentation. In this section the thesis mostly focuses on the set up of the experiment and how the system will be tested. It also discusses what tests will be done, and what results will be recorded. Besides these points, the section will also contain information about the hardware used, the map of the area in which the system was tested.

Which follows into the collection of the results, the section that mostly sums up what results have been found in graphs, tables and images. This is accompanied with some text describing the different tables but not any interpretation of the results. This is reserved for the next section, which is the discussion of these results. In which these results are used draw conclusions, graphs, tables and interpretations from the experimentation.

Afterwards the paper discusses some conclusions and suggests some of the future work which might be interesting to research based on the experimentation.

2 Related Work

Over the years many different ways of locating both devices and people have been identified. Varying from the use of Radio Frequency Identification tags (RFID tags) to using simpler motes that detect signal strength in a room to find people located in that room. The various techniques all have their advantages and disadvantages.

2.1 RFID Technology

One way of dealing with the problem is via the means of RFID Tags and receivers. These tags send their signal to receivers. These receivers in turn can use the signal strength of a device to calculate an estimate of the distance. There are a few ideas of using this technology for a location services in a building.

The first, which is interesting to discuss, was basically designed for the control of lighting in a building [4]. Testing it in an area, this area was in turn divided into rooms. Multiple readers where placed strategically through the area and they used signal strength to obtain an approximation of the location. This placement would then use the received signal strength of an RFID tag and with the help of an algorithm based on a Support Vector Machine (SVM) it would generate results for each given room.

After this they used a round robin to compare the rooms to each other, this could either result into a tie (1), a win (3) or a lose (0). Each room now has its point score and a rank is calculated. To further increase the efficiency of the system they used the layout of the testing area as an indication of what room you might go in next. This increased the precision of the results but could create a fault if a person moved faster then the system could compensate for. In these cases the system would get stuck in a room when in fact the tag itself was already in a completely different room. To solve this problem, this research compared the possible position to the location indicated by the measured values. If the possible position differed from the actual location for a longer period of time, the possible position is changed to the actual location. This last addition offered some interesting ideas when thinking of Features, as it allowed for the inspiration for the distance modifier theory for the Pin Points.

2.1.1 LANDMARC

Another interesting technology using RFID is LANDMARC[5] (Location Identification based on Dynamic Active RFID Calibration); this used a raster of RFID tags in a room and low number of the more expensive readers. The locations of these tags are known as the reader reads them out and records their signal strength.

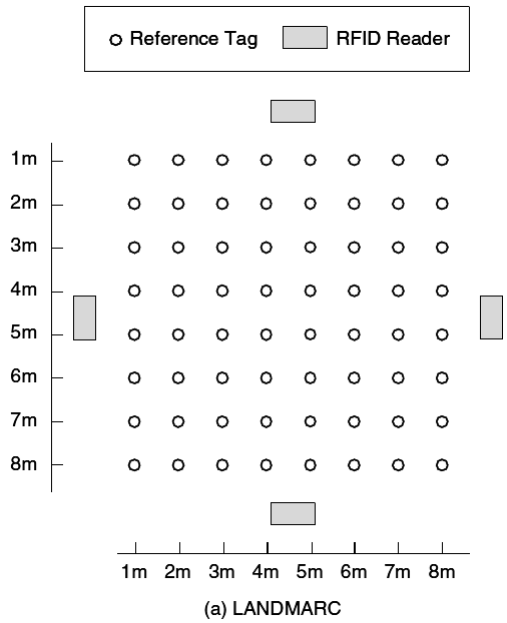


Figure 2: Raster layout of LANDMARC, Source: FLEXOR: A Flexible Localization Scheme [6]

When a person wearing a different RFID tag enters the room the readers will pick up his tag as will his signal strength. By using this signal strength to compare it to all other tags in that room via a formula that will give you a number. The lower this number is, the closer the person wearing the tag is to that tag in the room.

This tag is then used with the tags surrounding the user tag with the closest values to estimate a position in-between the raster of tags. This gives you quite a precise location of the user tag in that room, with precision of the tag lattice or higher.

Since RFID Tags are fairly low on costs the landmark system itself is quite cost effective and thus a good contender for cheap indoor location of other tags. The downside is interference, which still can affect LANDMARC to quite an extent and it has a lot of overhead for each new tag entering the area.

2.1.2 FLEXOR

A technology using the lessons learned in LANDMARC and extending their research is FLEXOR[6] (Flexible Localization EXplOits Rfid). In this technology they create cells of tags instead of a raster. These cells have a tag at the centre named a “Cell Tag”; “Boundary tags” indicate the border between Cells. The number of readers is still the same.

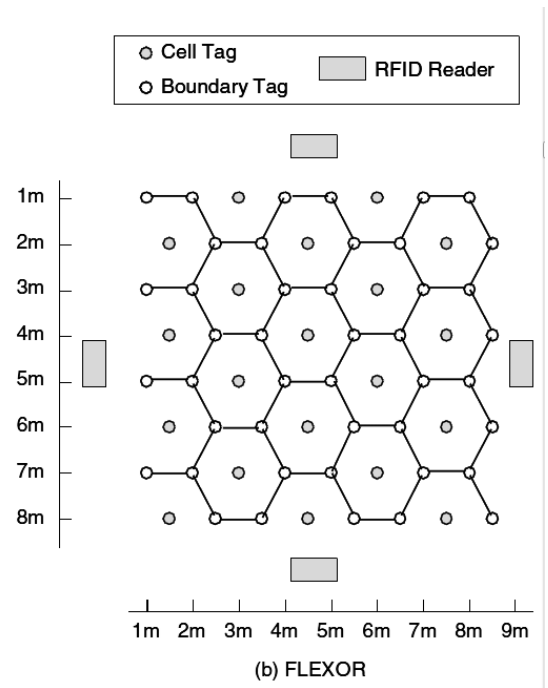


Figure 3: Cell layout of FLEXOR, Source: FLEXOR: A Flexible Localization Scheme [6]

The reason for this is that it is not always necessary to provide all of the information LANDMARC provided; in these case the region is enough for a location service rather then precise coordinates. This saves on computational time, as the system does not need to check all of the tags all of the time, the Tags between bordering cells are enough. FLEXOR also is less influenced by interference.

Like LANDMARC, FLEXOR tries to find the tag closest to the tag it needs to track. As mentioned earlier a precise location is not always needed, therefore FLEXOR supports both a “coordinates mode” and a “region mode”.

In the region mode FLEXOR only uses the cell tags to detect which region a tracked tag is in. This is done like LANDMARC by finding the cell tag that has signal strength closest to the measure signal strength of the tag you are tracking.

In the “coordinates mode” starts out with a region mode and then uses the “Boundary tags” to further specify a location. First FLEXOR finds the boundary tag closest to the tag it is tracking. The next step then looks to the two adjacent boundary tags in that cell and finds the one that is the next nearest to the tag it is tracking. By using these three tags and their coordinates one can calculate a coordinate by using a 3 nearest neighbour algorithm.

This algorithm uses the weight of each reference tag, which is computed by calculating the distance value between the tracking tag and the reference tags. This distance value calculation is the same as LANDMARC, but unlike LANDMARC it requires far less computational power to process. Since LANDMARC uses all reference tags and this only 3. Also FLEXOR requires fewer tags to perform in a comparable fashion to LANDMARC.

2.2 Received Signal Strength based Technology

Since a signal gets weaker over distance, it is a way to indicate your distance from a transmitter. Various objects in its path, like walls or people, also affect it. This usually causes problems but can also be used to detect objects inside an area. Discussed in this section of the paper are the various methods of using Received Signal Strength localisation outside of the earlier mentioned RFID based methods.

Claims are made that LQI (Link Quality Indicator) is a better choice in this field then RSS, but some studies show that this actually is far from the truth [2]. Signal strength can actually be used quite well and it has shown to be far more useable then originally thought. LQI is an indicator of the quality of the signal rather then the strength of the signal.

One method was to use sensors measuring the signal strength in an area by sensors at its borders. If a person then enters this area it would affect the received signal strength in such a way. By first measuring with six sensors in an outside environment and splitting this area up in a raster, they where able to test the affect a person would have while standing at each section of this raster [1].

A team consisting of some of the same researchers and this time in an indoor environment followed up the afore-mentioned research [10]. Here they showed in a more in-depth analysis what affects the signal and how it affects the signal. In this scenario they also kept in mind the inclusion of office objects like chairs and desks. Since you also have a multicast problem (walls reflecting the signal) the system required additional training compared to the earlier research.

Another way of using RSS is by the use of multiple transmitters and finding a sensor. These access points broadcast their signal as it degrades over distance. This degradation is used to calculate how far that device is from one access point [8]. By using multiple access points with multiple signals and knowing where these devices are one can make an estimate of the location of the sensor. This does need a lot of calibration in a building, because of multi path and line of sight problems.

One of these techniques is called “RADAR” (not to be confused with RADAR technology) [14]. This technology uses the signal strength and receivers to get an estimate of where the devices are in a building. It does have one drawback that each time a building changes significantly the system needs to be recalibrated.

SpotON [15] is system that uses Adhoc Wireless Sensor technology, it sensors can be placed randomly and it can do full 3D localisation. It does not require a central controller and in the above-mentioned features it seems to be quite unique.

As the above text shows RSS is quite often used for location-aware services. And wit the right amount of calibration and algorithms it can be tweaked to quite a precise mean of measurement.

2.3 Time of Arrival based Technology

Another way to find your distance from a radio source would be Time of Arrival [7]. This basically means measuring the time difference between the transmitter and the receiver and using this to calculate a distance. These technologies require some specific hardware and lower frequencies to work. As a radio signal travels with the speed of light it is a difficult process to just measure in software without the use of dedicated hardware.

This technology has some problems of its own and in that requires a different way of implementation. One of these problems is bandwidth, since sending a timestamp requires you to send data, which in turn has to be received and tested [8].

2.4 Angle of Arrival based Technology

Angle of arrival is another technique of indicating a position of a device. The idea behind this is finding the angle of the device to the antennas. For instance if there are two antennas they form an imaginary line together. As the transmitter is inline with the antennas (bore-side) will result in an angle of 0 degrees and a broad side location to the antenna will result in a 180 degree angle.

By adding a third antenna one can find a position of a transmitter between antennas one can extrapolate the angles into a position, since a third antenna will add at least two more angle possibilities on possible antenna lines. From these angles one can extrapolate lines with angles on the lines created by the two antennas. Since the device is somewhere on this line the second imaginary line from a third antenna with one of the aforementioned antennas will give you a more exact position, since the spot where these two lines cross each other should be the position of the device.

This technology is used with cell phones and cell towers to find individual cell phones [9], this technique has also been used with success to find missing people by their GSM signal. For good results however it would require direction sensing antenna.

2.5 Other technology

Of course not all techniques are based on Radio Frequency technology. Some of these techniques are described below.

2.5.1 Active Badge Location System

A technology named “Active badge Location System”[12], was designed in the early 90’s and used infrared light to locate badges worn by the occupants. It periodically emits an infrared burst, which in turn is reflected by the badge. From the system can track in which room an occupant is.

The problem with this technique is that light does not pass through walls or opaque objects. This would mean that each room would need it’s own infrared transmitters and receivers. On top of this everybody would need to wear a badge, This badge would need to be worn outside

of the clothing thus making it even trickier to receive a proper ID. Any obstruction will affect this system.

2.5.2 Cricket

Another example of a non-RF based technology is “Cricket” [13]. Cricket uses Time Difference of Arrival (TDOA) to find a position of a receiver. Cricket uses both infrared as ultrasound to find the distance between the various transmitters (or between various receivers and one transmitter) and a receiver.

This system has the same flaw as earlier in case of the infrared, on top of that ultra sound has a hard time passing through walls and again would mean the system would need transmitters and receivers per room.

2.5.3 SLAM

There is a lot of research done in the field of camera-based localisation. One of these technologies goes by the abbreviation of SLAM. SLAM means Simultaneous Localisation And Mapping. It builds a map while trying to locate the camera-based device. It is closely related to the field of Computer vision and is especially popular in the field of Robotics.

Regarding SLAM there have been quite a few papers, ranging from using only a Single camera on a robot[20] but it is also used to locate people or objects in rooms and buildings[21]. SLAM however does have a drawback is that it is quite hard to identify the persons involved, and that it would require quite some computer processing to process all of the feeds for all of the rooms involved.

2.5.4 NIST Smart Space Project

Rather then just localisation, the Aim of the NIST Smart Space Project [22] is to make a smart room for meetings. With this in mind they created a system that could gather information from two hundred and eighty microphones. This created huge amounts of data time stamped for the system to process. From this data they can extract information, like location of the people in the room, what was said or anything else that the 200gigabytes an hour of information can provide.

From this information, as mentioned before, a lot can be extracted and multiple ways of location aware systems have been using the technologies used at the Smart Space Project as a basis of their own. Like in the earlier mentioned SLAM field.

2.5.5 Indoor Localization Using Camera Phones

This system used Camera Phones and their GPRS connection to provide some localisation awareness [19]. The system was a very unique approach to the problem, as it basically uses the individual phones as eyes. This in itself was very interesting especially in the field of robotic eyes and identification of location.

It does however have a few problems, first being that the phone should always be on the outside with the camera pointing forward. Another is that the phone’s battery will wear out quite fast if the system constantly has to make a movie or constantly take pictures. The phone had to be worn around the users neck. All of the information was send over a network connection, in this case a GPRS connection, which will create traffic or even a lot of traffic if one has to send photos. The technology itself was although very creative and interesting.

2.6 Discussion

From the topics described above we got several ideas for our own system, especially the RFID based technologies where interesting. These technologies created the base for the theory behind the software based Calibration Pin Points. The difference being that in the RFID based research one uses Marker Tags and Localisation tags to find the Marker Tag, and the work tested in this paper uses a software-based approach aided by standard WiFi Routers. Also using the layout of the tags to estimate positions and finding the closest was inspirational to the Pin Point detection algorithms.

Even though the RFID based technologies are different, the Features used in that research are still quite interesting for our research to study and use for inspiration. The papers discussing the use of RSS for localisation where interesting and on the subject. Further reading about other technologies proved interesting to get an idea of the current state of technology. The next section of this document discusses the hypothesis of our system.

3 Approach

This section of the paper largely discusses our hypothesis and ideas and why we choose for this system. Regarding this I split up this section in several sub sections that have an impact on our hypothesis formulation or if it has to do with the hypothesis itself.

3.1 Signal Strength over Distance

As is well known signal strength decreases over a distance, this means that the further the distance from a source the lower the strength of a signal is. We tested this in a hallway of the second floor of the Bernoulli Borg. This way we already get an idea of how the signal loses its strength in an actual building without walls in its line of sight.

In figure 4 shows a graph with the result of this measurement. Interesting to notice is that the signal loss is actually quite linear. Even the distances where the lowest loss and the highest loss are separated, the average of the two still would give you linear line. In this research we assume irregularities are caused by the building structure, walls and metal beams that affect the strength of the signal in an erratic way.

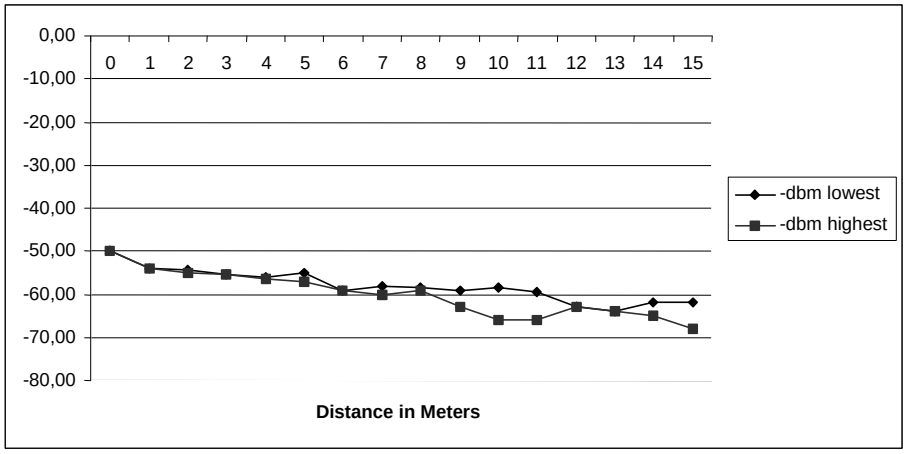


Figure 4: Measurements of signal strength over distance, where the blue line is the lowest loss of signal and the pink line the highest.

Even though there are some irregularities, the degradation of signal strength over a distance of a meter seems to be more then enough for use in our experiment as a means of identifying a location. Even the irregular measurements can be used to an advantage as discussed in the next section, which talk more in-depth about the problems one encounters in a building.

3.2 Problems in a building

In an ideal situation Signal strength is a very good way of measuring a distance from a signal source. However a building is not an ideal environment, walls, metal grating, metal beams and difference between floors can have a lot of influence on the strength of the signal.

Two of the biggest problems in a building are multicast and line of sight. Both of these problems deserve some more in-depth attention as they are used in the research for defining unique Pin Points. Multicast and Line of Sight will not only have a significant effect on the signal strength but will also helps defining rooms from each other.

3.2.1 Multicast

The first thing discussed is the problems caused by Multicasting. In this situation the walls of a building reflect the signal in such a way that it can actually provide you a slight boost in signal strength.

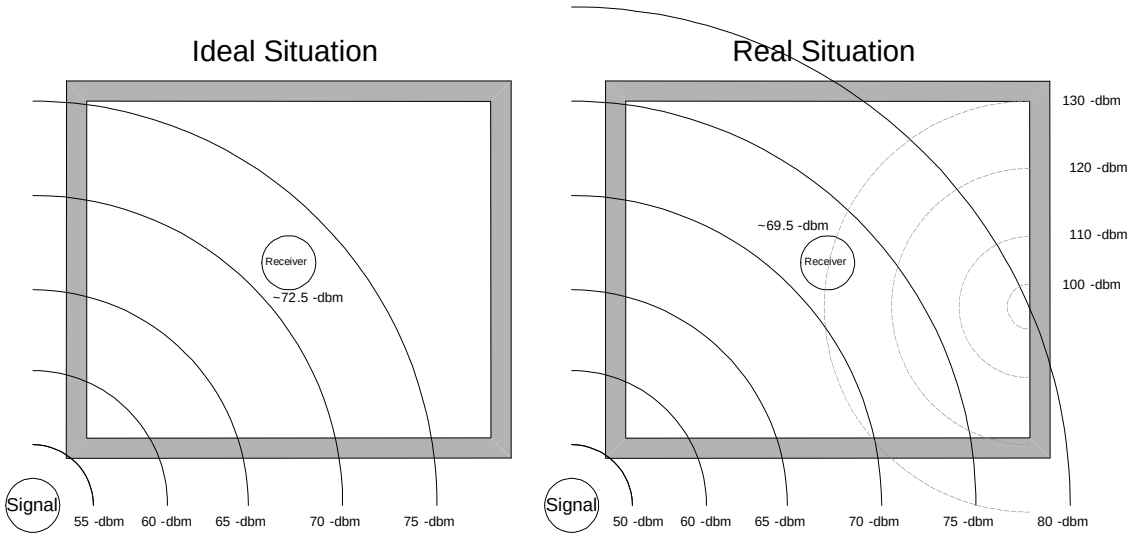


Figure 5: A simpler schematic form showing the multicast problem. The left picture is the ideal situation the right depicts a more realistic situation.

As shown in figure 5, multicast can seriously affect the readings for the receiver. A signal "bounces" of a wall in a room and affecting the signal strength at the position of the reader. This can result in a slight boost in signal. This boost is one of the larger problems for finding your position by signal strengths when using a linear degrading model in an actual environment and can affect the readouts. Thus the estimated model that would work in the ideal environment shown in the left diagram of figure 5 would be affected by this problem.

In this paper this is treated as a solution to the problem, since even if a signal reflects from a wall it still is fairly unique at that particular location. The hypothesis is that in this situation the measured values of three different signal transmitters will be unique for that area and that multicast will help in that respect for creating unique Pin Points per room.

3.2.2 Line of Sight

Another problem in an actual environment is the "line of sight" problem, a signal drops rapidly with each object in the line of sight of the transmitter and the receiver. Since a building consists of many walls they can actually affect the signal in a different way for each building.

As shown in figure 6 the relative strength of a signal can drop significantly. This is a common problem that plagues not only the usage of signal strength for location aware purposes but also when building a WIFI network inside a building. Often this is the cause of a blind spot with poor reception in a building with otherwise a more then adequate signal reception.

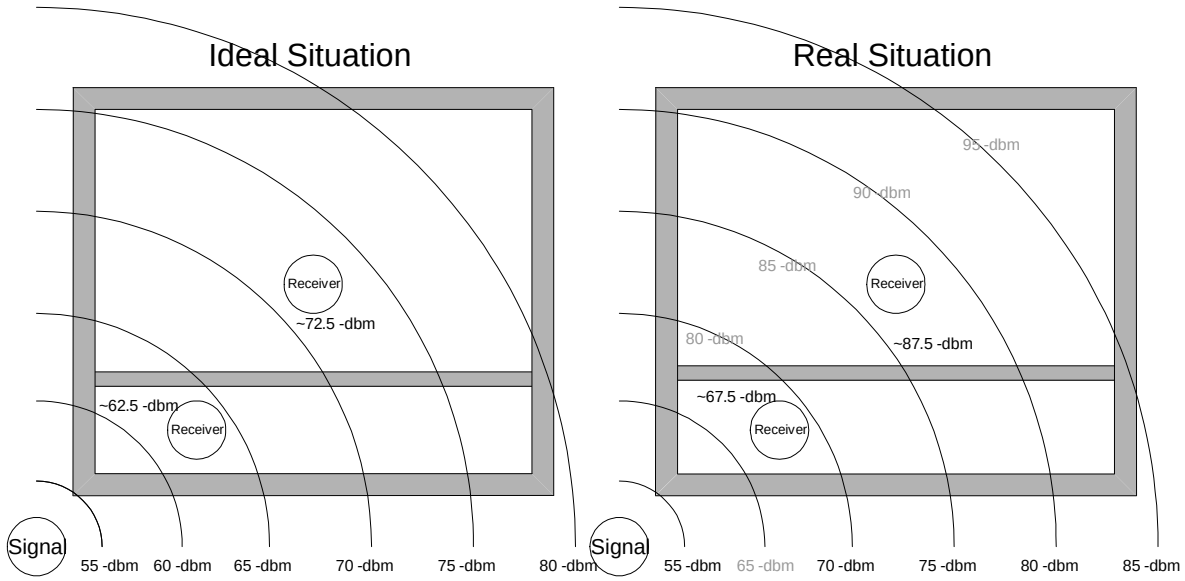


Figure 6: A simpler schematic form showing the Line of sight problem. The left is the ideal situation; the right shows how objects like walls affect the strength of the signal.

For us this problem is a two edged sword, in a building blind spots will also bother our system. But in this paper it is theorized that walls and rooms actually will make our Calibration Pin Points more unique. In our hypothesis the signal received from three or more transmitters will be very different this way from each room. Since rooms are often designed different on each of the transmitter but also different in line of sight between each transmitter and receiver we think it will give very specific readings for each room thus making it easier to identify our Pin Points.

3.3 Calibration Pin Points

As earlier mentioned our research wants to use software defined pinpoints in rooms to create a form of calibrated spots of which Features can be extrapolated and later tested on with the received signal from a smart device. The general idea behind this is that each section in a building has a fairly specific signature when measuring 3 or more signal strengths.

We propose to create these pinpoints by measuring for a longer period of time at a certain location. The values found over time are then stored in a file, which also contains the name and the graphical representation of this location. It is our theory that the values found at these locations will be very unique, as the gradual decay of signal strength plus Line of Sight problems and Multicasting, and will give a very unique representation of location.

From this information we wish to extract information that can be used for the positive identification of the earlier mentioned Pin Points, which in turn can also be used to get a room estimate of the location of a smart device. At the same time our research also tried localisation between the earlier mentioned Pin Points. These techniques are only shortly mentioned in this paper as its research is documented in the paper of my research partner: Simon P. Takens.

3.4 Features

Earlier this paper mentioned the use of Features but not what Features exactly are and the theory behind these. Since the actual implementation and theory behind each of the Features is discussed in later sections of this paper, this section is more about the global idea of Features, what they are used for and why there is more then one.

3.4.1 What is a Feature

Creation of a Feature, in our theory, is finding useful patterns in otherwise raw recorded data for later use of identification. Basically it means that a Feature is to try to find structure and logic within a large amount of numbers and time notations. These structures have to be easy to compare to measured data later on to identify if someone is at a Pin Point.

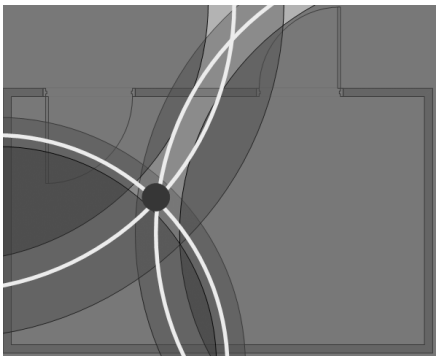


Figure 7: an example of a Feature, where the average between a minimum and maximum RSS is used to compare.

Figure 7 is an example of such a Feature and basically the simplest one. This Feature takes all the values and averages it as one value. The reason behind this is that the value later read by the smart device should always only divert from this slightly and thus can be used to positively identify its location at the said Pin Point.

Of course there are some problems with this approach that will be explained in the next section of why it is interesting to use more then one Feature.

3.4.2 Why more then one Feature

The answer to this question is simple; more often then not raw data contains peaks or valleys. This will affect the average number and maybe in such an extreme way that it is really different from the numbers measured most frequently. In fact taking the average is one of the weakest Feature as it may take a lot of fine-tuning for an allowed error. There is also no guarantee that the fine-tuning is the same for each Pin Point or even each Signal for a Pin Point. This makes it actually a very weak Feature, but still good enough to use in the final tests. In this research however we do not want to rely on this Feature alone.

Other Features need to be found with better characteristics that will help in a more reliable identification of a Pin Point. For example the most commonly found signal strength or the highest and the lowest value found. The system also allows for Features that require more then one measurement before giving a positive identification.

3.4.3 What kind of Features

There are a lot of ideas for Features but in general these can be split up in three categories:

1. Single Value Features
2. Multi Value Features
3. Multi Measurement Features

Features that fall in the first category are Features that create one value to compare a reading too from the raw information found, like the Average Feature. These are expected to perform worse then the other types with some exceptions. Errors like generating false positives or no

identification at all. A false positive is an identification of a location when you are in fact not even close to the location, a false negative is the reverse of this saying you're not on a location when you are.

An improvement on this is the Multi Value Feature, which compares one measured value to more then one and then gives a result. While we think this is more precise then it's single value alternative but still lacking enough possibilities to validate the design of a third kind.

Multi Measurement Features require more then one measuring value from a smart device before it gives a positive ID. This Feature will allow for the most complex of comparisons after generation. The only drawback behind these is that they require more time to identify a location because it needs more then one measuring value, typically between 25 and 200 or even more, before giving a positive ID. It is our theory that they will add a certain stability to the system, even if the detection itself is lower as generating a false negative or positive will not affect the amount of measures as much as only one measurement pass.

3.5 Ways of Feature Comparison

The earlier mentioned types of Features need to be compared to each other for a proper identification of Pin Point. There are a few techniques interesting for this selection. Since not every Feature is equally important the first thing we need to find a system for is the weight of each Feature.

3.5.1 Importance of Feature

As earlier mentioned not every Feature is equally efficient. This introduced the need for a system that added a higher importance to the Features that perform well, while not discarding the Features that are more prone to error. In general this means that the system gives an importance number to a Feature. This number is a weight in which to compare the results of a Feature during a check to the results of the other Features. The higher the number the higher a Feature can score.

The theoretical implementation of this is further described in the proposed method. In here we will also go deeper into the exact calculation of the comparison and why it is important for the two systems using this importance for their identification of a Pin Point.

3.5.2 Nearest Pin Points

As discussed earlier each Feature is assigned a number that defines its importance. However from this point we can still add some fine-tuning in the actual selection of Pin Points. One of the Fine tunings we found interesting was inspired by the RFID technology that was devised to control the lighting in a building [4]. They checked what rooms are situated next to a positive ID as candidates for the next position.

Our research does not work with rooms but some Pin Points are still closer to the detected point then others. These points we would give a bonus in total importance compared to the rooms that are further away and thus less likely for the smart device to move to next. The importance of nearest neighbours can be recalculated after a positive identification of a new Pin Point. The actual theory behind this is explained further later in this section of the paper.

3.5.3 Combination

The next step in the process would be to combine the two earlier mentioned methods. This means that the importance of well performing Features and a multiplier for closer Features to the last one found should both be implemented and work together to create a stable detection of Pin Points. More details about the theory behind this, is again explained in the next section of this paper.

3.6 Location of Access Points

Since the signal strength of each access point is important, the location of these access points also is a factor that should be taken into consideration. To just place all three access points in one room simply will not work. The problem with placing all of these access points in a single room is that the signal strengths degrade almost equally and thus are difficult to use as valid signal for localization.

A better way would be to place these access points at locations that are separated from each other by at least a room. Then the signal already varies enough to do some form of localisation but still there might be some problems in regards to room design with regards to multicasting and line of sight.

The third way and best way is to place the access points near the perimeter of the building or maximum ranges of other access points. For larger buildings this means sometimes placing an access point in the middle between two to locate areas, but for most buildings this means placing them near the outer walls in different areas of the building.

3.7 Single Value Features

One of the theories most important to our method is the quality of Features, the simplest of which is the single value Feature. In essence this Feature generates one value from supplied raw data that in turn can be used for comparison to one value read from a smart device. The general equation for this is shown in (1).

$$Result = \begin{cases} TRUE\ , if\ \ f(\vec{x}) \geq required \\ FALSE\ , otherwise \end{cases} \quad \textbf{(1)}$$

Which is a check if the measured value x is within the required limit after testing this value with the function f. Vector x contains the values measured for each of the access points.

Function f(x) can be different per Feature and returns a value that can be checked with the required value for that Feature. If this value equals or surpasses the required amount, the result will be true otherwise false. This means that for this specific Feature the value either passed or failed as a check for location. The “required” value is also something that is different per Feature; this can be a percentage comparison or a value comparison.

As Features need to be created, there also is a general formula per Feature that creates the Feature value to check on from the raw data. The general form of this equation is quite simple as it is described in (2).

$$\vec{y} = z(x) \quad \textbf{(2)}$$

This simple form allows for values to be assigned to vector y as function z generates this from raw data x. This value can later be used in the general formula described above.

The reason for having the two general forms is that they can be used as a framework for the later to be defined Features. Knowing all Features will respond in a similar manner makes it possible to check them in a general way that is the same for all. This means that in a list of Features there is no need for further defining specific code for checks or generation. More on this is described in the section that speaks about the importance of Features, which also discusses the general checking system.

3.7.1 Average Value

The first Feature used within our system is the average value of raw data. This is achieved by summing the raw values for each access point for comparison to later values. This is done as shown in equation (3).

$$z(x) = \left(\sum_{i=0}^{n-1} (\vec{x})_i\right)/n \qquad \textbf{(3)}$$

Where <x> is a list of vectors with values for each of the access points is shown in equation (4).

$$\langle \vec{x} \rangle = \{(\vec{x})_0, (\vec{x})_1, \ldots, (\vec{x})_{n-1}, (\vec{x})_n\} \qquad \textbf{(4)}$$

In these equations each value of x in list <x> consists of values for each access point. The values for each access point are summed into one vector that has a sum for each access point. After this it is divided by the total amount of values in the raw data. This results in one average value for each access point, which is stored and can be used for checking with the measured data.

The next course of action is to calculate a value, which can be compared, to the required value. This is done as shown in equation (5).

$$f(\vec{x}) = 100 - \left(\frac{|\vec{x} - \vec{y}|}{\vec{y}}\right) * 100 \qquad \textbf{(5)}$$

Basically this equation (5) calculated the percentage difference between the measured values in vector x from each access point compared to the average values calculated in vector y.

This value can then be compared to the required values, as stated in the general form before. Required in this case would be at least 3 access points have to be within a certain percentage difference of vector y. This percentage is not known in the theory here, since we want to test this with different values and see what the results are. Required in this case would be a number of access points passed and it has to be equal to or at least 3.

3.7.2 Mode Value

Another implemented single value Feature is calculating the mode value of the raw data. This basically means finding the most frequent signal strength and putting it in the vector y. This is done by first counting each unique occurrence of a frequency for an access point, as shown in equation (6).

$$\vec{y} = \langle x_0, x_1, .., x_{n-2}, x_{n-1} \rangle \quad \textbf{(6)}$$

Where the collection of <x> is defined by equation (7).

$$x_j = max \left\{ \begin{matrix} z_0 \\ z_1 \\ \vdots \\ z_{m-2} \\ z_{m-1} \end{matrix} \right\}$$

$$0 > j < n \quad \textbf{(7)}$$

The values of z are then generated in (8).

$$z_i = count\ unique\ frequencies$$

$$0 > i < m \quad \textbf{(8)}$$

After this process is completed one has to find the maximum amount of occurrences for that frequency in a given access point.

These values are put into a vector y so y consists of a vector of frequencies occurrences per access point. This vector can later be used for a comparison to a vector consisting of values measured. This comparison is identical to the earlier described comparison for the average value. It uses the same principle of calculating a percentage for each measured value before comparing it to the required values for a positive identification of a pinpoint. Again the requirement in this case is to pass for at least 3 signals or more.

3.7.3 Median Value

The median value basically is taking the middle point of a sorted list of values. In case of the total indices of this list being even you add the two and divide them by two. Now for the system this seemed an interesting single value Feature. Often in a sorted list the value found with the median is actually a value that is either common or close to the most common value, and thus might give very good results.

The equation for this is shown in equation 6.

$$z(x)=\begin{cases} n \bmod 2=0, \dfrac{(x'_{(n/2)}+x'_{((n/2)+1)})}{2} \\ n \bmod 2 \neq 0, x'_{(n/2)} \end{cases} \quad (9)$$

Where x' is shown in equation 10 and the collection of vector x in 11.

$$x'=sortascending(\vec{x}) \quad (10)$$

$$\langle \vec{x} \rangle = \{(\vec{x})_0, (\vec{x})_1, \ldots, (\vec{x})_{n-1}, (\vec{x})_n\} \quad (11)$$

These equations generate the median value by taking the middle value or the two middle values from the sorted vector and divide it by 2 in case it is an even number. This will select the median values for the Feature and stores it in the vector y.

The check itself again is fairly simple, just like with the other described single value Feature, the measured value x is held up against the generated median value y. If x is close enough to value y with a certain margin it will pass otherwise fail, formula 4 and formula 1 describe these steps.

3.8 Multi Value Features

As earlier mentioned an extension of the Single Value Features is the Multi Value one. This Feature can store more then one value per access point and thus allows for a more specific extraction of information. In general it is very similar to its single value predecessor outside of a few slight modifications in the general forms for the generation.

The general form for generation now is capable of handling an array of multiple vectors. Thus allowing for more then one vector value in y. This change is shown in equation 12, where the function z returns an array of vectors it generated from raw data x.

$$(\vec{y}_0, \vec{y}_1, .., \vec{y_{n-2}}, \vec{y_{n-1}}) = z(x) \quad (12)$$

The function z is different for each Feature and y is a collection of vectors with values. X is the raw data collected earlier. These values in the vectors y can then be used to compare to the values measured by the smart device.

While the comparison formula itself stays the same the formulas used to generate a comparison to the required values become slightly more complex. As now the formula compares it to more then one value, opposed to before which compares it to only one value per measured value.

3.8.1 Highest and Lowest Median Value of top and bottom percent

The simpler of the two Multi Value Features used is that of a highest and lowest Median value generated from the top and bottom percent of the raw data. Basically this will walk through the collection of raw data and find the lowest and highest value for each access point. The reason for this was that just taking the minimum and maximum values of the raw data was too prone to error.

The first formula described in equation 13, is that of a generation from the raw data.

$$z(x)=(top(\vec{x}),bottom(\vec{x})) \quad \textbf{(13)}$$

This basically looks through the raw data and finds the maximum and minimum value for each access point and stores it in a comparison sequence containing two vector indices. The first being the maximum measured value, the other the minimum.

The calculation of the said top and bottom values is done via the use of an algorithm. Because of flukes there might be unusually high and low readings, the system cannot just take the minimum and maximum values. For this reason it was needed to find a formula to obtain more realistic but also accurate values as was described earlier on.

The system does while making use of the following formula described in equation 14 for the maximum and minimum values.

$$top(\vec{x}')=\begin{cases} i \bmod 2=0, \frac{(x'_{(i-n)}+x'_{((i-n)+1)})}{2} \\ i \bmod 2\neq 0, x'_{(i-n)} \end{cases} \quad bottom(\vec{x}')=\begin{cases} i \bmod 2=0, \frac{(x'_{(0+n)}+x'_{((0+n)+1)})}{2} \\ i \bmod 2\neq 0, x'_{(0+n)} \end{cases}$$

$$\vec{x}'=sortascending(\vec{x})$$

$$i=length(\vec{x}')-1$$

$$n=\frac{i}{100}<10\,?\,10:\frac{i}{100} \quad \textbf{(14)}$$

In short we sort the values from high to low, then we take the top and bottom 1 percent. Since in raw data of shorter durations this can lead that you only have a few samples, the minimum length of samples is 10. From these values we take the median value that represents our maximum and minimum values.

During initial tests of the Feature over various forms of raw data, this seemed to give an accurate representation of most common maximum values and minimum values. This also filtered out the flukes represented by values that are caused by an erroneous reading or other factors that are not constant enough to be used in our Features. The reason for using the 1 percent mark is that for longer raw data pin point information there also is more data and more chance for it to contain inaccurate values. So as the raw data grows so does the generation of the Feature.

The Feature check itself is fairly simple as is described in equation 15.

$$result=\begin{cases} TRUE, y_1<x<y_0 \\ FALSE, Otherwise \end{cases} \quad \textbf{(15)}$$

If the measured value is between the minimum and the maximum value it returns a positive identification otherwise a negative one.

3.8.2 Random Values From Sample

Another multi value Feature is one only in theory; it is not generated but takes a random value for each of the access points from the raw data x and compares it to the measured values. In

essence this Feature is quite simple. The comparison even stays the same as those in the first Feature mentioned. It compares the measured value to the random selected y and calculates a percentage difference. If this difference falls within acceptable tolerances it passes, otherwise it fails. If it passes on a set number of access points the Feature provides a positive identification of the Pin Point.

The check itself is basically the same as every single value Feature; it just looks if the measured value is within the tolerance of the selected random value and returns a true or false if it does. For the formula behind these checks please read further up in the section discussing the single value Features.

3.9 Multi Measurement Features

The most versatile Feature type developed is the Multi Measurement Feature. This one requires more then one measurement from the smart device to give a positive identification of the Pin Point. This has as advantage that the system can include more identifying information in the Feature, and make it more agile. However it also will cost more time to get a positive identification as a certain amount of measurements have to pass the test rather then just one.

The general form for checking these Features is slightly different from the one used with Single and Multi Value Features as is shown in equation 16.

$$final = \begin{cases} TRUE\ ,\ if\ (\sum_{i=0}^{n-1} Result_i \geq finalRequired) \\ FALSE\ ,\ Otherwise \end{cases} \tag{16}$$

Where result is described in equation 17.

$$\begin{aligned} Result_0 &= \begin{cases} 1,\ f(\vec{x}_o) \geq required \\ 0, Otherwise \end{cases} \\ &\vdots \\ Result_{n-1} &= \begin{cases} 1,\ f(\vec{x}_{n-1}) \geq required \\ 0, Otherwise \end{cases} \end{aligned} \tag{17}$$

Since the check requires more then one measurement of the smart device, the general form of this Feature different. It is built up from the general form used in the earlier described Features. The difference being that instead of returning TRUE or FALSE, they return a one or a zero and store this in a results array.

After all the results are summed, they are checked if they pass or fail the final requirement. This is a number that indicates how many of the checks need to successfully pass for identification of the Pin Point. This last step returns the normal Boolean result of the Feature, which positively or negatively identifies the Feature for the Pin Point.

The general form of the generation formula is the same as that one of a Multi Value Feature. By returning a collection of y vectors that are used for each individual Feature. But even though the result is the same, the handling of this value in the checking formula is radically different from the Multi Value Feature.

3.9.1 Comparing to Random Values over Time

A Feature that came to mind that would use the Multi Measurement Feature was one that compares read values over time to randomly selected values from raw data. The generation of this process basically takes the signal strength at random times from the raw data and stores it in a collection of given amount of y vectors. This is shown in equation 18, which uses the random time values to select the signal strengths from the raw data.

(18) $$z(x)=[x(random\,time)_0, x(random\,time)_1, \ldots, x(random\,time)_{n-1}, x(random\,time)_{(n)}]$$

These values then need to be compared to the values measured by the smart device. Each time a new value is put in the Feature the old value slides forward to the next comparison position which in turn causes the value in that position to slide forward and so on. In total one would need about a less then the total amount of measurements to positively identify a Pin Point.

The comparison itself is described in formula 4, this basically is the formula filled in for each of the x values and measured value The only difference is now this formula is done multiple times and compared in a way described in formula 11.

3.9.2 Values over time

The more interesting Multi Measurement Features are those that use the standard single value Features described above to generate a Measurement over time. By using a sliding measurement window of values and comparing it to the smaller sub windows within the raw data the system basically uses a multi measured “single” value Feature over time.

For testing purposes we made multi measurement Features of the following types:

- Average Read Value over Time ***(3)(4)***
- Mode Value over Time ***(6)(7)(8)***
- Median Value over Time ***(9)(10)(11)***

Taking small time steps over the raw data you can create smaller versions of timed Pin Point information. These time steps can then be fed to the Single Value Features to create Feature information for each. The Features themselves are then stored as a consecutive array of Features, with each its own check and functions.

The check itself is even simpler, one can store the measured values in a similar array that slides over the Feature array, each of these values can be used in the Feature object to give a positive or a negative result with the current value provided by his time sample. With each new time sample all of the other samples slide over, the oldest is thrown out and the newest is added. The sum of all of the checks is then compared to the minimum required and if it passes the multi measurement Feature returns a positive pinpoint ID. This process is roughly described in Appendix A. In this diagram the first step shows the initial state of the Feature, the second provides it with new date replacing the oldest and sliding up the rest finally the third does the checks using the already coded single value Feature checks before returning the amount that passed. This number is then used to see if the Feature passes or with a minimum pass value.

The drawback of this system is that it might take a few seconds or even longer before identifying a pinpoint, but it has a big advantage in case the measured data gives an unexpected and erroneous value. Since such values are in the minority the sum of the checks still would filter it out.

3.10 Error Calculation

In the previous sections about Features we often talk about a margin when running the Feature checks. This is basically an error margin that exists when one measures data. Since not all rooms are considered equal in our theory, which also can be used to get a better identification of a Pin Point, not all Pin Points and Signals have the same Error Margin. Because of this the system will calculate a margin per raw data for a given pinpoint. This process is described in equation 19, which assumes that average value and mode value calculation over raw Pin Point data is available in Feature form.

$$error(\vec{x}) = \begin{cases} 10, if\ x' < 10 \\ 25, if\ x' > 25 \\ x', Otherwise \end{cases} \quad \textbf{(19)}$$

Where x' is described in equation 20.

$$x' = \left| (averagevalue(\vec{x}) - modevalue(x)) \right| \quad \textbf{(20)}$$

First calculated is the average value (3) (4) and the mode value (6) (7) (8), as we did in the earlier single value Feature, per Pin Point and per signal. The next step is subtracting the two values and taking the absolute to get the difference. The reason for using the average and the mode value is because the mode is the most frequent measured value while the average is affected by minimum and maximum values that are erroneous and thus create a difference. Since sometimes a Pin Point is located in a surrounding that has a characteristic low error measurement rating for certain signals, in these cases there is a minimum of 10 points allowed error. Some rooms however have the opposite for certain signals, for these cases the system has a maximum of 25. This top and bottoms are necessary because in our early tests we sometimes found values that wend to extremes either way.

This still allows for an error margin per signal per Pin Point while limiting the extreme cases in only allowing them a 25-point difference from the measured value.

3.11 Relevance of Features

After all the Features are defined it is safe to assume that not all Features are as capable in detecting a Pin Point as others. To not allow the worse performing Features to overshadow the better performing ones, a system was designed that would allow for weights to be added to a Feature. In the following section are about the decision making for weight values. Also it was discovered that some Single Value Features might perform better then initially expected.

3.11.1 Calculating what Feature has the most positive impact

To calculate a weight for a Feature the system ran for 94 minutes straight at a Pin Point that was known to be far from perfect. The whole time the system recorded the results per Feature and gave them a timestamp. With this information it was possible to see the performances of each individual Feature. These results are shown in figure 8.

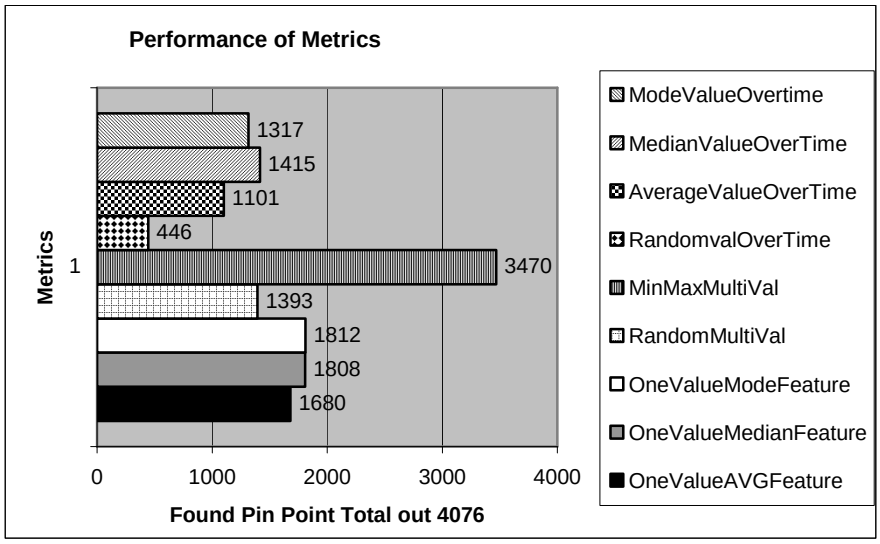


Figure 8: A graph showing the amount of positive identifications of a pin point during a stress test of 94 minutes

Surprisingly it shows that the Multi Measurement Features actually perform very poorly compared to their single value Feature counterparts. The reason for this is the huge fluctuation of values on the Pin Point we choose to test. It does however have another advantage, which also explains why it gives such a poor performance.

Since the values fluctuate extensively at this Pin point it often goes beyond the found scope for each Feature. Because a Multi Measurement Feature needs at least a few seconds to find it again, sometimes these Features cannot do this in time before the point starts to fluctuate again.

Now this is also an advantage at points that are not fluctuating as much. Because those Pin Points the less common fluctuation will not affect the other measurements as much and basically will be filtered out. From this one can say, that although a Multi Measurement Feature should have a lower importance than the Mode Value Single Value Feature, it should still be higher then the average which was often off in general testing and is more prone to be affected by error values.

Another point seen in this figure is that although the random value is not as often right as some of the other Features, it still sometimes is right when all the others are wrong. This is because of the inherent behaviour of the Feature and it having a chance of selecting the right value from its own erroneous values in the raw data. This was shown while plotting, shown in figure 9, the contribution of each Feature over time.

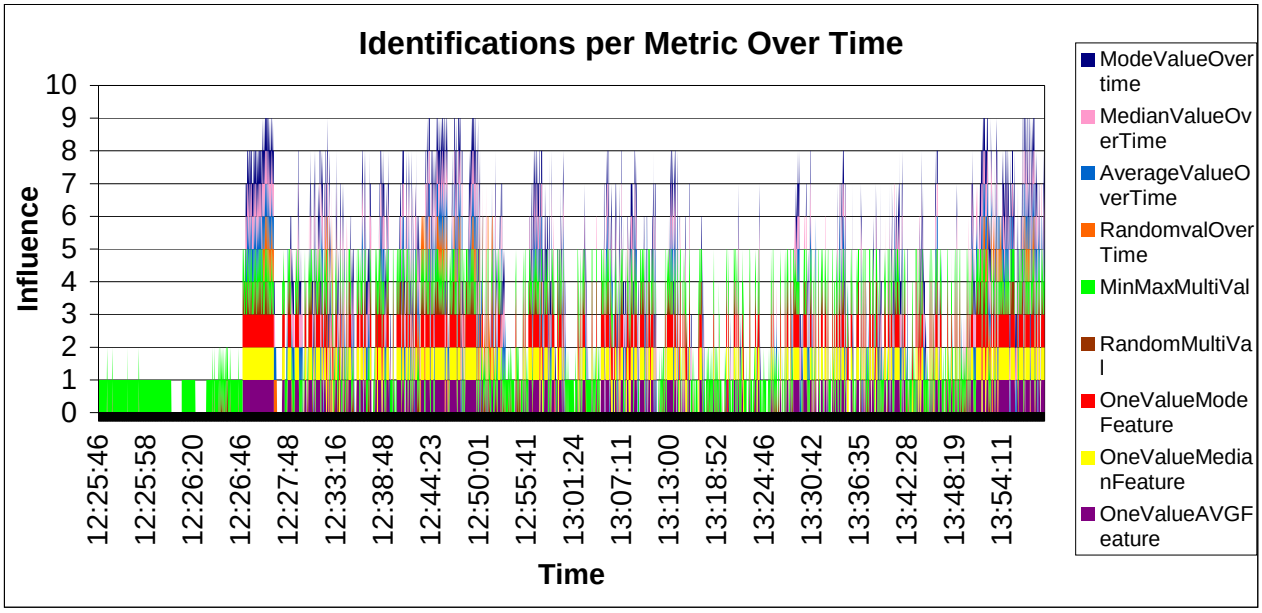


Figure 9: A graph showing the contribution of each Feature over time. Where a positive ID will add 1 to the height of the bar. The different colours indicate the different Features.

Basically each Feature has a colour and will be shown on the graph with the value of 1 if it returned a positive result, Features that did not return a positive result will not be added to the influence and thus are not shown. This value will be added on top of the other found Features to show its contribution and gives a clear view of the influence of a Feature over time.

It can be clearly see that the green colour of the Min Max Multi Value Feature stands out. This Feature is the most prominent in the graph. Another thing noticeable is that the Multi Measurement Features, although performing poorly in this particular setting, do not die out as fast as the others and sometimes even survive while other completely disappear. The random Feature also shows that during these periods it sometimes pops up with a valid identification. This demonstrates the already stated benefit in the earlier section.

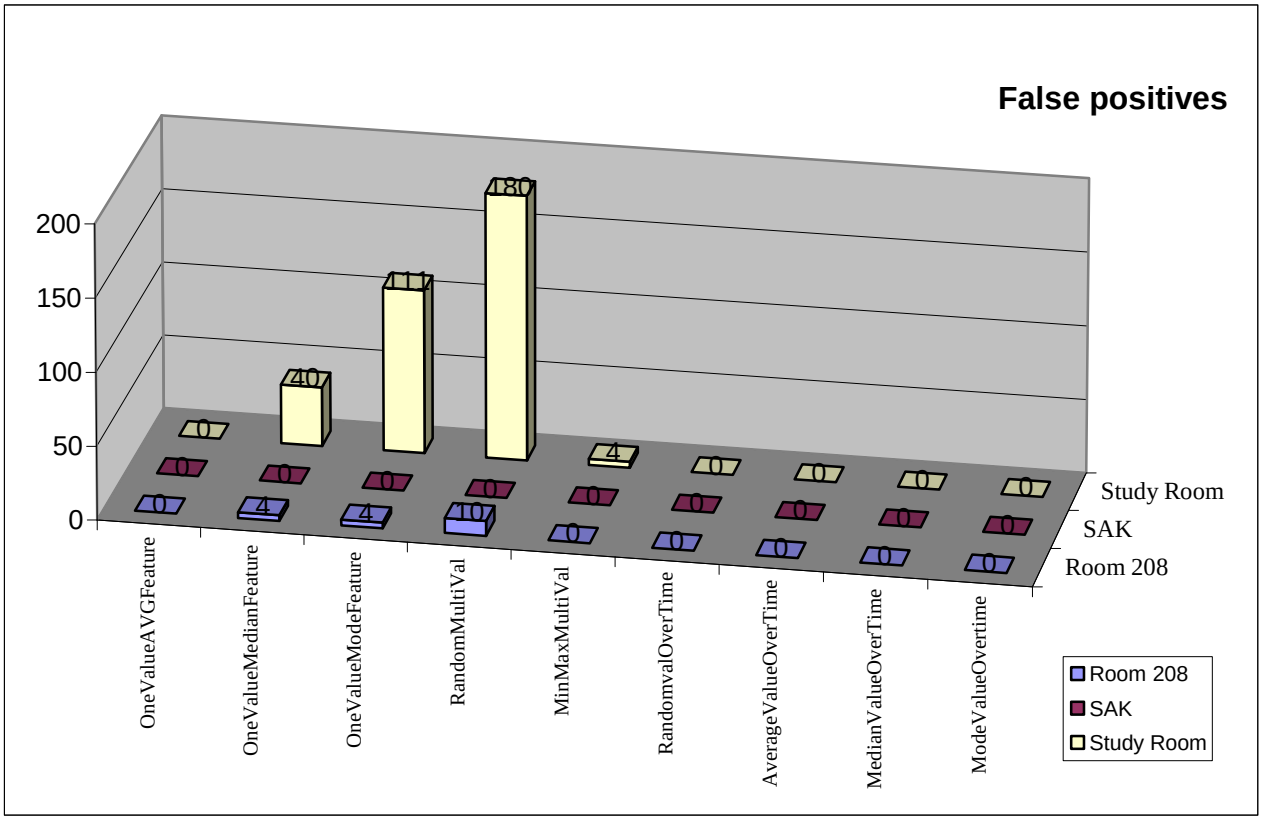


Figure 10: A graph showing the amount of false positives found per neighbouring room per Feature.

The next thing to take into account is the false positives generated by a Feature. Taking the 3 closest Pin Points to measuring Pin Point and seeing how often each Feature gave a false positive of its location gave us this value. To clarify this we show the results of this in a graph shown in Figure 10.

It was particularly surprising to see that the best performing Feature also had a low, near zero, false positive level. The Random Value Multi Value Feature on the other hand performed quite poorly compared to the others.

Taking into this into account the following simple equation, shown in equation 21, was devised.

$$importance(x, y) = x - y \quad (21)$$

Where x is the amount of proper positive identifications and y the number of false positive identifications.

There are several reasons to go for this formula as it also takes in account the number of false positives while still using integer numbers. By limiting the number of floating point operations we limit the amount of processing time required. A CPU is a lot more proficient in Integer operations than they are in floating point operations. Since the result from this calculation (an integer number) will come back every detection cycle, it is wise to spare its

power in this area. The usage of this value in the calculation itself will be explained in the next section of the paper.

3.11.2 Importance comparison while running

The importance calculation itself is fairly simple; it basically pre-calculates the sum of all the Features' importance values as a total. This total is then multiplied by a value between 0 and 1. This value is then rounded to a whole value that can function your minimum requirement if needed. This multiplier basically can be changed to receive different minimum requirements.

Basically what the system does is return the Pin Point with the highest value, there are then two methods, one takes the earlier mentioned minimum and checks it against that number to see if it is at least equal too or higher, the other just returns the name of the pin point with the highest value. The first one basically is needed to make sure false positives are curtained when walking close by a point but still not close enough that you want the system to identify the pin point.

3.12 Nearest Neighbour

One of the lessons taken from the researched work we researched is to take closer points into account over points further away. The closer points would be given a higher weigh then the more distant points. Calculated by equation 22, this weight is a number returned between 1 and 2.

$$multipliers = (1 + h * distances_1, .., 1 + h * distances_n)$$

(22)

Where h is defined in equation 23.

$$h = \frac{1}{Max(distances)}$$

(23)

The distance themselves are calculated and placed into a vector in equation 24.

$$\vec{distances} = \sqrt{(z_1 - (t_1)_1) + (z_2 - (t_1)_2)}, .., \sqrt{(z_1 - (t_n)_1) + (z_2 - (t_n)_2)}$$
$$t = (\vec{t_1}, \vec{t_2}, .., \vec{t_n} - 1, \vec{t_n})$$

(24)

1 being the Pin Point on the same floor that is furthest away, while 2 being the last detected Pin Point. This formula calculates the Euclidian distances between points and the last found point. After this it divides 1 by the largest distance to receive a linear value for each distance unit. This value is then multiplied per distance and subtracted from 2. This is the distance weight for each point in the system.

Since the number calculated above is a floating-point number, it will only be used to multiply the importance of each Feature after a completely new point has been found. This will make sure that the system does not have to do a floating-point calculation for each cycle, on top of the benefit for not having to calculate it for every cycle. The reason we choose for a 2 for the

last found point is to filter out false positives. It also has the added benefit of finding the Pin Point again after erroneous data has lost the point completely.

Since someone cannot go through floors or walks stairs as easily as going from room to room, every Pin Point on a different floor automatically gets the same value as the furthest point on the same floor. Also these points should not get a 0 weight either, because that would make it impossible for them to be selected, and it is possible for the system to be down for a large amount of time so one could have reached this point when the system comes back up.

3.13 Combining all techniques

The next step is to combine all of these techniques into the system. For instance by registering the Features by their logic classes we know what Features are active. When creating new Pin Points or loading them from a file, it automatically will extract the Features from this raw information as it knows which Features are registered with this class. It also contains a check that will check every Feature registered by comparing it to the measured values.

Afterwards a Feature either passes or not and the number of importance is taken from the registered Feature and added to the total value detected. Now if there was a new Pin Point that is different from the last one found, it multiplies this value by the found distance multiplier for that point and stores it for later comparison. If a new Pin Point was not found, the old multipliers stay valid; if there was not any multiplier calculated yet the total found importance will not be multiplied. After this either the highest point will be returned, or the highest point above a certain threshold.

More about the actual implementation in XNA, the Class Diagram and the whole logic is discussed in the paragraph “Implementation” which follows this paragraph. Here it also is discussed why we choose for a XNA and for a certain programming structure.

3.14 Creation of Pin Point

```
pinPointCreation(  TimeSpan timeSpan,String pinPointName, float[] location)
{
        DateTime timeCheck = DateTime.Now;
        WiFiInfo[] current;
        TimedWiFiInfo[] timedCurrent;
        PinPointInfo pinPointInfo;
        List<TimedWiFiInfo[]> rawPinPoint = new List<TimedWiFiInfo[]>();
        IntPtr handle = getWLANHandle();
        while((timeCheck.Subtract(DateTime.Now)).toMiliSeconds < timeSpan.toMiliseconds)
        {
                current = doScan(handle);
                int i = 0;
                foreach(WiFiInfo w in current)
                {
                        if(allowedSSID.contains(w.SSID))
                        {
                                timedCurrent[i] = new TimedWiFiInfo(DateTime.Now, w);
                                i++;
                        }
                }
                rawPinPoint.add(timedCurrent);
        }
        pinPointInfo = new PinPointInfo(location, rawPinPoint);
        File.Write(pinPointName, pinPointInfo);

}
```

Pseudo code 1: Pseudo code of the creation of a Pin Point

The creation of the raw Pin Point information can be described with the help of following pseudo code shown in Pseudo code 1. As shown it first initialises the WIFI hardware and asks for a handle with which to approach said hardware. The program also notes the time and stores to compare to later timestamps. With this earlier mentioned handle it asks the WIFI hardware to receive and report the different signals. This is then returned as an array of "WiFiInfo" objects, more about this object can be found in the section implementation. In short it contains all of the information about a signal the WIFI hardware can provide.

As these objects are received the program extracts the SSID and checks it with the allowed SSID's. If the signal is allowed it is stored in a so-called "TimedWiFiInfo" object and added to an array consisting of these objects. This basically adds a timestamp to the information. After all of the information has been processed in the "WiFiInfo" array, the "TimedWiFiInfo" array is added to a list.

After this the time stamp is checked with the original time recorded at the start. If the predetermined time has not yet passed the whole thing is repeated and more information is added to a list. If the predetermined time it is added to an object that also contains the Pin Points Location called PinPointInfo. When this step has complete the list is stored for later use. The list is also used to create Features, this is done both at the time of creation of a Pin Point and when the program starts in which case all of the stored raw Pin Point information is loaded from disk and handed to this process.

3.15 Creation of Features from Pin Point information

With the information gathered as described in the earlier section, one can create the Features needed for identification. This process is described with help from the pseudo code in pseudo code 2. In this process the method receives the information in the shape of a PinPoint Info Object. This object contains all of the information needed for the program to generate its Features. Since each Feature is able to create its values from a collection of TimedWiFiInfo objects the system only has to go by a list of empty Feature objects and ask them to return an object created from the TimedWiFiInfo provided.

```
Dictionary<String,IFeature> pinPoints = new Dictionairy<String, IFeature>();

FeatureExtraction(PinPointInfo pinPointInfo, String pinPointName, IFeature[] registredFeatureTypes)
{
        IFeature current;
        foreach(IFeature m in registredFeatureTypes)
        {
                current = m.doFeature(pinPointInfo.TimedWiFiInfo);
                pinPoints.add(pinPointName, current);
        }
        addToMap(pinPointName, pinPointInfo.Location);

}
```

Pseudo code 2: Pseudo code of the creation of Features from the Pin Point information.

The generation is done with help of the earlier defined formulae, each Feature has their own implementation with these in mind. These are usually defined as the z function in the formulae described for each particular Feature. In the pseudo code this is represented by the

m.doFeature function that returns, much like the z function above, a Features values. However it is in the form of an IFeature object.

This is then stored in a list that can contain objects adhering to the interface IFeature. Since all Features adhere to this list they can be added. This list is then added to a dictionary using the name of the Pin Point as its identifier. This dictionary is globally stored in the main program, for all functions to use when needed. This name is handed to this method via the call and comes either from the file name or the user when a Pin Point is created for the first time.

After this the system uses the location of the Pin Point stored in the PinPointInfo Object to place the Pin Point on the map and in the systems memory. This can be used by the system to do a distance correction during the identification of the Pin Point, but also when adding a Graphical User Interface to place the Pin Point on a map.

Again when the system is just started, this process is done for each of the Raw Pin Point information files saved in the working directory of the program.

3.16 Identification of Pin Point

```
String pinPointIdentification(String previousPointFound, int minimum)
{
	IntPtr handle = getWLANHandle();
	WiFiInfo[] current = doScan(handle);
	TimedWiFiInfo[] timedCurrent;
	String highestfound = "";
	int i=0;
	int highestValue = 0;
	int currentValue = 0;
	foreach(WiFiInfo w in current)
	{
		if(allowedSSID.contains(w.SSID))
		{
			timedCurrent[i] = new TimedWiFiInfo(DateTime.Now, w);
			i++;
		}
	}
	foreach(String s in pinPoints.Keys)
	{
		foreach(IFeature m in pinPoints[s])
		{
			currentValue += (m.doCheck(timedCurrent) * m.Importance) *
distance[s];
		}
		if(currentValue > minimum)
		{
			if(currentValue > highestValue
			{
				highestValue = currentValue;
				highestFound = s;
			}
		}
		currentValue = 0;
	}
	if(!(previousPoint.equals(highestFound)))
	{
		distance = recalculateDistances(s);
	}
	return s;
}
```

Pseudo code 3: Identification of Pin Point

After all of the information has been generated and created in the earlier points, the system can do a check if the found data passes a Feature or not while returning a score. The steps to this are described with help of pseudo code 3. The first step, like when gathering the raw

information, is to initialise the hardware and receive a handle. With this handle the program then asks to return scanned information.

As before it uses the allowed SSID list see if an SSID's signal strength is allowed, the difference this time is that after it creates a list of allowed signals currently measured it hands it to the first Feature of the First Pin Point of the global dictionary. This Feature then checks the values with its own check. Before returning either a true or a false for passing. These checks where described in the earlier formulae usually by the function f. These checks are done with the error calculation in mind. At maximum the found values can only be deviant from the Feature's values the values of the error calculation in formula 13.

After this each 1 or 0 (true or false) is multiplied by the importance of that particular Feature, which is stored inside the Feature itself. This value is then again multiplied by the distance correction for that particular point, which at a first run is equal for all of the points. When a proper point is found this correction is updated and stored.

Finally this value is checked against a minimum threshold, if it passes then it is compared to the highest value found so far. If it beats both it will be stored as the new highest value found. When all of the Pin Points have been checked the name of the highest value is returned as a String. If the previous point however is different from the new found point, all of the distance corrections are recalculated using the earlier described formula 15. Of course when no Pin Point is a good match the system returns an empty String.

4 Implementation

4.1 Software Design

The graph, shown in figure 11, shows the design of the system in a generalized and abstract manner. The general design of the program with the WiFi API, Container Classes and Pin Point functions each being separate DLL files (Dynamic Link Library). The reason for this was the need for a modular system with separate functions. The advantage behind this is that each of these parts can be changed without affecting the other parts. Even if a new version would contain a bug, the rest of the system would still function provided one used the old library instead.

This for instance came in handy when the amount of Features started to increase. With this increase the need for an automatic system to add Features over a manual one became apparent. With the new system, we could just add the Features to the Pin Point functions, register them to the Feature Logic class and that class would take care of the rest for each Pin Point available.

Another advantage is that when there is an error or bug in one of the sections, the system can just work with an earlier version. The program itself still works even if it is with an older version of that DLL.

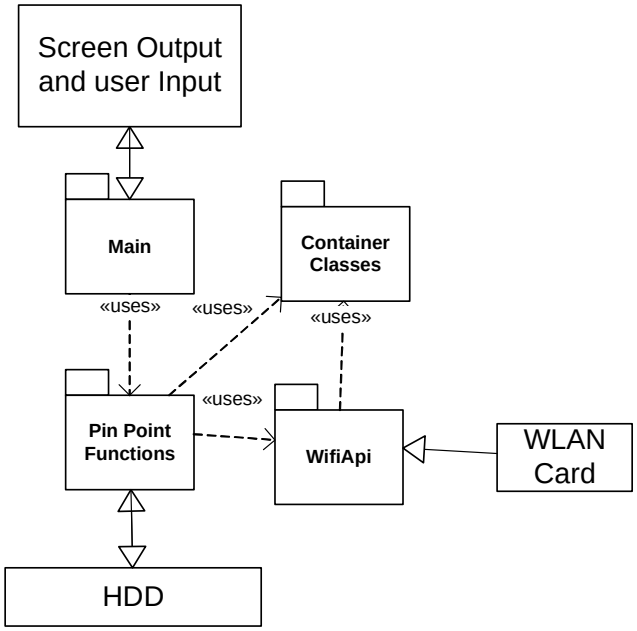


Figure 11:The General System design.

As shown in figure 20, the program consists basically of four main sections. The Main sections consist of the Feature Logic, Main Program and the Graphical UI. The Container classes mostly contain all the data types and structures. WiFi API, is the API that interacts with the WiFi card as to provide the information and measurements needed for the simulation. The last one is Pin Point functions; this one contains all the Features and a helper class that allows for easy Pin Point operations.

4.1.1 Main

The Main program contains mostly the main program, logic and GUI of the system. This program basically is the manager of the whole system. It uses Pin Point functions it has access to all of the separate functions through an abstraction layer. This layer would allow for easier coding and cleaner code.

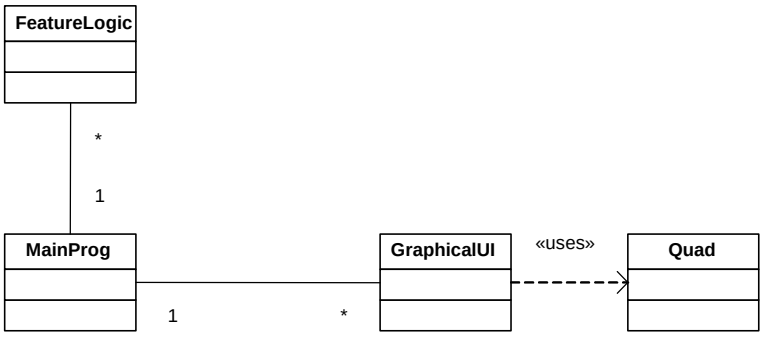


Figure 12:The Main section

As shown in figure 12, the Main section consists out of 3 important parts and a Quad Structure used for our textured map. Each of these parts has their own very specific function as defined bellow.

Main

The Main's function is like the top manager of the company. It basically starts the different processes and keeps control over the main logic loop. This program starts the Graphical User Interface, registers the Features with the Feature Logic and also acts as the communication between the Graphical User Interface and the Feature Logic classes.

Shown in Appendix B, the main function has the following process: first it starts the initialisation, which initialises both the Feature Logic as the Graphical User Interface. The latter is started in a separate thread so it can run independent from the rest of the process. After this the system starts to register each Feature type with the Feature Logic class. It is

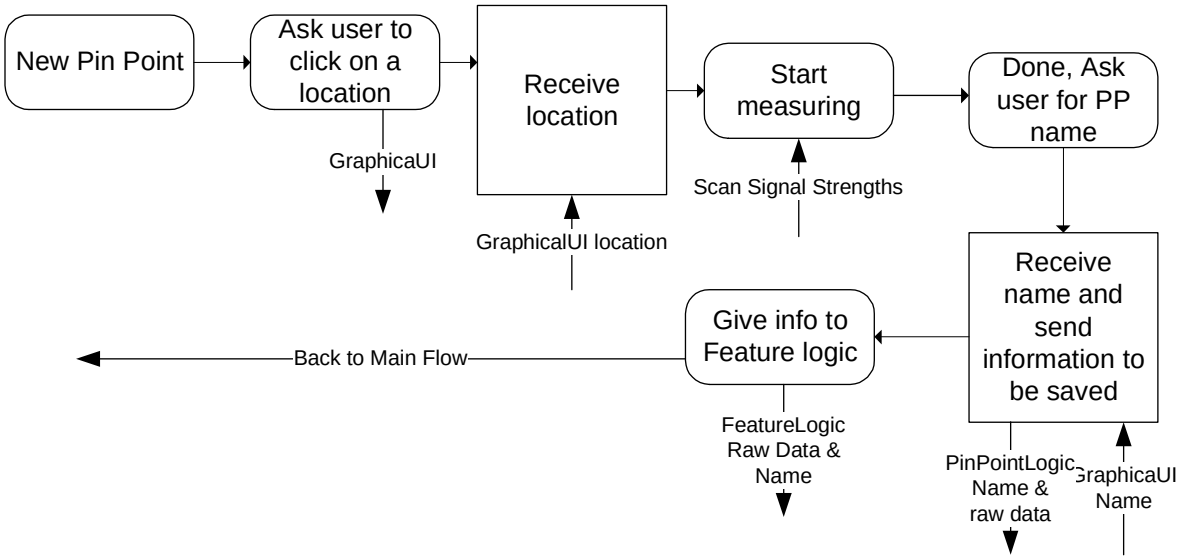


Figure 13:New Pin Point Flow

because of this that the Feature Logic knows which Features are present and will initialise

each one of them given the raw data of a Pin Point. It also registers the different allowed signals, which in this case are GeoLoc1, GeoLoc2 and GeoLoc3 with Feature Logic in a similar fashion as the Features themselves.

Following this it starts the Main logic loop, by first using Pin Point functions to load all of the Pin Point information from the disk. After this it hands this information to Feature Logic. While this happens it waits for the green light of the Graphical User Interface and the Feature Logic that they are done with initialising. After this is done the system waits for the user input to know what simulation to start or if he should create a new Pin Point. It is also possible to start a diagnostic mode that will print all of the signal strengths to the command line.

If the System gets the order to make a new Pin Point it will ask the user to click on Pin Point location. The system will then command the Pin Point Logic inside the Pin Point Functions section to start receiving the information from the WiFi API. This Logic will do so for a set amount of time before returning a collection that is the raw information for that point. Here after the main will ask the user for a name for the said Pin Point. It is with this name that Pin Point logic will write this information to the disk. This information is also handed to the Feature Logic so it can extract the necessary information from it. This is shown in figure 13 in a sequence of processes.

The other step is to start the simulation, which can be either the simulation described in this paper or another. The other simulation would be described in the paper of my research partner. The simulation runs mostly inside Feature Logic, and will return a point by its name. The name of this point is then handed to the Graphical User Interface accompanied by a colour. Which in turn will turn this specific point to the colour indicated and thus main provides the communication between Feature Logic and the Graphical User Interface. All of the above logic is describe in Appendix C by a flow diagram.

Feature Logic

As earlier mentioned, Feature Logic does most of the heavy work. This class manages and creates instances of every Feature registered, run each simulation and handles the checks of each Feature. Basically Feature Logic is the one that does most of the work for the system related to Features.

As Features are registered with Feature Logic, it keeps a list of empty instances of these Features for it's own use. The moment new raw data is delivered to it; Feature Logic generates the proper Features for this raw data and stores it for later use. As the main function provides Feature Logic with the raw information, Feature logic itself will handle the rest. Shown in figure 14 is the process of creating Features from raw data.

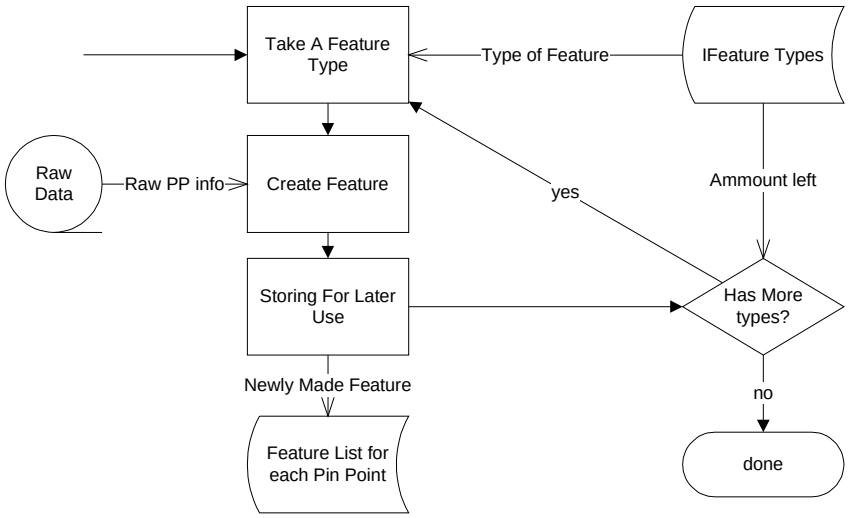


Figure 14: Flow chart of the Raw Data to Pin Point process. The closed arrows indicate flow while open arrows indicate a data flow

It also handles the simulation itself. If provided with signal strength it will check any Feature it generated. Afterwards Feature Logic factors in the importance and distance multiplier as described in the theory behind our system and will select the best score. Now Feature Logic can return either the best score or the best score over a given minimum. If the minimum is not met it returns an empty String. The steps described here are shown in Appendix D. with the help of a flow chart.

Since, as earlier mentioned, there can be a minimum value threshold but it can also be omitted for test comparison. In the testing itself the absolute values are shown without a minimum. The minimum value will be discussed during the testing itself. As minimum values are an interesting research topic in its own right.

Since most of the checks itself are done within the Features themselves, Feature Logic only has to provide them with the value measured from the WLAN API. This kept the code compact and easily portable for the use of adding more Features in the future or removing old ones that are not living up to expectation.

Graphical User Interface

The Graphical User Interface basically does what its name describes. It uses the XNA toolset to provide a 3D accelerated UI for the user. It is started on a separate thread and basically has one separate object called quad to create a plane on which a map of the environment is projected. After it has been initialised it waits for the Main to provide it with the point information to draw the points on the earlier mentioned quad. This basically runs in a separate thread. Figure 15 shows our GUI while running the simulation.

Outside of sending commands to the Main via the “XNA Console” described earlier, it mostly is a passive process from the rest of the system. It changes the camera internally from the mouse position and waits for changes to its point directed to it by the Main. A precaution to make sure the points drawn on the map are kept up to date, which often is a problem in multi threaded applications, the main writes it’s information to a separate buffer. This buffer has a semaphore that will indicate when the information is updated. In turn when this semaphore indicates that writing is done the graphical user interface can read the updated information from this buffer.

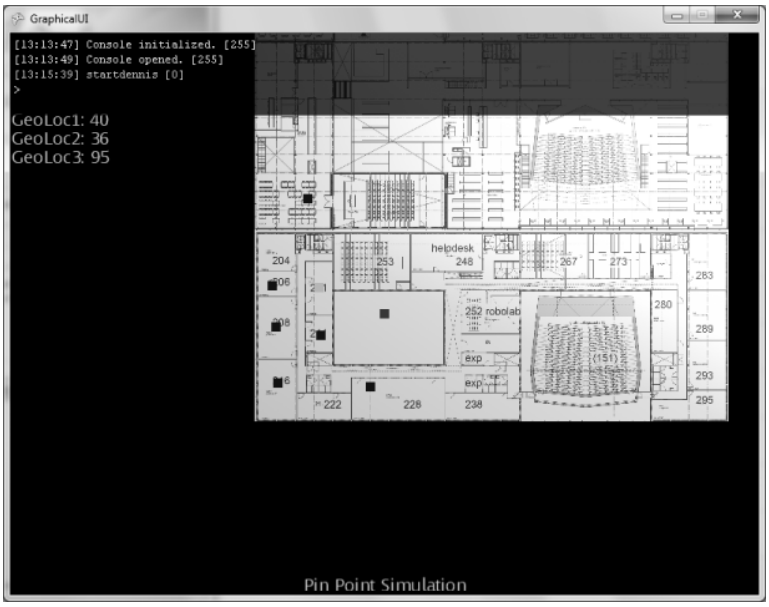


Figure 15: The GUI of the Simulation

In case the semaphore is set, the Graphical User Interface will not read from this buffer. Instead it will use its own buffer with the latest updated information before this semaphore was set. After GUI detects that the writing is done via the semaphore, it will read out all the updated information and changes it’s own buffer of points. Afterwards it will set the semaphore to a value that indicates it is done reading and new updates can be added if needed. The main will not update the information if this semaphore is not set, as to not create deadlocks but also not to add information while the updates are being read.

Now if the Graphical User Interface has detected a command being entered into its console, it will cause a listener to act on it setting a command, which in turn will cause the Main to read that specific command and detect what the user has instructed it to do.

Most interactions however are via the earlier mentioned buffer, which handles the points and text information.

4.1.2 Pin Point Functions

Another part of the system, which was integrated via DLL are the Pin Point functions. This DLL provides the system with helpful tools and methods for the raw data. It also contains the Features themselves. The function of this DLL is to provide all basic functions regarding Pin Points and Features. It contains of 2 major parts one of which is a collection of many smaller parts. The main part is a class filled with functions and tools to help you build, load, safe and work with raw Pin Point data. It mostly is an abstraction layer for the main program. The other is a collection of Features based on an Interface IFeature. This one contains classes for every Feature described above and can easily be expanded via the use of this interface. The general design of Pin Point functions is shown in Appendix E.

Feature Interface and Feature generation

The Features are created with the use of an interface. This interface dictates the minimum functions needed to make a proper Feature. Each Feature has to at least implement these to make it compatible with the system. The functions allow for the creation of a Feature, providing the return of a Feature given a set of raw data, returning all of the necessary data and variables and last but most important providing a check. This check returns a passed or failed Boolean value, for that specific Feature given a current measurement. The design and methods required plus the structure of this set-up is shown in figure 16.

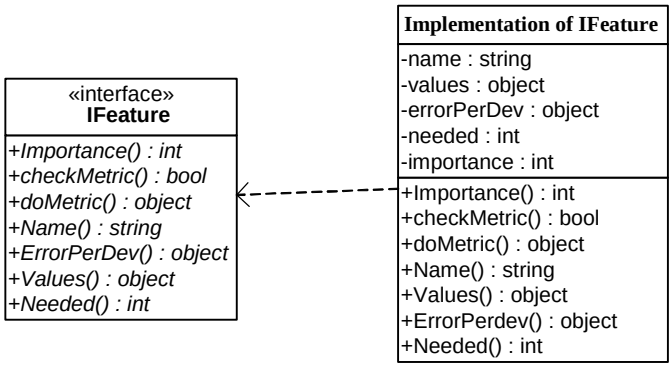


Figure 16: An example of how the IFeature Interface is implemented. By using it's own values and the properties (C# properties instead of getters and setters) to quickly make a Feature.

As shown in this Figure, IFeature consists of a few basic functions to provide the earlier mentioned:

- int Importance, A getter and Setter for the importance value
- Bool checkFeature, this is the method that checks a Feature when given an array of measured values of the type TimedWiFiInfo. More on this type in the section about Container Classes
- Object doFeature, this is the method that will return a Feature of the same type when given suitable information. This can also be done in most Features by handing this info directly to the Feature when creating a new one. But that is not part of the minimum required specs
- String Name, a getter and setter for the name of the Pin Point this Feature was created for.
- Dictionary<String, float> ErrorPerDev, getter and setter for getting the error allowed per Accesspoint.
- int Needed, the getter and setter for getting the minimum amount to be passed for this Feature Type.

When provided with these points the newly added Feature has enough implemented to be standard and is compatible with the system. In this set up IFeature allows for the creation of standard Features that would also be compatible with the system in a timely fashion. It would just need to be registered with the Feature Logic unit in the main program and Feature Logic will take care of the rest.

Pin Point Logic

The more cumbersome of the two parts contained in this part of the system. It basically provides an abstraction layer for easy interaction with the raw data and the generation of this. This ranges from the creation of a Pin Point to the interaction with the WiFi API. Each of these function are discussed to some detail as they are vital to the interaction, creation and maintaining of Pin Points.

AllowedSSID

This function basically is a getter and setter in which one can register the name of an access point by adding it to the list. The rest of the system then uses the list. This to make sure that the raw data and measurements are not contaminated by other access points not part of the system. During a test a signal was encountered that used multiple repeaters. This signal gave roughly the same signal all through the building. This signal was not useable for localization of our device and only contaminated the measurements.

calcErrorValues

This function calculates the error values from the raw data provided. The calculation itself is described in the section about error calculation in Proposed method. It returns a Dictionary object that contains each error allowed per SSID.

loadPinPoint

This function loads the raw data from the disk using an XML based approach. This approach is a lot slower and takes more space then the later implemented binary approach. That function is called pinPointFromFile.

savePinPoint

The exact opposite of loadPinPoint, it saves the raw information to a file on the disk with the same name as the Pin Point. It also contains information like location of the Pin Point and all of the raw information collected. It too is in XML that takes more space and time to save then the later added savePinPointBin function.

pinPointFromFile

This method tries to load one hotspot from the disk with a given name, it does so in a binary format that was added after finding out that the system was quite slow in saving and loading huge amounts of XML based data. It made the system about three times as fast but also the files where about ten times as small.

savePinPointBin

The exact counter of the earlier mentioned pinPointFromFile function. It basically saves all the information including the location on the map and raw data onto the disk in a binary fashion. This is a lot more efficient for the simulation then the use of the XML based system.

LoadPinPointsBin

This functions loads all the available Pin Point information located in the working directory of the system. This is a handy function for the system as it provides the main with all the information it needs in one method to create the Features needed for the simulation. On average this function is about three till four times as fast then the earlier XML based approach when loading all the files.

MakePinPoint

Maybe most important function contained within Pin Point Logic. It basically listens to the WiFi device and collects all of the necessary information to create a Pin Point. This function does the listening for a set amount of time and filling the raw information for each allowed SSID into a list of Dictionary objects.Tthis Dictionary contains a String with the name of the SSID and a TimedWiFiInfo Object. TimedWiFiInfo contains all of the information for that

particular signal plus a time stamp. When the time is reached this is returned to the main which then asks the user for a name for this Pin point and a location before giving it back to Pin Point Logic to save to the disk. This list is also given to Feature Logic to generate Features.

doScan

A test function which returns all found values of each SSID to the command line, this is mostly used for testing the hardware and the system. It is used to see if listening to the hardware will give the proper values or if the hardware or WiFiApi is malfunctioning.

4.1.3 WiFi API

This basically contains all the code to interact with the WiFi Hardware of a device. This basically is a driver, which will return all of the signals received by the hardware. It was written mostly in C# while using some C++ based keywords to create a layer between the two.

Using Pinvoke, intptr and other structures provided by C# it is possible to interface with C header files and to approach otherwise native code that is nott yet ported into the .Net standard API for C#.

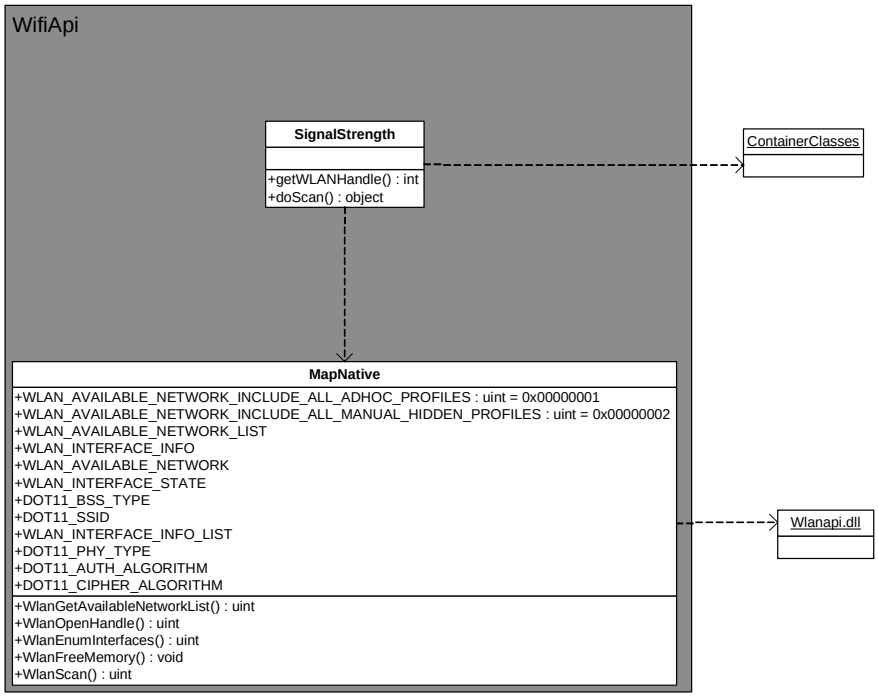


Figure 17: The structure and methods of the WiFiApi, with the WiFi API itself are inside the blue square. Outside showing Wlanapi.dll (a Windows DLL) and ContainerClasses as being external

WiFi API provides a managed driver to the hardware, after being told to return all of the info needed it will use container classes to return an array of WiFiInfos. This is an object from container classes, which contains all the information available for a certain SSID. Shown in figure 17 is the basic structure of this DLL. Although this DLL is small, it is vital to the functionality of the system.

4.1.4 Container Classes

Figure 31 is a class diagram of the classes contained in this part of the system. It basically provides different types and objects that contain data and other information necessary for the system to operate. As shown in the diagram it contains the following classes:

- PinPointInfo
- TimedWiFiInfo
- WiFiInfo

Each of these is basically a container for information. It is only a small DLL mostly containing data structures. PinPointInfo contains all of the raw data in the form of TimedWiFiInfo objects and the location on the map. This is used by both the main program as the Pin Point logic, to extra the location and Features from. The structure of this DLL is shown in figure 18.

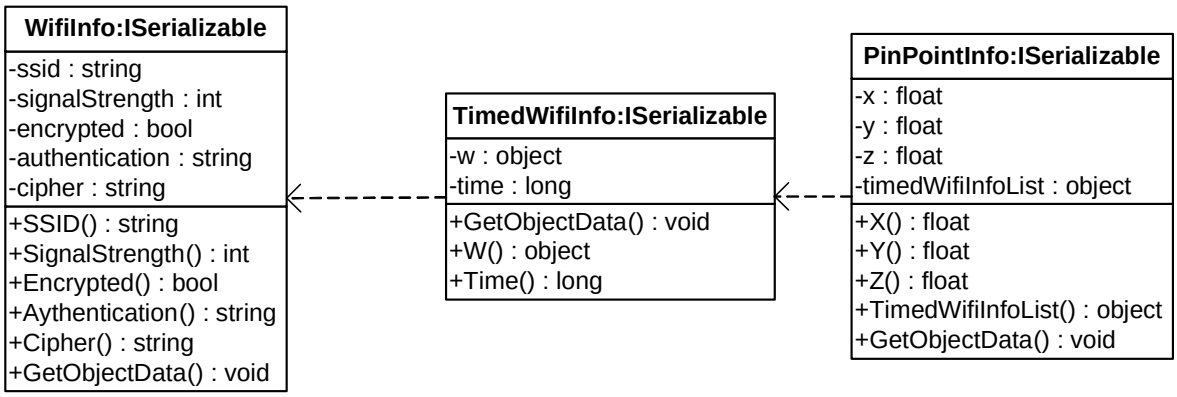


Figure 18:Class Diagram of the ContainerClasses DLL. This class containers data containers each of which contains a superset of the other where WiFiInfo being the simplest and PinPointInfo containing the most

WiFiInfo is the object that contains all the information from an access point. This ranges from signal strength to encryption. After the WiFi API receives its data from the hardware it puts them in an array of these objects. TimedWiFiInfo is similar but it contains a timestamp from when this particular information was requested. This is necessary for the generation of Features when there is a measurement over time.

4.2 Software development

More then just hardware, measuring points and Access Point positions, there also is the issue of software and the development of the software. To develop the system several design and develop decisions had to be made. While researching the problem several different systems where tried in various development environments. Even a Wiimote was tried out only to find that its sensor capabilities are not of a quality needed for our research. From these tries decisions where made because of experience in developing in this environment, the documentation and support, the tools provided and finally the quick development/test cycles the environment allows for.

Other choices where: a multi-threaded approach for our problem, separating the measuring, logic, helper functions and Graphical User Interface into separate threads. Although this caused trouble for our older Notebook, the younger HP actually had a speed increase as it could spread the work over its cores. More about the programs design and decisions follows after a more in dept description of the environment decisions and development tools.

4.2.1 Visual C# Express 2008

Because of earlier experience with the Visual Studio packages, it was decided to look into the “free” express editions [17] of this development package. Since mobile development was not needed at this time, the free versions of this environment would suffice. During this development it proved to be more then capable in handling the development of our system thus not warranting the purchase of a paid license. The debugging tools and code management

enough support for our project. Even the migration to a more XNA based program and development provided little to no problem

As earlier stated previous experience in developing in this environment was present. Especially in using the C# language and the Windows Environment regarding native libraries this was a plus. As it was needed to write a natively coded WiFi API based on the DLLs it would only make sense to approach the problem from a perspective one is familiar with. The other plus for this tool was the documentation online. This is provided by both the MSDN network and by people providing excellent tutorials. So more often then not a problem would be solved quickly with the help of MSDN or a Microsoft Blogger providing his insights into a solution.

4.2.2 Microsoft XNA Game Studio 3.1

During the development the system was mostly text based. As the system grew, so did the need to zoom in and out on a map and place points. Wanting to spare the CPU for the actual workload it was decided to migrate the project to the XNA Game Studio Platform 3.1. XNA Game Studio Express 3.1[18], henceforth XNA, was mostly used for the development of our Graphical User Interface.

Although the environment was mend to build Hardware Accelerated games for the Xbox 360, Zune and Windows platforms, the same development kit can easily be used to build a 3D accelerated environment for a GUI. Since XNA is completely in C# and using the same structure as the rest of the system, the migration to this platform caused almost no problems

XNA is more then just a graphical toolset, it also provides mouse and keyboard listeners handy for the system. This we could use to create a camera that moves with the input from the mouse. Since XNA includes many tools to import different image files like PNG, the floor plan could easily be imported into the UI for reference.

As the developing progressed the need for a console in the same window as the GUI increased. For this reason we used an open source free to use tool for XNA. This tool allows for the creation of a full command area defined in any way possible. Community support like this saved us a lot of time and effort to build our system rather then to reinvent the wheel ourselves.

5 Experimentation

This section in the thesis focuses on the hardware used and software design used to test the theory and system. It is mostly to give some insight on the environment and set-up used, but also to explain what was tested. Further more it explains why the choice was made for certain development environments, languages and the general structure of the system.

5.1 Notebooks

The test environment ran on 2 notebooks, which are of different age, and gave each their own role according to the capabilities of the hardware.

5.1.1 Medion 6200

The first notebook is a Medion 6200 Notebook, it the older one of the two with a 15-inch screen with the following specs:

- Intel Pentium 4 2.4ghz
- 512 MB DDR2 Ram
- 80 Gigabyte 2.5 inch HDD
- Geforce 5200 Mobile
- Creatix CTX712 WiFi Adapter
- Running on Windows XP

Figure 19: Medion 6200

In initial tests it showed that it had some problems running the simulation, mostly because of the multi threaded architecture of the system. Normally this would not be a problem as it would send the measured signal strength to a server who does all the processing, but since in this case the server was also the machine doing the actual calculation on a second thread while running a 3D accelerated User Interface for the graphical interface on a third thread, it started to choke and even starve the measuring thread just to keep the others running.

Because of this starvation the Notebook, shown in figure 19, was used as the third access point in the simulation with the name “Geoloc3”. In this role it would broadcast this SSID and was very capable of providing a steady signal. Its location will be discussed in the section dealing with the access points and their locations in the simulation.

5.1.2 HP Touchsmart TX2650ed

The second Notebook is a HP Touchsmart TX2650ed; it is the newer of the two and is a 13.3-inch touch screen Notebook/Tablet PC with the following specs:

- AMD Turion X2 ULTRA (ZM82 @ 2.2ghz)
- 4GB DDR2 Ram
- 250GB 2.5 inch HDD
- AMD/ATI Mobility Radeon 3200
- Broadcom 4322AG 802.11a/b/g/draft-n adapter
- Running on Windows 7 32 bits.

Figure 20: HP Touchsmart TX2650ed

Of the two notebooks, this one ran the system a lot better then the earlier mentioned Medion. The Notebook, shown in figure 20 became our main platform for testing the system. It had the added benefit in being smaller and thus easier to carry.

It had a better GPU as well, which provided better support for the Graphical User Interface used for the presentation of the system. As it was needed to zoom in and out of the map in real time while showing the latest position, it was far more interesting to use a Graphically accelerated Interface then one just using text. Because of this User Interface all of the above mentioned features are supported without taxing the system.

5.2 Access points and Locations

This section talks about the location and specs of the access points used and why these access points where placed at these locations. The reason for naming them individually and registering them with the system is that in the building there are signals with multiple antennas and repeaters. These signals have signal strengths, which only fluctuate slightly through the whole building. This would influence the simulation in a negative manner, because it would always pass on that particular signal. To counter this we acquired two access points further described below.

5.2.1 Access point specs

As access points the simulation used the cheapest router we could find. These routers are of the Belkin brand, namely the Belkin Wireless G with the type number F5D7230-4 v7000. The SSID's of these routers where named respectively GeoLoc1 and GeoLoc2. These where then placed at separate locations in the building. These locations would be closed off when nobody was attending, both for the safety of the router and to rule out tampering with the routers. Figure 21 is a picture of the router used.

Figure 21: Belkin Wireless G router

Outside of a slight signal loss, the routers performed their function of just transmitting their SSID. For the purpose of testing the system this was ideal since it would allow for tests in a less then perfect environment with a less then perfect router, thus simulating actual conditions in which the system should work. It also helps in defining the problems when using less then perfect conditions both hardware wise and with people walking around when the system is running.

5.2.2 Location on Map

Every router is located in a specific location, this is important for the simulation as they should remain at this given position for the whole time. The routers also needed to be behind locked doors when left unattended, for this reason two rooms where found. One router was placed in the Repro room on the first floor, while the other found a home in the Robo Lab on the second floor.

These positions are also unique and spread apart from each other so that each signal is degradation is unique. A problem with the system is that if you place all the routers/access points in the same room, then system will have trouble devices. This is because the signals degrade almost as if there is only one router. The third “access point” was located in the same room where we worked. The Medion notebook earlier described provided this signal.

5.3 Map of the area

Since the tests are only on the first three floors of the building we only made a map of these three floors. As the strength of the routers only cover a section of this map, the system is only tested in this section. This is because of the limited range of our routers. The system does however support registering more routers, so to cover the whole area would be a possibility just not needed for the test. The maps where provided by the website of the university in PDF format and where converted into textures..

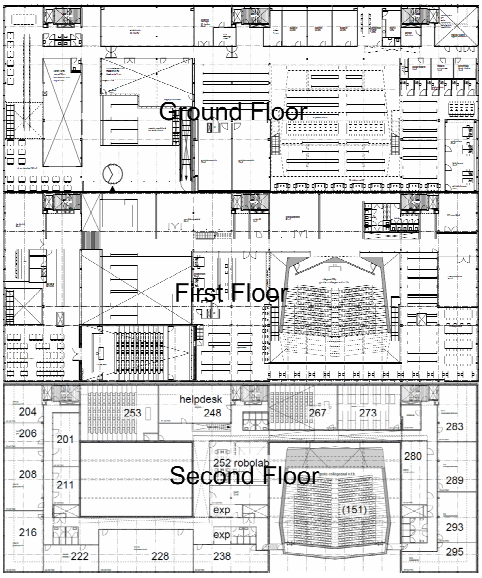


Figure 22: Map of the Bernoulli Borg, courtesy of the University of Groningen

Figure 22 shows the basic layout of our map with the ground floor being at the top and the second floor at the bottom. The user interface also uses this texture as do some of the graphs used in the results section of this thesis. The reason being to keep the map consistent with the map used by the university itself.

First thing noticeable about the map is the big open space in the middle of all three floors. Originally it was assumed this could be a problem but it proved to be non-existent as it hardly interfered with the range of the devices. This also shows the intricate structure of the building and that it is a good test case to try our system in.

5.3.1 Access Point Locations

The access points themselves are located in several different locations, each of these in a different room and one on a different floor. The first one, GeoLoc1, is located in the Repro room on the first floor. This was done with permission of the Operator of the Repro room. He would keep an eye on it and lock the room when he was leaving. The location is shown on the map in Figure 23.

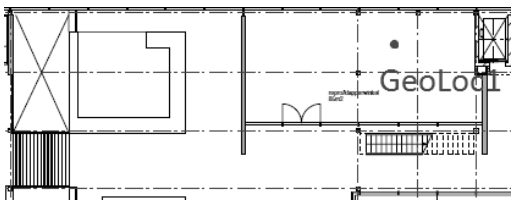


Figure 23: GeoLoc1 location

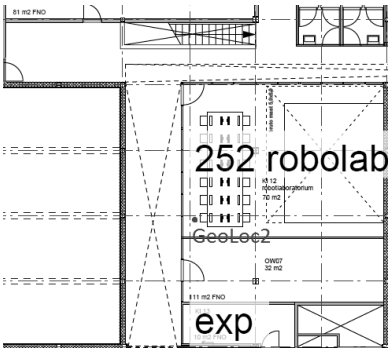


Figure 24:GeoLoc2 location

GeoLoc2 is placed in the Robo Lab, this was done with the permission of the man responsible. This location was chosen to provide for a completely different location from the Repro room while still having a room that is under surveillance. The location of this room is shown in figure 24 and is on the second floor of the building.

GeoLoc3 also known, as the Medion 6200 notebook is located in the same room where most of the coding and early tests where done. These positions where always under surveillance mend that there was no outside control over the hardware outside of the people responsible for the earlier mentioned rooms. Another good point of this location was that it was on the complete opposite of the room GeoLoc2 was in. The location is on the second floor and shown in figure 25.

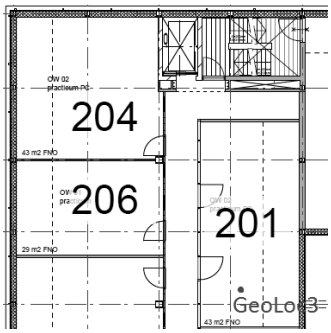


Figure 25:GeoLoc3 location

5.3.2 Location of Pin Points

Next issue was the position of the Pin Points as these points will be the calibration points. It was decided to have one per room in the place where we where going to do our tests. There is one Pin Point on the first floor located in the Cafeteria to demonstrate that the system can handle multiple floors.

The location of these points are shown in figure 26 and basically are set up as a good test for walking from point to point and to see how well the system can track the measuring device. Also since it surrounds our working area we can easily test new changes to the system without having to travel to the test area this limited the time needed setting up an experiment. It also allows for a quick turn around for testing and bug fixing in the software.

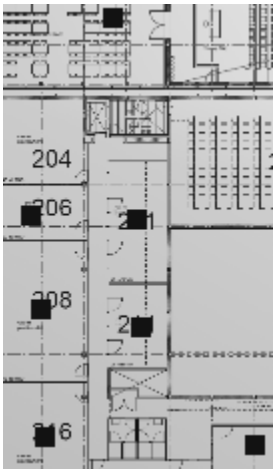


Figure 26:Pin Point Locations

5.4 Various Feature tests

Because it was needed to test with different situations, to show the increase and decrease in usability, certain tests where planned. The following section talks about each of these in finer details. Although part of this was already done in the section “Approach – Relevance of Features”, it would be a good idea to revisit these results when talking about the results.

5.4.1 No Importance or Nearest Neighbour

The first test planned is one that has does not have a Nearest Neighbour correction or any importance to the given Feature. This would give a base line with just showing the different performances of the Features, but also show the base performance of the system and allows for a nice comparison compared to the test to follow.

5.4.2 All Features Equal Importance

Another test that is important to run is to run all Features with the same importance while keeping the Nearest Neighbour correction. This will show if the Nearest Neighbour will help and to what degree. It is expected that it will mostly be an improvement while being on the move and making sure you don not get as many false positives. It also is a demonstration that it has a more stable point once found and not a fast switcher. This is expected because main point found would have a higher weight then those surrounding it. Thus this will help us with a more stable location of the device.

5.4.3 No Nearest Neighbour correction

Of course the second to last test is to have an importance for each Feature but no Nearest Neighbour correction. The idea behind this test is to show the improvement of having different weights for each Feature. It is also expected to be a more stable test then the uncorrected test. Which means less switching between neighbouring points, but also that it finds points faster. Again these will be compared to each other and the test without any importance or neighbour correction.

5.4.4 Complete System

The last and most interesting test is the complete system. This includes all of the Features, corrections and importance correction implemented and running. This then needs to be compared to the earlier tests described. This comparison will show how much of an improvement these corrections actually are.

5.4.5 Minimum

There will also be a discussion in the results about a minimum value. As this can help against false positives but can also make it more difficult for a point to be selected. It would be interesting to see the effect of different minimum values on the tests. It is also interesting to see what a good minimum value would be and if this value differs between the path test and the stress test.

5.5 Testing with two signals

5.5.1 Why test with two signals

Another test planned is a test with just two signals. The reason for this test is to see how the system would perform if one of the access points would be faulty or not even there. The impact of this would be interesting, as we mostly expect to have a lot of false positives. Since there are then are more places on the map that can have the same signal strength for those 2 SSIDs. This however might be slightly countered by the effect the walls have on the signal quality. Still it is expected to be vastly imprecise compared to all of the other systems but it will provide with an interesting comparison.

5.5.2 The set up

The general set up test with two signals is to not use the other notebook as a signal. Also the minimum requirement needed for passing would have to be lowered. For the other tests this number was set on at least three signals. Outside of these points the tests will be as close to the test set-up as the earlier described with all importance correction turned on and nearest neighbour correction.

5.6 Test areas

To properly test the system it was necessary to look at specific points. The first test is to check at one point and see how often in a period of time the system finds this point or loses it. The second test would be testing the system while moving on a given path. This is done to see how well the system can keep up while walking around. Furthermore it would be interesting to check false positives.

5.6.1 Testing at one point

To test this in a proper manner the simulation will run in a given spot for an extended amount of time. The system would track all of the possible configurations (Importance, Neighbour, No Importance and Neighbour, etc) while running. This is to create a proper comparison with as little influence from different times as possible.

5.6.2 Path testing

As in the earlier tests, all of the configurations are logged at the same time. Different this time is that the notebook is carried from point to point. During this test it will be interesting to see how well the system keeps up. Furthermore it is a test for the system's ability to detect devices that are not standing still and how fast it adapts to new locations. Shown in figure 27 is the planned path to move along for the test and to see the results afterwards.

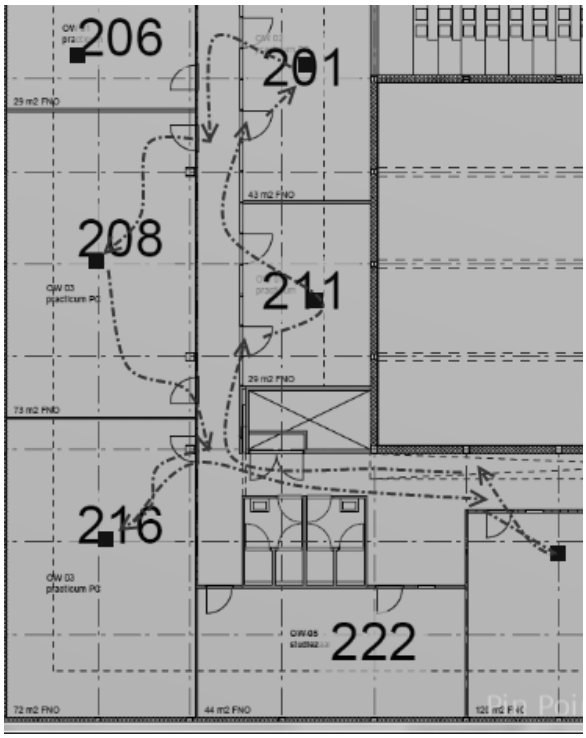


Figure 27:The path planned for the path test, starting in stopping in room 201

5.6.3 Checking for False positives

This is done afterward by using the gathered information and from this we can extract the values that would indicate a false positive. Since everything is stored, for testing purposes in CSV format, in the simulation it should not proof too difficult to extract. After this information is extract a comparison can be made too see how well the different configurations performed.

6 Results

After running the different tests the following results have been found. Split up in different sections each showing another aspect of the raw results.

6.1 Totals

In the following Table are the results of the different score totals found for each point. This table serves the purpose of showing the actual differences between points. Written in red is the actual Pin Point of the stress test. During the discussion however this table will be plotted as a graph for each of the different corrections. This is done to make the discussion slightly more visual rather, though the numbers themselves remain the same.

Keep in mind that for the Nearest Neighbour corrected numbers the score could be twice as high as a maximum and thus the numbers can be higher.

	Room 207	afstudeerruimte	Room 228	Room 216	Room 208	Cafeteria
No Correction	**34271,00**	165,44	804,33	48,89	111,78	103,00
Importance Correction	**36999,38**	133,44	1001,20	88,74	92,00	96,75
Nearest neighbour Correction	**68179,30**	276,10	1247,44	52,10	166,36	103,00
Both Corrections	**73865,12**	225,39	1455,70	94,57	137,07	96,75

Table 1: Totals for each point during the stress test over 1 hour and 43 minutes. With three Access Points. Marked in red are the values of the actual point

6.2 False positives vs. Actual over time

In this test all of the results are plotted over time, showing the different peaks and values of each of the points at a given time. The first graph shows the results of the test that has no corrections over the values. The graph below shows a time slice of the full data, this is for clarity.

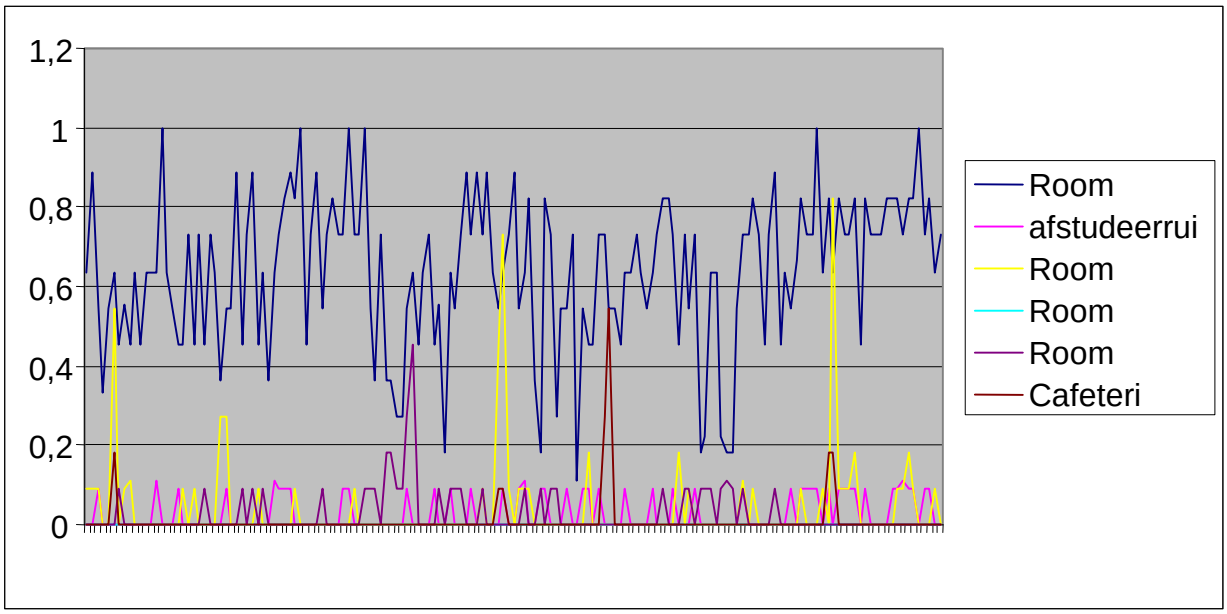


Figure 28:Showing the locations indicated by their score over a time of 30 minutes. With each location having it's own colour. The higher the score the higher the line will be on the graph.

In this graph each coloured line represents the values found for a Pin Point at a given amount of time. Since this is the stress test these values where taken over a longer period of time.

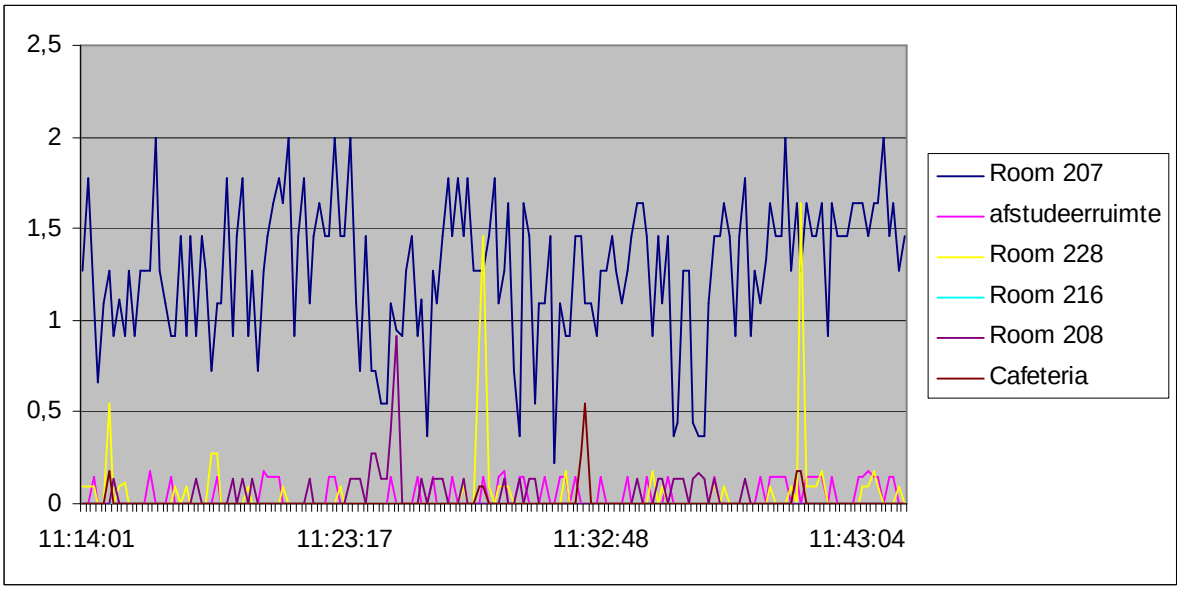


Figure 29: Showing the locations indicated by their score over time. With each location having it's own colour. The higher the score the higher the line will be on the graph.

Shown above in figure 29, are the same values in the time slice that where plotted earlier, but this time with the Nearest Neighbour correction applied to the values. Again the different colours indicate the different rooms and their scores.

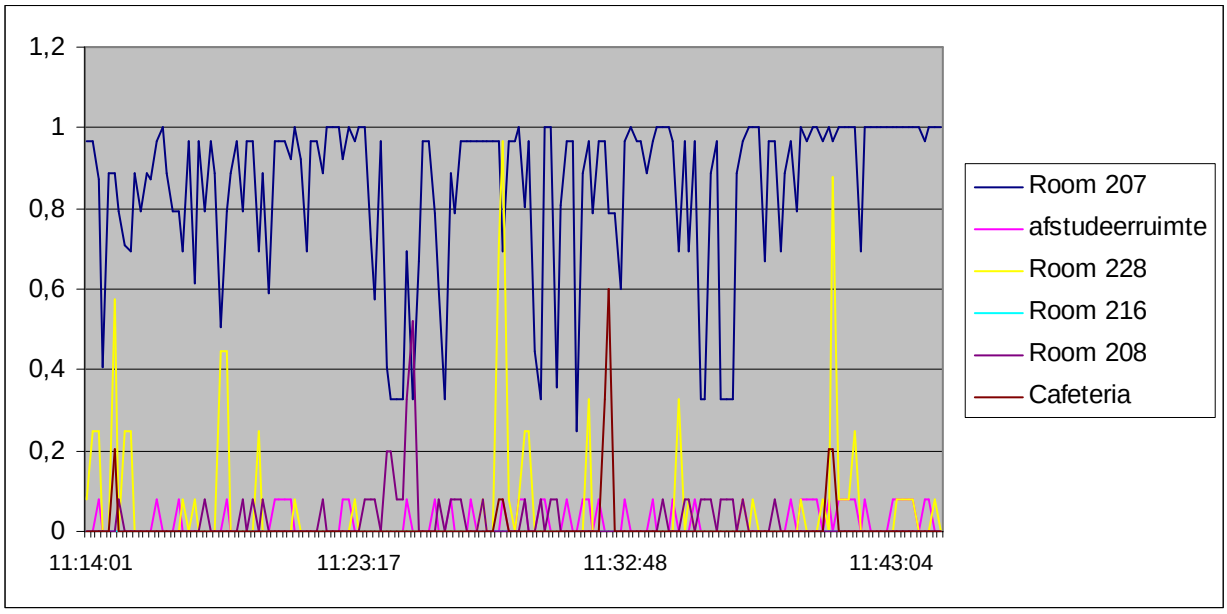


Figure 30: Showing the locations indicated by their score over time. With each location having it's own colour. The higher the score the higher the line will be on the graph.

The numbers corrected by a Feature Weights correction are then shown in Figure 30, the last test with all three access-points is shown in figure 31. In that test both the Nearest Neighbour correction and the Feature Weights correction have been applied. Again both of these plots are a time slice and the colours indicate the scores for the different rooms.

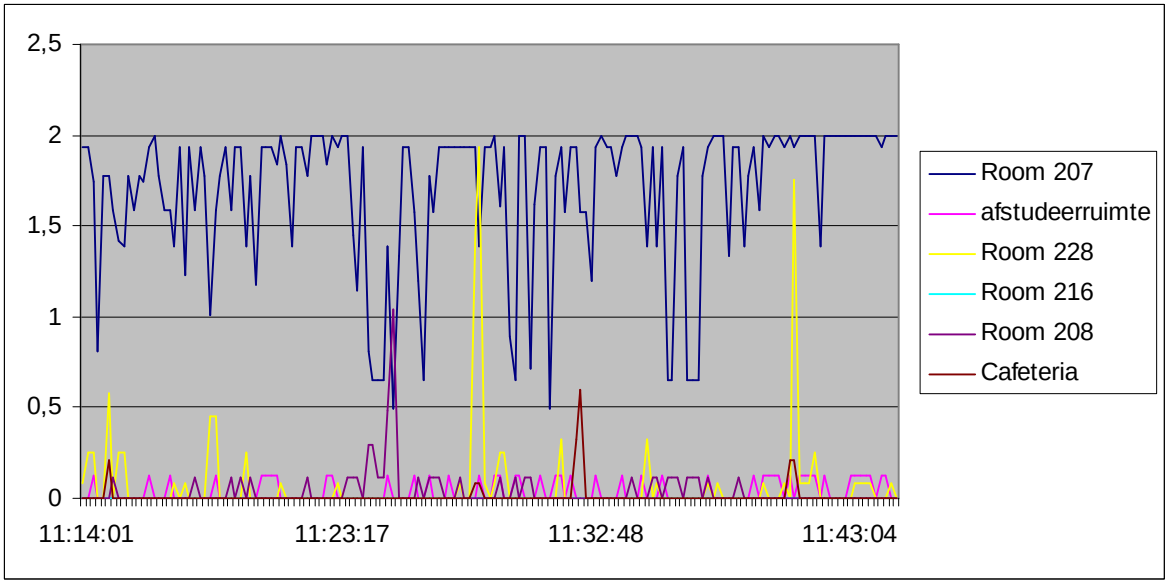


Figure 31: Showing the locations indicated by their score over time. With each location having it's own colour. The higher the score the higher the line will be on the graph.

In Figure 32 the results of the test with only two access-points are plotted. Since these results are a lot more erratic the time slice was cut short to about 15 minutes. This was done to make the graph easier to read as before it was not a clear picture. Again each room has their own colour.

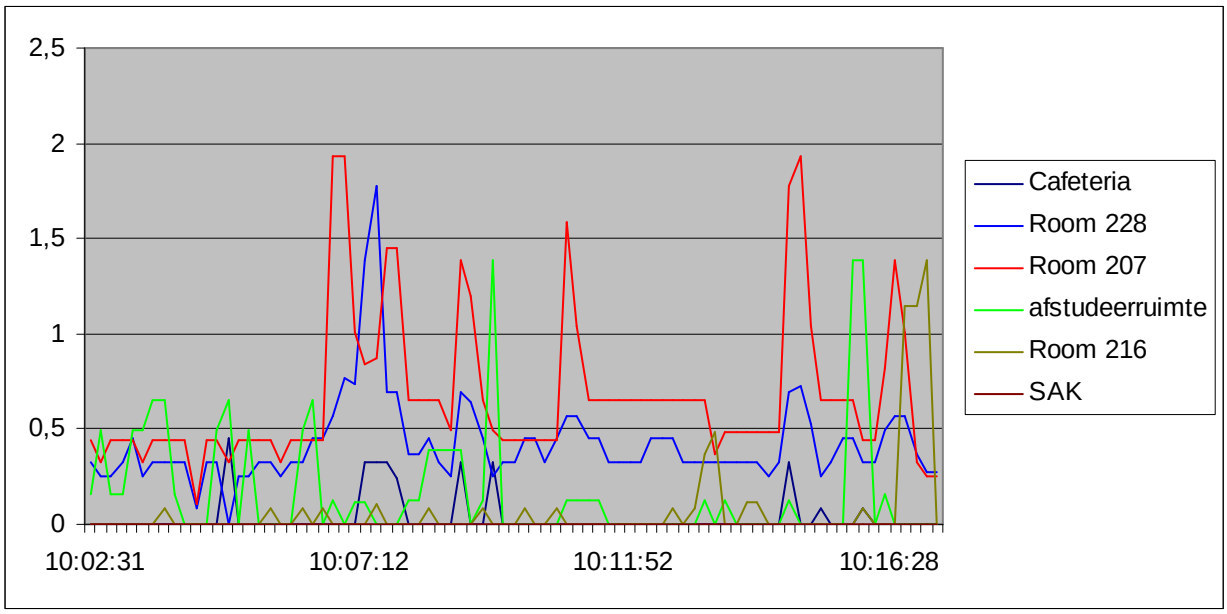


Figure 32:The results for each point during the stress test for two broadcasting access points instead of three plotted over time.

6.3 Positives occurrences over time comparison

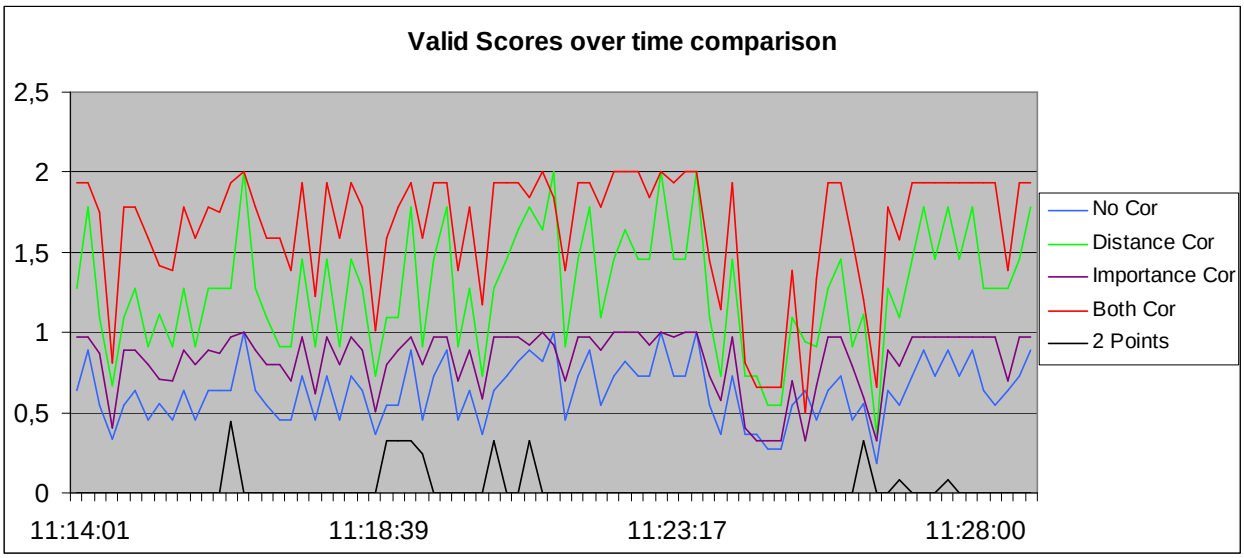


Figure 33: Valid scores over time for the actual point, for each of the different tests.

Another time sliced graph is shown in figure 33, in this graph we took 15 minutes from each of the tests and plotted them in the same graph. All of the lines are taken at the same time, with the exception of the test with 2 points. Who where taken at different times but over the same course of a time span. It functions more as a comparison of the degradation caused by using only 2 of the minimum of three access points.

6.4 Peaks and Dips

Since the above tests had to be plotted as a time slice for clarity, it was necessary to create a few tables that showed the total number of dips and peaks per test. A peak is a raise of the score while a dip is when the score is lowered. The reasons for these numbers are to show the stability and the average drop or raise of a score in the complete stress test. These are shown in tables 2 and 3.

Total amount of peaks

	Room 207	afst.ruimte	Room 228	Room 216	Room 208	Cafeteria	SAK
No Correction	**425**	179	229	12	163	24	0
Importance Correction	**304**	167	225	11	148	24	0
Distance Correction	**422**	179	229	12	163	24	0
Both	**304**	167	226	12	148	24	0
2 Points	**212**	218	255	162	0	89	18

Table 2: Amount of peaks and dips per test per Pin Point.

Table 2 is showing the total amount of peaks per point and per test, this can be an indication of how erratic a signal is behaving over the total test time of one hour and forty-five minutes. The 2 points test however had a shorter duration of one hour and fifteen minutes. Thus explaining the lower numbers for the actual room.

Average Difference between peaks							
	Room 207	afstudeerru	Room 228	Room 216	Room 208	Cafeteria	SAK
No Correction	**0,210067**	0,090037	0,158573	0,151791	0,095343	0,189739	0
Importance Correction	**0,243277**	0,084351	0,180481	0,236595	0,090516	0,193377	0
Distance Correction	**0,421664**	0,143865	0,210259	0,161761	0,14392	0,189739	0
Both	**0,489032**	0,135413	0,245904	0,238184	0,137544	0,193377	0
2 Points	**0,449806**	0,308543	0,220664	0,242891	0	0,322684	0,274223

Table 3: The average difference between peaks for each room and test.

In Table 3 the average difference between peaks and dips is shown. This the difference between each dip or raise summed and them divided by the amount of peaks and dips shown in the earlier table.

6.5 Map tests with different Thresholds

In this test the notebook was taken over a set path while recording the results. Each of these results where then plotted over the actual map. Starting out in a green colour ending in a red colour and using the alpha channel to indicate the strength of a found point. When a gap in the measurements was found of at least two seconds the system would place lower from the previous ones. It is then easier to see when a room was revisited or if there is a gap in the measurement.

As the pictures where too big to fit in this section they where added as Appendixes. Appendix F is the results from the non-corrected results. In these results the Features all have the same weight and there is no distance correction. From left to right there is a threshold of 0, 0.25 and 0.5 as minimum values.

In Appendix G the results of the nearest neighbour corrected tests are shown, again using the same colour scheme and thresholds over the different pictures. Appendix H shows the Feature weight corrected numbers and I the numbers of the whole system with all of the corrections running.

Appendix J however shows the results of the tests that only used two of the three access points. This is a test to see how well the system would respond when the minimum requirements where not met.

7 Discussion

7.1 Test results with no correction

As described in the experiment paragraph, the first results discussed are the not corrected results. This means that all Features are of equal importance and that there is no Nearest Neighbour correction. This will also allow for a baseline result that can be compared to later results. In each of the results several tests have to be discussed.

The first is a stress test; in this test the machine is at a set location for a longer amount of time as the system is running. The results from this test are then recorded; afterwards it is possible to extract information from these results. In this stress the results regarding the totals, values over time and the amount of false positives are discussed. Afterwards the discussion also reviews the impact of a threshold on the given results.

The second test is a path test, which again will talk about returning topics for each test. These topics focus more on the following of the path, the signal loss, false positives and the impact of a threshold in this test. Besides these topics, there usually also is a section about further findings and the discussion here of.

7.1.1 Stress test

Totals

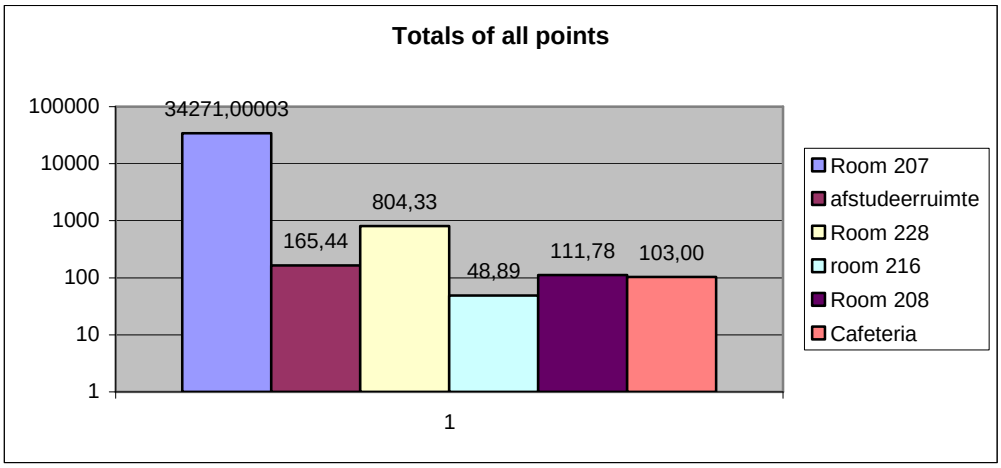


Figure 34: Totals for each point during the stress test over 1 hour and 43 minutes for the uncorrected numbers.

Shown in figure 34 is a graph representing the total sums of values for each of the Pin Points involved. These values are significant, because it tells us more then just how often a point was visited. It also shows the score a point gets.

As it is known that the system was measuring for more then 40000 (43,260 to be exact) points. With this in mind some interesting results can be seen from this graph. Each bar represents the score for a room compared to the total possible 43,260. It is very apparent that room 207 is performing much better then the others. This is a positive thing, as this location is where the test was performed.

By looking at figure 34 certain assumptions can be made regarding false positives. When compared to other tests, this test can be used as a baseline to see just how much this figure has improved or degraded. Of interest is the difference between the false positives and the actual point and then compare this difference to the differences of other tests.

This baseline can then be used to compare with the other tests. The amount of false positives however is quite low and the actual location score quite high. Figure 34 shows a total number for room 207 of 34,721. The maximum score possible would be 43,260; this is the amount of measurements times the maximum score. Using these numbers we can see that the actual point scored a 79.22 percent of the maximum score possible. This is above the expectations for uncorrected results. Since the system does not need the maximum score to receive a positive identification, one can make the assumption that even the uncorrected test is capable of tracking the room location of a device.

Over time

The other important graph was a test over time. This test basically plots the results over the time measured per point. In this graph each of the Pin Point receives its own colour. The results of this graph for this particular test are shown in figure 35. This graph shows that the biggest contributor to false positives is room 228. Sometimes this room even receives a maximum score. On overall it seems the actual location mostly outdoes all other locations.

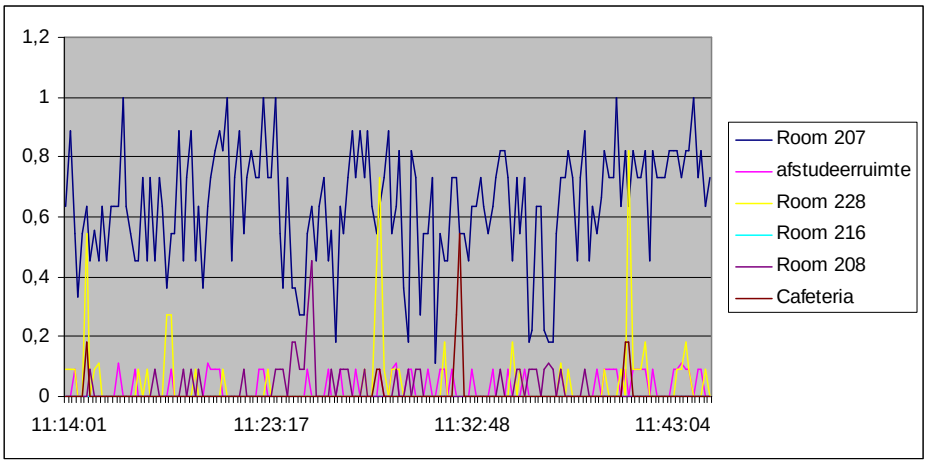


Figure 35: A time slice plot of the stress test for the uncorrected score numbers per Pin Point.

This maximum score is achieved when all Features give a positive identification. In most cases the actual location still scores higher. However it does show that it is important to give weights to the different Features. When weighted Features prone to false positives or error will not be able to affect the results as much as in this test. Room 207 however still outdoes all other points over time; this again is a positive note in the uncorrected test, as it bodes well for the corrected results.

The graph does however show a very erratic line in general. The location value often dips down or even remains at a low level. This sometimes causes the other positions to score higher or very close to the actual position. There are even hints of complete gaps when looking at the graph, for instance at about 11:40 the graph shows a significant drop in signal for a longer period of time. Which is dangerously close to becoming a gap in the signal.

False positives in general

In general the number of false positives seems to stay quite low. Even in this test, the number of times a different point would score higher in the stress test then the actual point is rare.

Although these scores do exist their appearances tend to be sporadic. It will be interesting to watch for improvement in the discussion of the other results. The differences between the values in the total graph also are points of interest. Since between the different tests this can be a clear identification of improvement.

Both Figure 34 and 35 basically already show very desirable results in regards for false positives. This however is still to be tested during the path test in which the device will be carried around a pre set path through the test area. It is expected that the switching between points will cause for an increase of false positives as the system doubts between multiple points. It is however expected that eventually the system will settle for the actual point.

Impact of the threshold

Another possibility is the introduction of a minimum threshold. This value will basically be a border between accepted values that are higher then the minimum and values that are lower. When using this threshold at values of 0.5 and 0.25 it can be clearly seen in the over time graph that gaps would start to appear. It does however filter out some of the other Pin Points that are now basically behaving as background noise.

In case of a 0.25 we see that only room 228 and the afstudeerruimte still apply for a positive identification if the room 207 point would dip in value. This will cause a reduction in false positives. It will also cause for an increase in gaps in the actual point. These gaps are caused because of the erratic behaviour of the main point in this uncorrected test. It will however make sure that most other points are not able to score a positive identification during a dip. This is better then a slight signal gap, as one can show the last known position for a device.

However once the threshold increases to a minimum of 0.5, it can be seen that almost no point ever crosses it outside of the actual point. Only one signal occasionally passes this threshold and another only seems to barely make the value just towards the end. However such a threshold would cause for some big gaps while measuring. Especially the time length of these gaps makes it harder to keep the last found point as the device location. It is expected that these problems are solved to a certain extend with the implementation of the corrections.

7.1.2 Path test

Like the stress test, the path test results where gathered together with the results of the other tests. This is done to create a proper comparison in numbers but also to eliminate factors that might influence the numbers. Figure 36 shows the path test by colouration, the starting colour was green and the end colour was red. Transparency was used to indicate how high the score was on a found position. If a position is more transparent then a neighbouring position, it means that it failed more Features then the neighbour. Gaps bigger then two seconds are represented in the graph by placing the next point on a lower vertical position. This will not just show the gaps but also show when a point is being revisited.

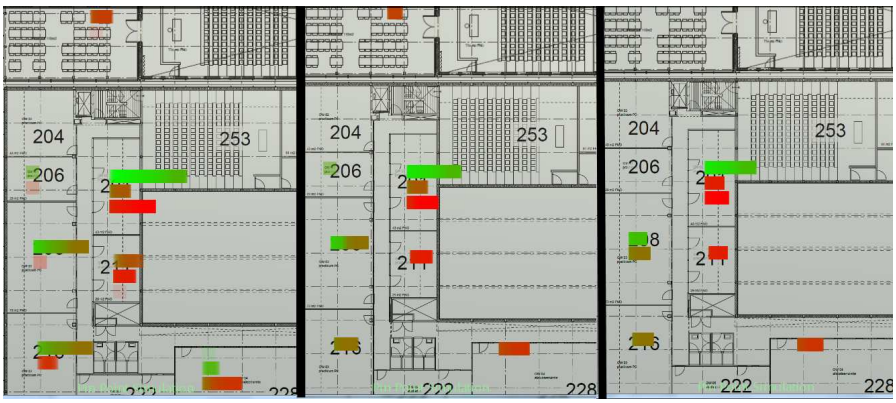


Figure 36: A side-by-side comparison of the measured Feature scores. The least transparent means a higher score for a point, red is the end point green is the beginning. The colouration between the two indicates the transition. The left picture shows the result with no threshold, the middle is with a threshold of 0.25 or higher scores and the right are for 0.5 values or higher.

The path itself is described in figure 29. The time it took was about two and a half minute, from 10:44:30 till 10:47:00 we walked the route described in the before mentioned figure. The figure also shows the map three times, each for a different minimum threshold. The left most allows all values to be shown. The middle and right show the impacts of a threshold of 0.25 and 0.5 on the found results and how they affect the path detection.

Following over time

Followed over time it is possible to see that the system does generate some false positives. Most of the time they are weaker then the actual positive values unless switching between rooms. However sometimes these values can be strong enough to be an actual competitor. The effects of these can be seen in the competition between room 208 and 216. At one point they seem to be almost equal in strength with the same colour. Even a threshold does not solve this situation. There also seem to be ghost measurements; these are very faint signals like those in room 206.

Another thing is the doubting between rooms when someone is walking in a hallway, the brown bar in room 207 and room 211 shows this. This is not a big problem as at those times one is not in either room and a slight threshold would already solve this. A bigger problem however is the ghost signal the system detects in places that are not even close to the actual point, luckily even these seem to disappear with the introduction of a slight threshold.

The path test itself was a great success even without the added correction one could already make out from where to where the device was taken in the two and a half minutes it was carried around. This bodes very well for the other tests, which added weights to the Features

and a distance correction. There are however some improvements one can watch for especially the doubting between room 208 and room 216.

Loss of Pin Point

One of the things happening during the test was a slight Loss of Pin Point during the test. This caused for some holes in the measuring, which we can see in the figure 36 by the slight colour difference between the starting point room 207 and the next point 208. This loss was not detected until the reviewing the results from the tests that added a weight correction. In these tests it finds the proper point sooner. A slight green bar appears either connected and with a threshold on the next vertical line. This means that during these test the signal was really low and received very poor scores of which the actual point was still the highest.

Since the signal loss was quite low and only would last a second at most, it did not break the test and it was still possible to track the device with little or no problem.

False positives

As earlier mentioned the results did contain some false positives. They are often weak but since they sometimes appear with values that are quite close to the actual point it is safe to say that they will exist when tracking devices for a longer period of time. It is expected that the tests with corrected numbers will lower the number of false positives. When they do exist it is expected that they do not score as high as in this test.

Another point is that this test sometimes seems to doubt a bit between rooms, as is shown in figure 36 by the doubts it has if the system is located in room 208 or room 216. The brown bar is the same colour and transparency in both meaning they are close enough to be mistaken for each other even though we are clearly in room 216. Even with a higher threshold this bar still exists. It is expected that this is caused because of worse performing Features influencing the reading and that especially the weight correction with lower improve this. This particular problem will be a comparison point in the discussion of the other results.

General findings Threshold impacts

The threshold themselves seems to be a wise choice to add. It greatly reduces ghost findings (very weak point detections) and false positives. In fact just by adding a threshold of 0.5 to the uncorrected results one can already achieve very good results. Which indicates that even without any corrections outside of a threshold the system is more then capable to detect the room where a smart device would be located.

7.1.3 Further findings based on the results

One of the things noted during the stress test is that the values found are quite erratic, it is hoped that the addition of a weight correction will counter this. With this correction the better performing Features are able to contribute more to the detection score then the others. This erratic behaviour can be the cause of problems during the path tests. As moving around will already cause dips and raises in the measured signal strength having an erratic Feature measurement will only add to this. Erratic signal strengths can cause improper detection of points even for a longer period of time.

The total amount of false positives seems to be quite low even in this test. The expectations are that this will still improve since the corrections will add certain stability and earlier switching then the uncorrected tests. However how big of an improvement they actually are is still to be seen.

7.2 Feature Weights Correction

The next set or results are those that are corrected with the earlier discussed weights. These weights basically make one Feature more important then others based on their earlier results. Better performing Features receive a higher weight then those of poor performance. It is expected that this will be a cause of improvement in all of the tests.

7.2.1 Stress test

As in the earlier test, the first results discussed are those measured during the stress test. This time we mostly looked for improvements compared to the un corrected tests. As before the first graph shows the totals. As there is no distance multiplier the highest amount available is the same as before and can just be compared one on one.

After this there is a graph showing the values over time in relation to all the points. Again this is compared to the earlier found values and if the results are still as erratic as before. We would be also be looking for dips and if they are as low as before or if there is an improvement.

Totals

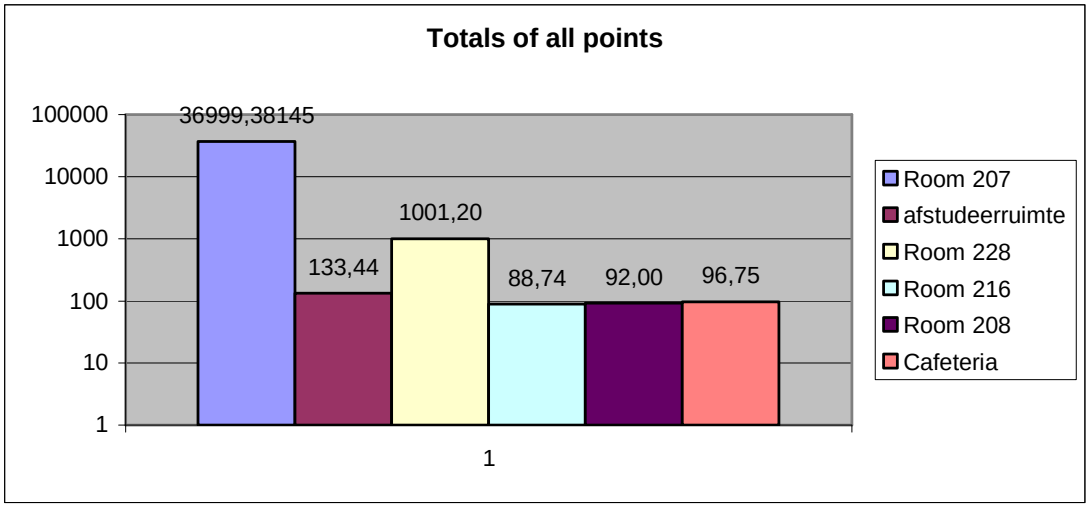


Figure 37: Totals for each point during the stress test over 1 hour and 43 minutes for the Importance Corrected numbers.

Shown in figure 37 are the totals of all the Pin Points over time. As before the maximum score is the same at 43:260. Since all the measurements where taken at the same time, it is safe to say that the difference is caused by the weight correction rather then external influences.

One of the first things noticed is that the actual Pin Point is in total is about 2500 points higher then in the graph in figure 34. Another point of interest is that the biggest false positive contributor in the earlier test has also increased. However the difference between the two has also increased, thus showing that even in that aspect it was performing better al be it not by as

high a numbers as was initially hoped for. All the other points have gone down in value thus indicating improvements in general.

Considering that it has the same maximum score as the not corrected test, we can see an improvement in the scoring percentage. The percentage of the score is about 86 percent ((36999.38/43260)*100 = 85.52) of the max possible score compared to the 79 percent of the previous uncorrected test ((34271/43260) *100 = 79.22). It will however be interesting to see if this number can get even higher in the combined distance and error corrected test, as the scores are already quite high.

Over time

The next graph shows the results over time. Like earlier this graph shows the values of each Pin Point over the time of the stress test. This is graph is shown in figure 38 and the first thing noticeable is that the signal behaves far less erratic then in the earlier results. The dips also seem to be higher in score with the exception of two dips that are at the same level.

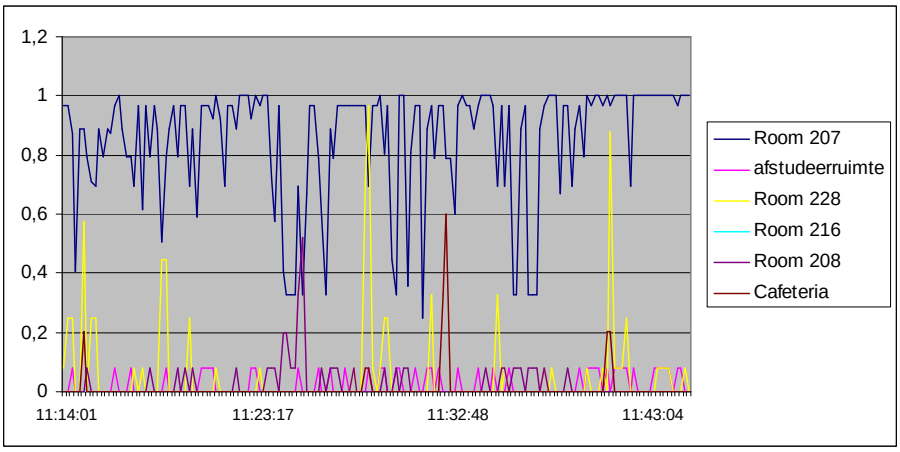


Figure 38: A time slice plot of the stress test for the Feature Weights Corrected scores per Pin Point.

The stability really is a good addition at almost all times the measured values for room 207 seems to be between 0.7 and 1.0. The peaks of room 228 also seem to have diminished compared to the uncorrected test. The peaks that exist seem to lower then the score of the actual room with the possible exception of one. The very low scores found before at about 11:40 seem to have improved only slightly. The hope is that the distance correction will solve this and push it closer to a 0.8 rather then a 0.3 it has now.

False positives

False positives in general seemed to have died down when one takes the importance factor in account. Only a few times did other points get close to actually generating a false positive. This in itself is not a problem as the main signal still usually is stronger then the others. It is expected that during the test of the complete system, the distance correction will further minimize the number of false positives.

It is however promising that the difference between the actual point and the false positives have increased and the mount of false positives during the over time stress tests have been reduced. This means that there is an increase in the amount of times a device will be placed at its proper location compared to the uncorrected tests.

Impact of the threshold

Adding a threshold of in this test actually would help a lot, since the signal itself is far more stable then the earlier one a threshold of 0.5 minimum would only add a few slight periods of signal loss. It will however filter out most of the false positive numbers outside of the few spikes caused by room 228.

This however we think will be fixed in great regards with the tests using the distance correction in the complete system results. In this case the last found point has a higher multiplier and thus points that are further away will be decreased in relation to the last found point.

7.2.2 Path test

The next test done with these weighted measurements was the path test. Again the discussion reviews different aspects from following the path over time till the impacts of a threshold.

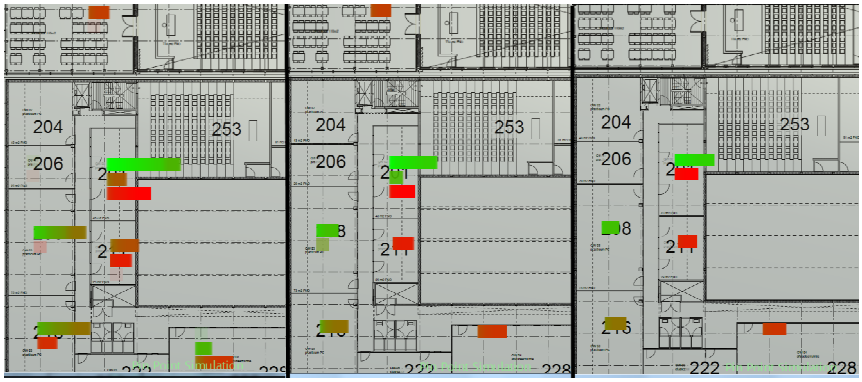


Figure 39: A side-by-side comparison of the measured Feature scores. The least transparent means a higher score for a point, red is the end point green is the beginning. The colouration between the two indicates the transition. The left picture shows the result with no threshold, the middle is with a threshold of 0.25 or higher scores and the right are for 0.5 values or higher.

One of the first things noticeable in figure 39 is that the values themselves seem to be higher, with less "ghosting". Ghosting is the very faint false positives that seem to be at multiple places on the map. Ghosts usually are not a problem as there is usually a bigger value at the same time in a different place. They would also disappear with the use of a slight threshold.

Following over time

On thing that is possible in this is the following over time even with no threshold present. When looking to the colour tints in figure 39 in each of the room, it gradually progresses from green to red. This is a clear indication of a path, even in the two rooms that formed a problem in the uncorrected measurement there now seems to be a difference that becomes clearer with an increase of threshold. Room 216 contains more red and brown then room 208 thus indicating that we probably visited that one on a later time.

When adding a threshold, the "Ghosts" disappear. Outside of a slight one in room 208, whose value appears to be similar to the first value of room 216 after a slight signal gap. This still indicated that it probably was caused when moving from 208 to 216. This indicates a clear path from Pin Point to Pin Point even with the lower of the two thresholds.

With a threshold of 0.5 as a minimum all of the false positives disappear and the path becomes very clear. This is an improvement again over the earlier lower value that still showed a false positive in the cafeteria area of the map. Even though this value probably was lower and in an actual working environment probably would not show up at all.

Loss of Pin Point

When looking at the results, which are not affected by a threshold, one can see almost no signal loss. This is not surprising considering the results in the uncorrected test. However it also seems that the few near signal losses that where showing in the uncorrected numbers have dissipated slightly. The transparency seems to almost completely disappear at any given point when in a room. This shows that adding weights to better performing Features is actually very beneficial to the path test.

Even when adding a threshold there seems to be little loss, of course the system has gaps when walking through the hallways but technically one is not in a room when that happens and thus the system should not put the device in one of the rooms.

False positives

The amount of false positives seems to have diminished greatly. Even without a threshold the system seems to be able to place the device in the correct room most of the time. When before there was a problem finding the device between room 216 and 208 there now, seems to be an indication that the device was in 208 before visiting 216. The moment a threshold, even a slight one, is introduced the false positives seems to disappear and the strongest one located in the Cafeteria also quickly vanishes when the threshold increases.

These findings bode very well for the combined test, where the distance is also taken into account.

General findings Threshold impacts

It seems that yet again a threshold is actually helping in a major way. When there is a desire to remove lower error values the threshold serves that purpose. In fact it seems to remove a lot of doubt of where a device is. It also does not indicate that the device is in a room when the device is actually in a hallway. These are all important and beneficial results to a rather simple to implement, and already implemented, technique.

7.2.3 Further findings based on the results

These results tell us that the added bonus of weighted Features is one that is particularly showing of during the path tests. However it also produced improved results during the stress test when compared to the earlier findings in the non-corrected tests. During the path test the system has a far lower number of ghosts even without a threshold. It also seems to have less doubt to find a room compared to its neighbours. This especially was a problem with rooms 208 and 216 in the uncorrected test.

7.3 Nearest Neighbour Correction

The next set of results is that which only use a distance correction but leaves the Features equal in importance. This is done mostly to see the impact this has on the measurements and to see if this is a positive addition. Of course one has to take in account that the highest value

now is two instead of one so some tests need a few slight adjustments to be comparable to the uncorrected test.

If an adjustment was made this adjustment will be stated and explained. There will also be an explanation for the reason for the adjustment and how it now is comparable to the earlier measurements. Of course this also goes for the last test, which includes both the weights and, the distance correction.

7.3.1 Stress test

Like before the first focus is on the stress test, of course this time we need to keep in mind that the maximum score has significantly increased. This noted we can still check for the distances between Pin Points found and see if a point would be close to taking over and forming a threat as a false positive. Also it is expected to be a more stable signal, as signals that are further away automatically score lower then the last found signal. This will make the chance of finding a false positive far lower then before as the score for an actual found point is often higher by quite a margin.

Totals

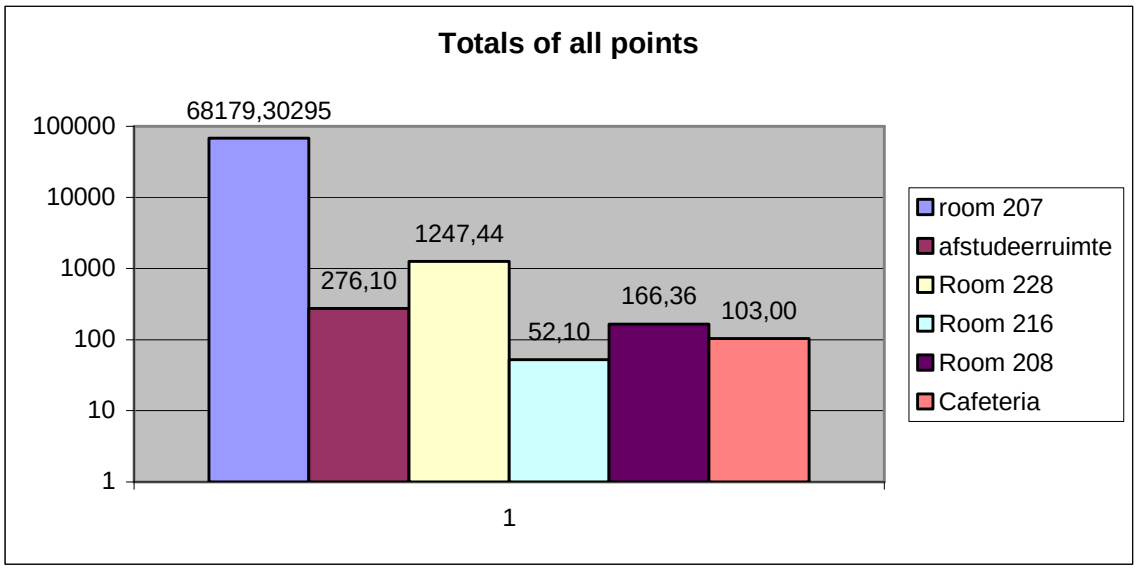


Figure 40: Totals for each point during the stress test over 1 hour and 43 minutes for the Distance Corrected numbers.

As before the first thing discussed are the totals, which are shown in the totals graph in figure 40. Basically it remains the same but to create a comparison percentage, as done with the earlier comparisons we multiplied the total amount of points one can obtain by two. This is since the maximum score also has increased.

If one looks to the percentage alone the score of 68,179.3 only takes up 78.8 percent. This is lower then the uncorrected tests. This is however only by a percentage of 0.4 percent. When keeping in mind that the tests still starts with a maximum at one and only uses the multiplier when a point has been found and when it switches points it also loses some of that score, it is only natural that the value is lower considering the set-up of the test.

The difference between points however is larger by far. Even though the other points also had a very slight increase, the increase of those points is only marginal to slight in comparison to the tremendous increase of the actual point. This difference means that there is a tremendous decrease in false positive scores during the stress test and it would probably stay at the given point rather then to switch to a false one. The decrease is also caused because we just doubled the maximum score possible, since this is of course the maximum score compared to the normal non-distance corrected tests.

This should prove to be a stabilizing factor in filtering out points that are not even close to last found Pin Point. The system can still receive this point as a false positive but it would have to find it first before getting a score increase. This should greatly reduce the chance of switching between points that are not even close.

Over time

Again these results where also plotted over time. This basically gave an improved version of the non-corrected graph shown in figure 35. Room 228 still caused a few false positives but the other rooms simply do not get close enough to take over the highest position as shown in figure 41.

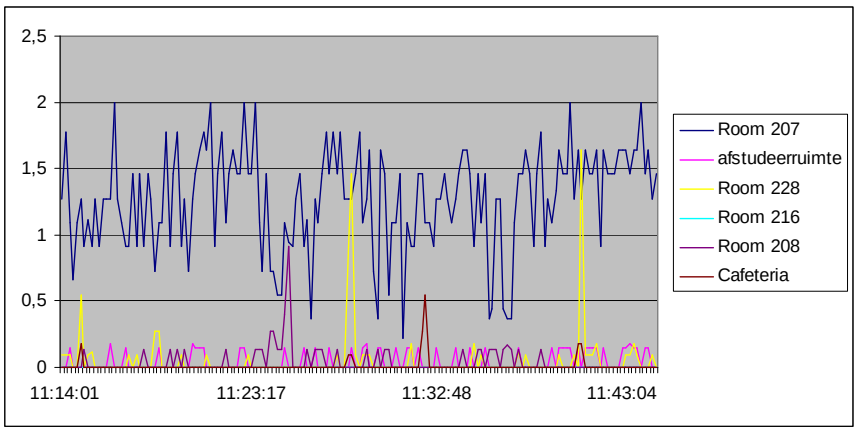


Figure 41: A time slice plot of the stress test for the Distance Corrected scores over time per Pin Point.

One the first thing noticed is the far higher score and remaining score of the actual Pin Point in room 207. While room 228 still causes about 2 major false positives it really is the only one which such high scores. Often after such a positive the system often quickly finds the proper pinpoint again and room 228 is marked down to a level on which it is of no threat.

When only looking to the other rooms it is possible to spot, as earlier mentioned, that they usually do not receive a high enough score to pose a threat of a false positive, in fact they more look like background noise then actual threats to the systems precision.

However like before with the uncorrected test, the results seem very chaotic. This is to be expected, as it is basically an increased scale of the uncorrected test with the possibility of gaining one extra point. It does however add some stability in the that other values do not pose as high a threat as they did before.

The gap found at about 11:40 still is present al be it closer to 0.5 then to 0.2 like it was in the uncorrected tests. The hope is that the weight correction together with the distance correction will push this upward that the signal remains above 0.5 all over. This would mean it would be perfectly safe to introduce a threshold of 0.5.

False positives

As expected the threat of false positives has reduced over the uncorrected tests, however room 228 remains able to create some false positives. It does seem that the Afstudeerruimte is able to approach the signal in the end, but one has to keep in mind that this room is pretty close to room 207 and thus also gets a multiplier bonus. Even then it still does not go over the value of 1, which means that it probably never is selected as a found positions.

In general it seems that the actual location almost always is higher by quite a margin and it is expected that adding weights to this system will improve the system even more. Especially regarding the earlier mentioned stability of the numbers.

Impact of the threshold

Like in the other tests we think a threshold would help a lot in really keeping out the unwanted values. Although in this case just selecting the highest one would already give good results seeing the huge difference between the signals. However adding a slight threshold of say 0.5 would already cause the system to select the proper location from the start rather then to remain at one erratic position. However since this correction basically multiplies the value in respect to the last found value, a threshold might actually make it harder to find the proper Pin Point again after it was lost to a false positive.

7.3.2 Path test

Like in the previous test we also recorded the performances of this correction during the path test. The path tests using different thresholds are shown in figure 42, which again uses a 0, 0.25 and 0.5 as threshold values.

Figure 42: A side-by-side comparison of the measured Feature scores. The least transparent means a higher score for a point, red is the end point green is the beginning. The colouration between the two indicates the transition. The left picture shows the result with no threshold, the middle is with a threshold of 0.25 or higher scores and the right are for 0.5 values or higher.

On of the first things one can notice here is that all of the false positives are far more transparent then before during the uncorrected test. This actually means that their values are lower in respect to the maximum value possible. However since the base values are still with an uncorrected weight there still appear to be some ghosts like the two in room 206.

Following over time

When tracking the device over time one gets a general idea of the path of the device that was tracked. There is however still an uncertainty between room 208 and 216, as before with the uncorrected test. This is however to be expected, since besides a distance correction it basically still uses equal weights for each Features. Though a distance calculation does add

some stability, neighbouring rooms can still receive high scores. Scores high enough to be seen as a false positive for being higher then the actual Pin Point.

Since this was mostly fixed with the weighted Features, it is not expected that it will be a problem with both of the corrections enabled. Outside of that it is very well possible, especially when adding a slight threshold to follow the device from room to room.

Loss of Pin Point

The Loss of Pin Point or Pin Point location seems to be about the same as the uncorrected test. This makes sense since all of the Features, even the ones performing significantly better, have the same weight. As adding a distance correction mostly makes the point itself slightly more stable and makes switching to neighbouring points easier then to points that are not in the vicinity. Outside of that it is to be expected, even adding a threshold seems to have the same effects as the uncorrected test.

False positives

These remain mostly the same in case of neighbouring rooms, like the room 208 vs. 216 problems. The scores for the Pin Points further away are significantly lower in comparison to the uncorrected results. In fact even the one in the cafeteria now shows as partially transparent where before it was almost solid. As it mostly stays the same for neighbouring rooms it does however show the need for weighted Features to be enabled and it is expected to see an improvement in the combined numbers.

7.3.3 Further findings based on the results

It seems that with the distance multiplier the ability to add a threshold is just a bit less of an impact, in fact one can even discuss of adding an even high threshold at say 0.6 or even 0.75. Since the threshold really seems to create a big difference between the falsely found points and the actual points. It also seems to guide the system when on the move from room to room as the false positives which where in completely different rooms, referred to as “ghosts”, seems to have decreased in strength in comparison to the actual Pin Points.

Even though they are marginal improvements with each step, as expected with the rather good result with the uncorrected test, they still are improvements and this again bodes well for the final results using both corrections over the values.

7.4 Complete System

The final results with regular testing are the results with the complete system. All of the corrections are then implemented. Again this was done at the same time as the other tests to make sure there are no other influences besides the corrections. As before the first results discussed is the review about the findings in the stress test before concentrating on the path test.

Again if adjustments need to be made for the interpretation of the stress test, this will be mentioned together with the reasons for the adjustments. An expected improvement compared to the other test, even thought compared to the weighted test this might be a hard feat to accomplish. Especially in the path test it might be hard to see an actual improvement. During

the stress test however this will more apparent to see, as that one focuses more on the raw numbers and results.

7.4.1 Stress test

As before the first focus is the review of the stress test results. These results where taken at the same time as the others but this time with all corrections implemented. This means that the numbers will be changing to the effects of both corrections. Especially for the stress test this has consequences, as one will see during the totals test in difference between the Pin Points. It is also noticeable during the over time test. There should now be a more prominent result for the actual Pin Point compared to the other results.

Totals

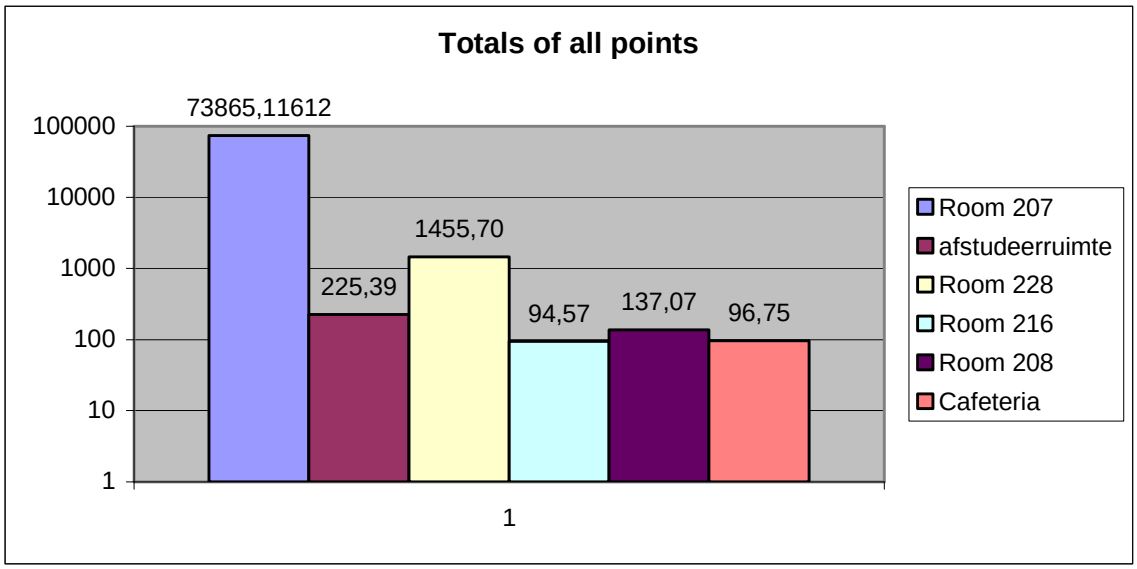


Figure 43: Totals for each point during the stress test over 1 hour and 43 minutes with both corrections implemented

As before the first review is the discussion of the totals. These results are shown in figure 43. One of the first noticeable things is the tremendous difference between the actual point in room 207 and the other points. In fact this difference is greater then every other difference in the other total graphs shown in figures 34, 36 and 40. In fact in most of the false positive numbers there was a decrease in total compared to the distance corrected test.

Although most differences have improved, the one compared to Room 228 has only improved slightly. The distance modifier can explain this, as it also increases the possible scores of points close by and Room 228 is still close enough to gain a boost. In overall the addition of the weighted Features to the distance correction is a positive one.

When looking to the maximum possible score one can see that the value of 73,865.12 is 85.4 percent of the maximum score. This is only a minor reduction compared to the 85.52 percent of the weighted only test. Especially when considering that the higher maximum score usually lowers the percentage. This is, as explained before during the distance only corrected test. This is because the maximum score is not always obtained, as the point first needs to be found

before it receives a proper multiplier. In it self this is a huge improvement compared to all of the other tests especially the uncorrected numbers.

Over time

As before the results where also plotted over time. Shown in figure 44 is the graph that plots the different values of the different Pin Points over time. The first thing noticeable is a reduction in the competition to the actual Pin Point. The only other Pin Point coming close is the ever-resilient room 228. The other points however do not even seem to be a threat. It even goes as far as almost completely eliminating them to the level of background static. It remains to be seen if these results will be reflected in the path test, but the stress test was a most definite improvement over the earlier results.

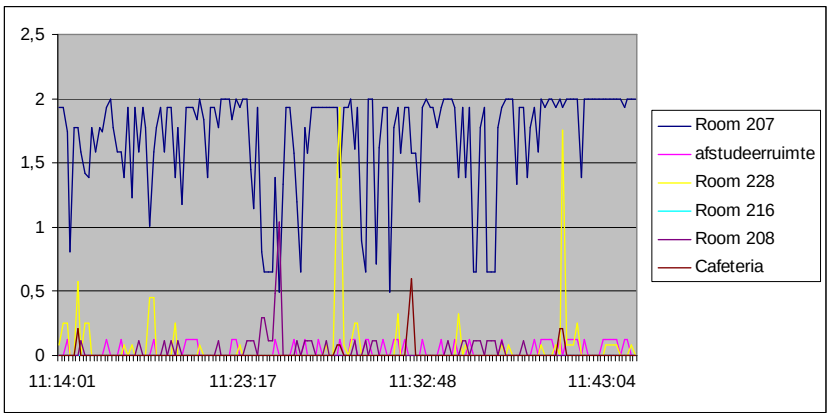


Figure 44: A time slice plot of the stress test for the scores with Both corrections implemented over time per Pin Point.

When for instance compared to the graph in figure 40, the weight corrected results over time, the Cafeteria and the afstudeerruimte actually have some peaks. In this test however these peaks are completely gone. The graph does show the same stability of the true Pin Point as the one in figure 38. The instability found in figure 41, which was recognizable from the uncorrected test results, has now improved. This stability will give false positive less of a chance then when this score is erratic. As a slightly more erratic signal is more likely to fall to a point where a false positive signal can catch up and cause it to be higher then the actual point.

The gap shown before seems to have improved again, almost to a score of 0.8 as was hoped. There seems to be no score at all under the 0.5 limit which is a great result considering 0.5 is the maximum threshold we use during our path test.

False positives

As mentioned before room 228 was again a troublemaker, as it always caused some high results even in the previous tests. It does seem however that the amount of times that it causes trouble has significantly been reduced. Only three times it threatens the main Pin Point. One of these times it does not seem to get a score as high as the main Pin Point, but all of these times the duration of threat seems shorter. The amount of other false positives compared to the earlier results has gone down so much that the system scored almost a perfect score.

Since a perfect system is not feasible, we believe that these results are more then reasonable for a system that it otherwise completely new. So considering a time span of almost 2 hour, the length of this stress test, to only have a few seconds of false positive would be very reasonable. This would mean that in a whole day the system would only place the device in the wrong room for less then a minute. This minute would not even be continuous but

fragmented over 24 hours. As a person usually does not sit in a room for 24 hours on top of the minute being fragmented, it would mean that the system would correct itself within a few seconds.

Impact of the threshold

Since the difference is quite high a threshold would basically make sure that other Pin Points are completely filtered out. A threshold of 0.5 would almost completely filter out all of the other Pin Point score. If the Pin Point scores will drop, it would cause a small loss of Pin Point identification rather then a false positive identification. This however is preferable to actually having the system identify a false Pin Point, as the last known position still would indicate the position of the device rather then a completely false position.

Even with this threshold the system seems to receive a stable score near the maximum for the actual Pin Point. This means that the amount Pin Point Loss with such a threshold would be very low. It is hoped that this will show in a more graphic way during the Path Test Results that follows.

7.4.2 Path test

Like with all other results the complete system numbers where also gathered. This again was plotted on a map with different threshold levels. It is not however as easy to see a difference as before during the stress test, However if one knows what to look for in figure 45 one can see some major improvements. As before the threshold values are shown from left to right the threshold values are 0.0, 0.25 and 0.5. And also as before the colour gradually changes tint from green to red as time passed by.

Figure 45: A side-by-side comparison of the measured Feature scores. The least transparent means a higher score for a point, red is the end point green is the beginning. The colouration between the two indicates the transition. The left picture shows the result with no threshold, the middle is with a threshold of 0.25 or higher scores and the right are for 0.5 values or higher.

Following over time

Even with no threshold it is possible to track the device over time. When looking to the end and start of the bars it is possible to see that the end of a bar often connects in colouration to the one in the next room on the route. Even though without a threshold there is some doubt between rooms like between 208 and 216 it is not as apparent as before with the uncorrected and weighted test. This can be seen in the colouration of the 216 bar, which slightly redder. It also seems that the distance correction helps as the strength of the findings in the cafeteria are as transparent if not more transparent then in the distance only corrected test.

Also there only seems to be a slight false positive ghost in room 208, which probably is caused by our walking back to room 207 via the afstudeerruimte. The other false positives are not ghosts as they are stronger but still disappear fast with the introduction of a threshold. Indeed with the 0.5 minimum threshold it seems that all of the false positives have left and one can clearly make out the route taken with the device.

Loss of Pin Point

There only seems to be a slight loss of Pin Point score, which almost is non-existent unless one adds a small threshold to the numbers. This only seems to remove the false positives and loses the Pin Point once one leaves a room. This is a good thing, as losing those false positives when walking through a hallway is better then to have the system think you are in a room you are not. Now there is an option to approximate the position that is researched in the paper of my research partner [16], in the case a point is not found. Outside of that option it is better to just point to the last known location rather then to a false positive.

False positives

On the subject of false positives one can be brief, the amount has decrease significantly compared to the uncorrected test and to the other two tests. The amount of slight ghost signals has decreased to only one point in room 208 when walking by. There is one slight false positive in room 228 right in the beginning with a green colour and the ever-persistent Cafeteria one has become less high in score. All of these seem to disappear the moment even a slightest threshold is introduced so when selecting the highest value as the point where the device is, these points would not even record as positive ID.

The same goes for the doubting points found between room 208 and room 216, even with no threshold there already is a clear indication that the device first visited room 208 and then room 216. The longer bar and the redder colouration show this. In fact if one adds even a small threshold, it already becomes clear that there is a distinct colour difference between the rooms and that there can be no doubt about what room was visited first. There is still a slight doubt between room 228 and 216 shown by the red bar below the longer bar, but this was mostly caused by the walk too room 228 and disappears when the higher threshold is used.

All of the other false positives like the very light ghost in 208 are mostly caused by walking around and disappear quickly. This makes sense as room 208 is close to the path back to room 211 and later 207. Again this score is too low to be selected as a Pin Point, and the threshold would eliminate it when walking in a hallway.

Threshold impacts

As noted before, and even during the earlier test, a threshold can be a huge bonus in removing light erroneous scores from the measurements done by the system. At a 0.5 minimum threshold al of the points which are false identifications seem to have disappeared and one can clearly see the path walked from room to room by the colouring of the points. These results really are better then expected as one can really track the location of a device.

7.4.3 Further findings based on the results

Outside of these results we identified a slight dip in the score that did not show on the other non-weighted Feature. It shows on the weighted results but only becomes apparent after one

knows what to look for. When looking to room 207 one sees that when a threshold is added there is a slight single dot before moving to room 208. This is caused by a low Pin Point score that is lower then the 0.25 thresholds. This in itself is not a big problem as moving between room 207 and 208 caused it, the system also corrects itself before the device even reached the hallway, as can be seen by the higher threshold of 0.5.

It does however indicate that both the weighted importance and the distance calculation make it easier for the system to find points when the Pin Point scores drop to low to be noticeable. Since the System finds it again before one even makes it out of the room.

7.5 Test with 2 Access points

The final test was done separate from the others, as it was not possible to do them at the same time. This test was basically to find out the degradation of the quality of service when only using two access points rather then three. Again a stress test and a path test where performed and the results where stored, however this time the discussion will only focus on the most favourable of the results instead of looking to all 4 possibilities (Uncorrected, Weight Corrected, Distance corrected and both corrections implemented). In this case the test that performed the best, was that one that had both corrections implemented, these still gave some results which of Pin Point localisation.

7.5.1 Stress test

The first results up for discussion are the results from the Stress test. This test ran over a time span of about one hour and twenty minutes. This time again all of the results where recorded and put in a totals and a graph that shows the selection of points over time. After this there also is a graph that shows the scores over time for the different Pin Points in the System.

Again this time the test was done in room 207 using the same Touchsmart notebook but without the signal broadcasted by the Medion notebook. Only a few slight changes where made to the system as it now also accepts minimum scores where it passes for only two of the three access points.

Totals

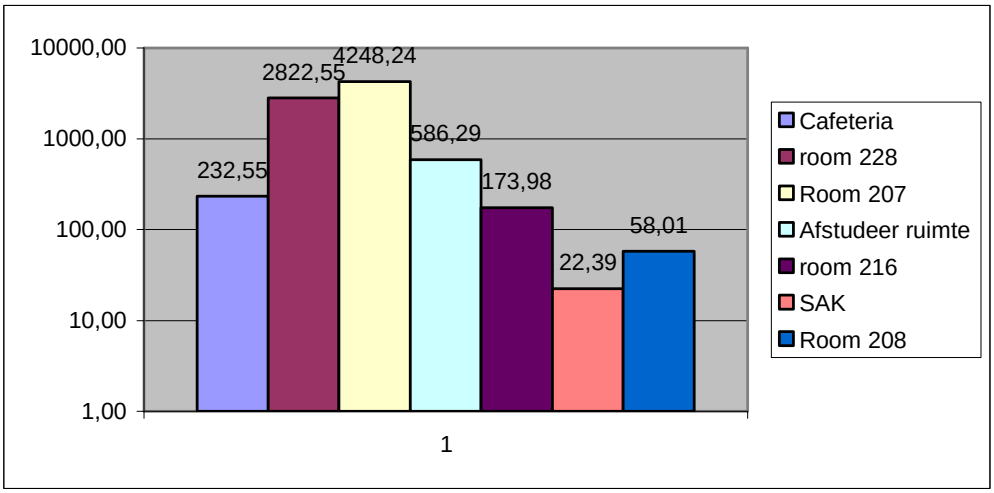


Figure 46: Totals for each point during the stress test for two broadcasting access points instead of three.

In figure 46 the results of the totals from the stress test are shown in a bar graph. The first thing noticeable is the tremendous increase of the detection of other Pin Points then Room 207. Room 207 is still the signal that gets highest score, but it is a lot lower then before. In fact Room 228 is getting over half of the score Room 207 is getting.

Now when looking to the map this might look a bit weird as the neighbouring rooms 208 and SAK are getting the lowest scores. This however makes a lot of sense, since one only uses two signals the range band of a Feature can pass extends like a circle around that point. When thinking of the locations of Geoloc1 and Geoloc2 and their distances in relation to the two highest scoring Pin Points, one can see the distances are very close to one another.

So basically because the distance between GeoLoc 1 and the Pin Point in 207 is close to the distance between GeoLoc 1 and the Pin Point in room 228 and the same holds true for GeoLoc2 they basically the system is easier to confuse. Normally the third point would settle this matter and provide a third score that would be different.

Even with this, it was surprising how well the system performed when knowing it was crippled before hand. It does not get a maximum score by far but still the highest.

Over time

As before these same values where also plotted over time, this time in figure 47. This figure does show that room 207 still has the better scores. However it also shows quite a few false positives for the other rooms. Again this can be explained by the fact that with only two access points there is less of a unique signature.

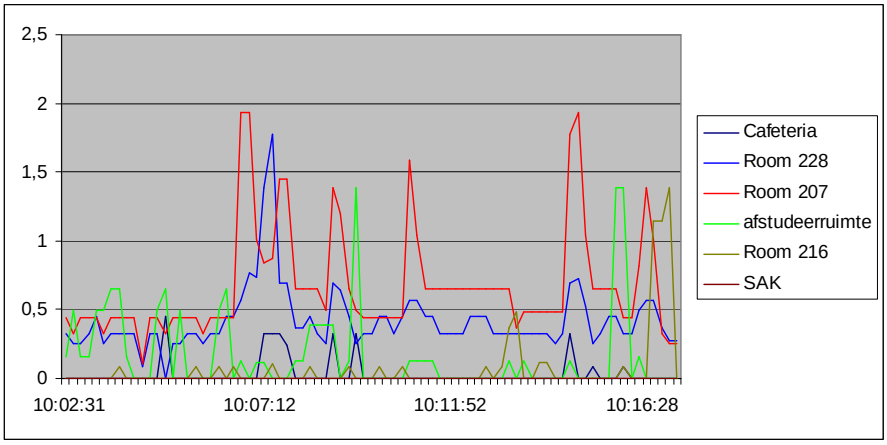


Figure 47: A time slice of the results for each point during the stress test for two broadcasting access points instead of three.

Outside of this the system does seem to be able to place the device in room 207 for quite a period of time, while returning good scores. This was really surprising, as the initial expectations where for the system to loose the location of the device most of the time. What is remarkable is that the rooms that are not close to these bands seem to score very low, thus again showing that Signal Strength/Quality in combination with Calibration Pin Points are more than capable of finding smart devices. By adding a third room the system would be very precise in tracking the device.

False positives

As expected there was a significant increase in false positives in this test. This time however there is not a solution as they are basically false positives that are positive to the system. Unless one adds a third signal there is little to no fix for this problem. Especially during the

path test that is to follow one will see the problems arising when only using two broadcasting access points.

Impact of the threshold

A threshold in this case, unlike before would actually do little to no good, as often the actual point itself would perform under the mark or just slightly above it. All rooms that actually would score low before now score very low. They cause little to no false positives compared to the ones that are in the same distances from the access points earlier discussed. Adding anything above a 0.5 threshold would cause more signal loss and would even cause for the wrong Pin Points to be identified a lot more then the actual Pin Point.

7.5.2 Path test

Now very interesting to see is the Path test, our interest would be if it was even possible to follow the signal over time. During the stress test the result had shown that thresholds might actually have a negative effect, and it was expected to affect the path test as well. The path test itself is shown in figure 48 using the same techniques as before with the colouration and alpha channel.

Figure 48: A side-by-side comparison of the measured Feature scores. The least transparent means a higher score for a point, red is the end point green is the beginning. The colouration between the two indicates the transition. The left picture shows the result with no threshold, the middle is with a threshold of 0.25 or higher scores and the right are for 0.5 values or higher.

Outside of that the aim was to see if it was even remotely possible to follow the path of the device. How abundant and significant the Loss of Pin Point was compared to the three access point tests. A huge amount of false positives was expected, which this time might not be possible to fix with a threshold. This threshold would also affect the score of the actual Pin Point.

Following over time

Interesting is that when looking to the lower 0.25 and 0 thresholds the system is actually able to follow the path in a very coarse manner, We know that we have started in room 207 and ended there, which the system actually shows. However the exact rout is not easy to make out, it remains a guess by the amount of false positives generated.

In fact we might as well have started in room 211which is then neighbouring room. There also is a slight indication of starting, a green blip, in room 228 which like before makes sense as it is the room with similar distances. Also like the uncorrected signal with three points there seems to be a lot of doubt between room 208 and 216.

Loss of Pin Point

There seems to be little to no loss of Pin Point, which is not surprising as there are fewer broadcasting access points that can malfunction or stop transmitting. It was not a big problem before already, so with fewer access points it would actually be less of a problem. There is however some Loss of Pin Point caused by false positives, but this is not because of degradation in scoring of the actual Pin Point because of Signal Strength or failing Access Points.

False positives

There is a tremendous increase of false positives, in fact when walking through hallways it even wants to place you in 2 or 3 rooms at once. Also when one is in a room, it sometimes places you in another room The system still is able to place the device in the right room for a good amount of time, but the removal of one of the access points certainly had an significant negative effect. There is however little one can do to improve this outside of just using three access points instead of two.

Threshold impacts

As was already expected during the stress test, a threshold does little or nothing to help. Yes it removes some of the false positives but it also removes some of the actual points one does not want to loose. In fact in this case a threshold would not be wise to add at all, as it shows that when adding a 0.5 minimum threshold there actually is a different starting point, room 211, then the actual starting point.

Reason for bad results

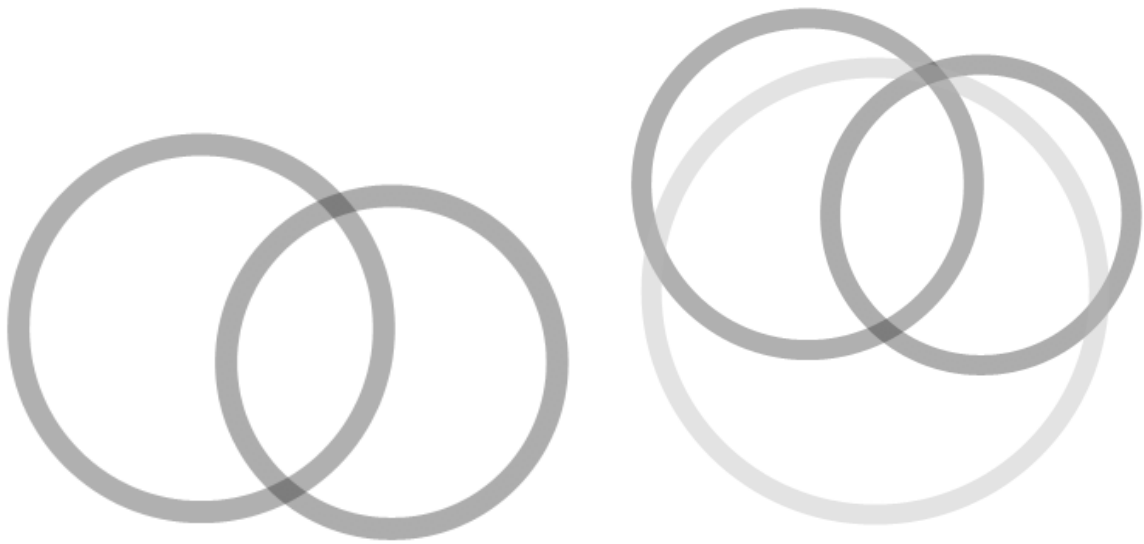


Figure 49: A simple diagram showing the different signal overlap for 2 and 3 signals. Where the coloured bands are the ranges for particular signal strength at a given spot. This range can ideally be obtained in a perfect circle around the transmitter. This of course is an unrealistic assumption for a real test.

Since the system has only two signals there is more of an overlap then with three. Shown in figure 49 is the reason for this overlap. Where three frequency bands only have one unique point of overlap two overlap at two points. This means that system might confuse rooms with each other since it gets the same results in each. Of course the above drawing is in an ideal situation and the Multicast and Line of Sight problem might make rooms in which Pin Points

are located more unique. This however can also add to the problem, as multicast can cause for regions to over lap closer together and thus make more rooms fall into this overlap.

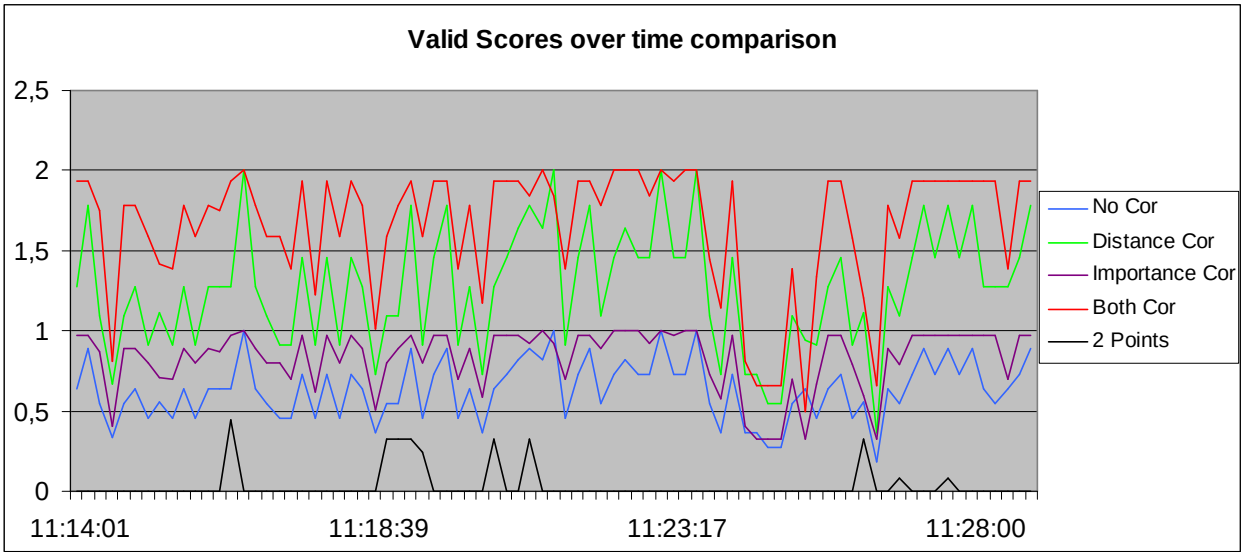


Figure 50: A time slice comparison of each of the valid scores of 15 minutes.

7.6 End result

7.6.1 Unexpected

We did not expect the 2 signals tests to perform as well as it did and the normal uncorrected test to do so well. It meant that the Feature weights and distance corrections later added had smaller effects then originally planned but still an effect none the less. Good Features seem to allow for simple classification. Another unexpected turn was the instability of the Pin Point scoring during the stress test when a weight correction was not included. It was not expected to be that as strong as it was, but where glad to see that better giving the better performing Features a higher weight helped against this minor problem.

Both of these things can be seen in the graph in figure 50, which shows that the two signal tests actually is not performing that badly but has a lot of loss where it has false positives, it does however also show that the uncorrected test and the distance only tests sometimes can contend with the importance weighted ones, which often lie very close to one another in values alone. During the path test the importance weights showed their benefits, as it was better able to indicate a Pin Point then the distance only and uncorrected test.

Unexpected was that the Feature Weight corrected scores show a higher drop and raise in the average difference between these shown in Table 3 of the results. This can be seen in both the Nearest Neighbour corrected scores as the scores that only use this Feature Weight. It means that when the score does drop it drops slightly further then in the other tests, but it also means that the increase of a raise is higher. This is probably caused when a higher scoring Feature does fail or succeed; its drop or raise is also causing a higher impact.

Luckily the frequency of drops and raises was tremendously lower, as was shown in Table 2 of the results, in these tests then in the other tests, thus meaning the signal is a lot more stable. This stability was something we did not expect to happen to such a high extend as was shown in the test.

7.6.2 Expected

It was nice to see the theory confirmed that adding Calibration Pin Points is a big help in the localisation of a device by using the Signal Strength/Quality of an access point. It even performed better then was initially expected. Also, as was expected, the devices could even be on the move and still the system was able to track which room it was in or had visited in the past.

The system also was as fast as was expected, it was able to track a device easily 7 times a second while recording all of the information used for the three Access Points tests and writing these results to a the hard disk. During this it also was running the GUI and showed the location on screen. An average computer would be more then able to track multiple devices at once without breaking a sweat.

Also as expected the system is cheap to implement, as the routers themselves where the cheapest ones we could find and still it performed very well. This basically is in the spirit of the Smart Houses for All project earlier mentioned in this document.

8 Conclusions

The problems regarding Location aware systems for Smart Housing is a complex one. Since one has to do more then just find the position of people, the system also has to identify the person. Already a lot of research was done in this field with various results, but mostly all of them interesting even if they are not completely in the line of this paper.

In this paper we looked at the usage of a Signal Strength Identifier for this problem, to be specific to create a map that would help deal with the problems a building provides regarding Signal Strengths. We looked at different techniques, some using RFID tags while others used Infra red sound, Cameras and IR light. We also looked into different location techniques that used a Time of Arrival or Angle of Arrival method. Before we settled on the idea of using the Signal Strength of a cheap Access Point instead.

From this the work was divided into two lines. The first was discussed in this paper, and mostly entailed the creation of the Pin Points and the extraction of viable information from this raw data. This would create a map for the second paper to use for a more precise localisation of the device. These Pin Points could then be placed through out the building as to create beacons of calibration and thus helping with the problems of multicasting and line of sight.

For this solution we have build a program that would use cheap and ordinary Access points or Wireless routers to locate a smart device location with the precision of a Pin Point. This would test the efficiency of the Pin Point system. Since the precision of finding the earlier created Pin Point again after time mend that they can be used to provide information. With this in mind the following research questions where stated and answered.

Is it possible by storing measuring values to solve the problem of a buildings layout and interference?
To an extent yes, some problems with multicasting and interference remain, although a majority of the interference can be filtered and taken into account. The precision of the system would also benefit with more Pin Points rather then viewer. Of course better quality transmitters would also have a more stable signal quality.

To what level is it possible to negate this interference, by using these stored values and the Features extracted from them?
As shown in the results and the discussion of these, the interference can be filtered out for a major part. Since the system would have at least one Calibration Pin Point in each room, it would provide with a decent map of the decaying strengths. Thus corrections can be made to localize the device with these Pin Points as Calibrating factors over the measured strength for that location.

What other features can one add besides the stored data, to further negate this interference?
From the stored data Features can be extracted, these Features do not all perform at the same level of quality. To help with this we gave the Features weights, better performing Features got a heavier weight then those of worse performance. This seemed to provide a more stable detection and higher scores in general compared to a more erratic reading to a test without these weights.

Another extra feature implemented was a distance correction; since a person can only walk in a certain speed it would be foolish to give all of the Pin Points the same value. With this correction points that where far away where now filtered away to a level of no longer being a threat. Especially Pin Points at another floor could now be filtered out as a person would not be able to walk there as fast as he would to other points on the same floor. This correction provided with a stability of remaining at a point and filtering out false positives especially when considering a path test.

To what extent can this stored data, Features and other features be used to find a device with stored data location precision?
This proved to be quite a capable system of detection for Pin Point precision, outside of some minor flukes the system as able to find the device a large amount of the time. Especially with both of the features added it performed very well as can be seen in the discussion of the results.

In conclusion the system performed better then expected, especially during the tests where the device was tracked over time. It was actually really possible with quite a precision to track which rooms the device had visited. It was also interesting to see that Signal Quality or Signal Strength is very useable for localisation of smart devices. In fact it was surprising that the amount of false positives where as low as they where, even in the first tests. Also surprising was that it had little to no Pin Point Loss even with the cheap routers used in the simulation.

Because of the cheap hardware used we also proved that the system is quite cost effective and therefore a good contender for indoor localisation in the Smart Houses for all project. With more and more smart devices becoming available, with constant decreasing costs in a market that is already quite saturated, even the device itself should not be a big contender on the budget. If devices however are a problem and Motes or other specialised devices are a requirement, the simplicity of just sending the signal strengths to a server would mean that any device that can receive signal strengths could be used. With just a small program that sends the received signal strength to a server on that device one should already be able to properly track the device from room to room. This and the results we received during research are our strongest points for our system.

There was however an expected reduction in precision during the tests with only two access points There was an increase in false positives, which was to be expected. But the system did show it is flexible enough to even handle severely corrupted information and still come to a good educated guess of the position.

It is expected however that with the use of slightly more expensive routers which have a more stable degradation of the signal but also a more stable signal strength the system would benefit especially when it has to work on a 24 hour 7 days a week basis. In which more is asked from the Access points and thus a better quality signal and hardware is an added benefit.

That said there is still plenty of room for further research, especially in the generation of Pin Points, the minimum size of raw data, improvements to tracking the device in between Pin Points [16] and learning Pin Points. All of these options will allow the system to grow up and make it even more of a contender to be used for Smart Houses.

From the research done we can conclude that the system performed better then expected, in an actual environment, the system is cheap to implement and easy to port to other mobile devices. This concludes the paper on a positive note and with hope that further research will actually improve and use these techniques to fruition and perfection.

9 Future work

As with any research there is always room for improvement or work based on earlier findings. The findings found in this paper are no exceptions. For instance we ourselves also found a few points in which we think the system can either be improved, would be a good thing to add and to try out.

9.1 Learning Pin Points

The first thing expected to be a huge benefit to the system is the addition of learning Pin Points. Basically what this would do is, if the score of device ranks above a certain minimum or receives the maximum score that that signal strength is added to the raw Pin Point information. Of course one cannot keep adding to the raw information and somewhere there has to be a maximum.

The size of the this raw data, the minimum score before the new signal strengths are added and what data to throw away are all subjects that can be researched. One can for instance research how the size of the raw information relates to the precision of the localisation but also in what way data is removed. In case of the removal of data, is a First In First out model superior to a model that removes the extremes values found or a combination of models. Another nice point would be what the minimum size of the initial raw data has to be for the system to properly start correcting itself.

Learning Pin Points in itself sounds like a very interesting addition to the technique. Which we expect would be nice to look into with further research, but fell beyond the scope of the time allocated in our current research.

9.2 Further precision for location between Pin Points

As my research partner's paper [16] shows we also did some research in finding the devices location in between the Pin Points. It would be interesting to add more precision to these methods that we investigated. We looked into a few of those already, but it would be interesting to see if further improvement is possible. Especially when thinking of following the device actually through the house with a precision that is one square meter or less.

We think this can be done with improving the techniques and smart deployment of Pin Points maybe even multiple Pin Points per room. Techniques which might be interesting to add is to make an automated system that looks to the average degradation of the signal quality between pin points, and when between some points this seems to be above the average it assumes there is a wall in between. For further precision one can also look into different methods and even having the server itself keep a history of the path a person usually walks and learn to improve its precision from that.

These points again fell beyond the scope of our current research in the given time, but still are really interesting to further look into as it appears to be a very promising addition to the system.

9.3 Testing the system en mass.

We already tested the system in a working environment, not a laboratory environment, but it would be nice to do further research in an environment with multiple devices being tracked an

mass. For instance an actual home or working place with an actual server simulating that the server turns lights on or off or regulates the heating, as is shown in the Smart Houses for All projects. This would than be with the addition of our system and multiple people walking around with a smart device with our system running in the background.

Interesting to see then is how the system performs and how precise it actually is, especially if it detects people in the right rooms and if it is properly interfaces with the systems researched and developed by other people. Also it would be an interesting test, as errors are bound to come up a lot faster in such an actual test over a time of days. This again could be researched maybe with the addition of the earlier mentioned learning Pin Points or Precision improvements between Pin Points.

9.4 Multiple Implementations

One last thing that might be interesting to do is to create multiple implementations for different devices and see how they stack up in results. For instance one could make an implementation for Symbian Devices, the new Multi Tasking IphoneOS, Motes that can receive WiFi data, Android Devices, Meego, Windows Mobile and/or BlackBerry. Then look at how they stack up and perform over time.

Especially when walking around and in stress tests it can be very interesting. Since the code is easily ported, as it is just sending it's signal strengths to a server. The Hardware and OS themselves might be able to influence the results to a great extend and thus create very different results per device.

9.5 Further research in General

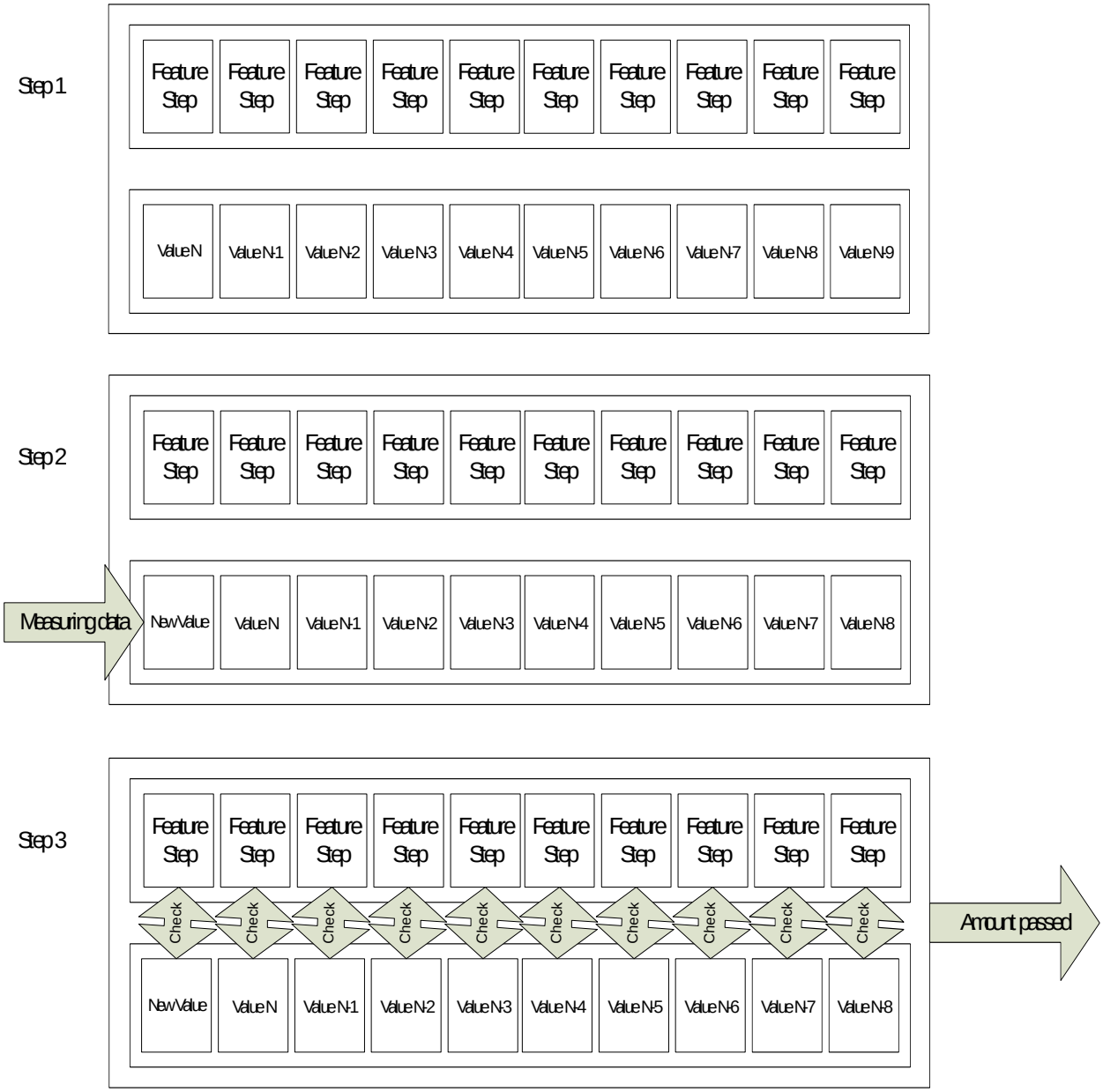
Of course these are just a few points we think are interesting and some other people might find other options or interesting research to do based on our findings. We think that this research will allow for further research topics and that there is great room for further refinement and improvement even beyond the few options mentioned here.

References

[1]: “Object tracking through RSSI measurements in wireless sensor networks”, F. Viani, L. Lizzi, P. Rocca, M. Benedetti, M. Donelli, A. Massa. 8th May 2008
[2]: “RSSI is Under Appreciated”, Kannan Srinivasan, Philip Levis.
[3] http://en.wikipedia.org/wiki/IEEE_802.11
[4] “An Indoor Localization Algorithm for Lighting Control using RFID”, Zi-Ning Zhen, Qing-Shan Jia, Member, IEEE, Chen Song, and Xiaohong Guan, Fellow, IEEE
[5] “An Indoor Localization Mechanism Using Active RFID Tag”, Guang-yao Jin, Xiao-yi Lu, Myong-Soon Park †
[6] “FLEXOR: A Flexible Localization Scheme Based on RFID”, Kuen-Liang Sue1, Chung-Hsien Tsai1, and Ming-Hua Lin2
[7]: “Algorithm for TOA-based indoor geolocation”, M. Kanaan, K. Pahlavan. 1 August 2004
[8]: “Application of Channel Modeling for Indoor Localization Using TOA and RSS”, Ahmad Hatami. May 2006
[9]”Digital cellular telecommunications system (Phase 2+) Location Services (LCS); (Functional description)”, Stage 2 (GSM 03.71 version 8.0.0 Release 1999).
[10] Real-Time WSN-Based Localization of Passive Targets in a Domotic Scenario”, F. Viani, G. Oliveri, P. Rocca, and A. Massa
[11] “An Agent-Based Location System” Peter Jaric
[12] “The active badge location system”, R. Want, A. Hopper, V. Falcao, and J. Gibbons, ACM Transactions on Information Systems, vol. 10, no. 1, pp. 91–102, 1992.
[13 “The cricket location-support system”,]N. B. Privantha, A. Chakraborty, and H. Balakrishnan in Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (Mobi-Com’00). New York, NY, USA: ACM Press, 2000, pp. 32–43.
[14] “Radar: An in-building rf-based user location and tracking system”, P. Bahl and V. N. Padmanabham in Proceedings of the IEEE INFOCOM 2000, 2000, pp. 775–784.
[15] “Spoton: An indoor 3d location sensing technology based on rf signal strength”, J. Hightower, R. Want, and G. Borriello, Department of Computer Science and Engineering, University of Washington, Seattle, WA, USA, Tech. Rep. UW CSE 00-02-02, 2000.
[16] Indoor location tracking using Signal Strength Pinpoints
[17] http://www.microsoft.com/express/
[18] http://creators.xna.com/en-GB/downloads
[19] “Indoor Localization Using Camera Phones”, Nishkam Ravi, Pravin Shankar, Andrew Frankel, Ahmed Elgammal and Liviu Iftode Department of Computer Science, Rutgers University, Piscataway, NJ 08854 {nravi, spravin, afrankel, elgammal, iftode}@cs.rutgers.edu
[20] “Real-Time Monocular SLAM with Straight Lines”, Paul Smith, Ian Reid and Andrew Davison, Department of Engineering Science, University of Oxford, UK, Department of Computing, Imperial College London, UK, [pas,ian]@robots.ox.ac.uk, ajd@doc.ic.ac.uk
[21] “SLAM with a Single Camera”, Andrew J. Davison, Robotics Research Group, Department of Engineering Science, University of Oxford, UK, ajd@robots.ox.ac.uk, http://www.robots.ox.ac.uk/ajd/
[22] NIST smart space system. http://www.nist.gov/smartspace

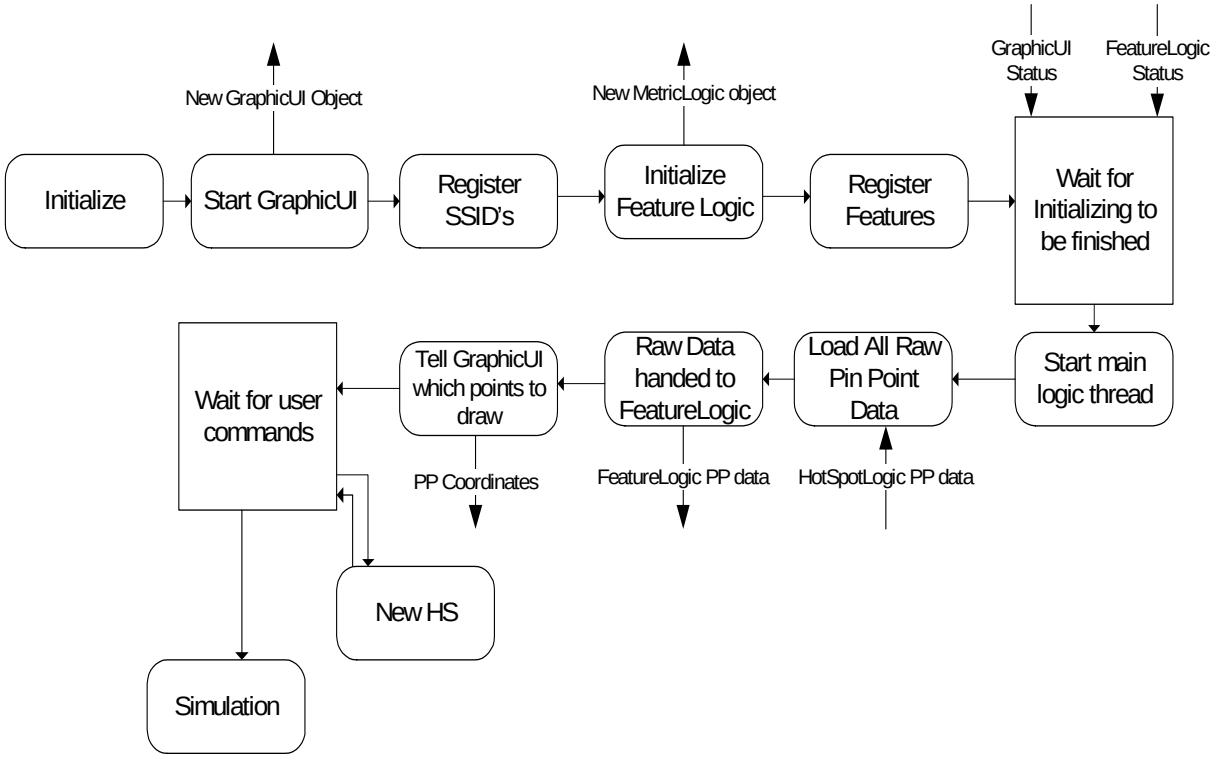
Appendix A:

A diagram showing the basic flow of a Single Value Feature filled Multi Measurement Feature



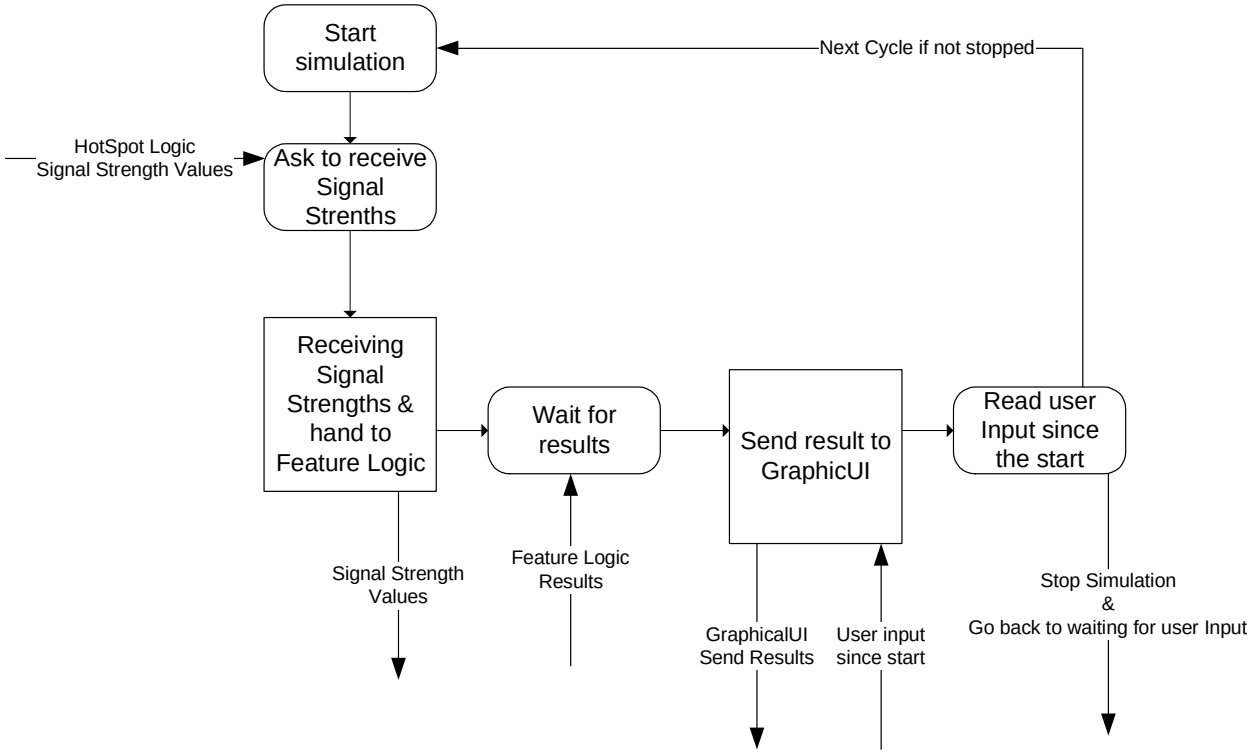
Appendix B:

A diagram showing the flow of the Main program



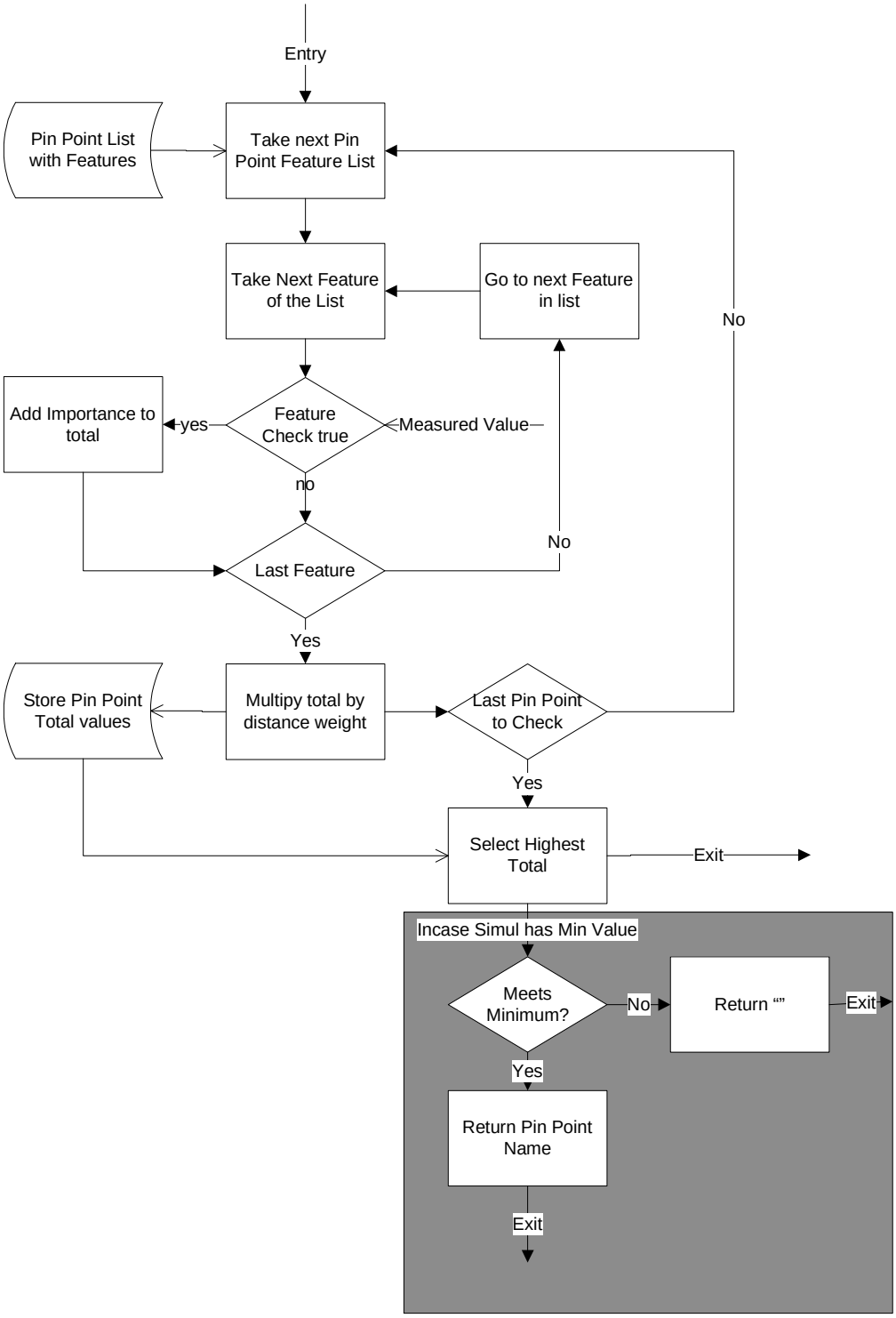
Appendix C:

A diagram showing the flow of the simulation process



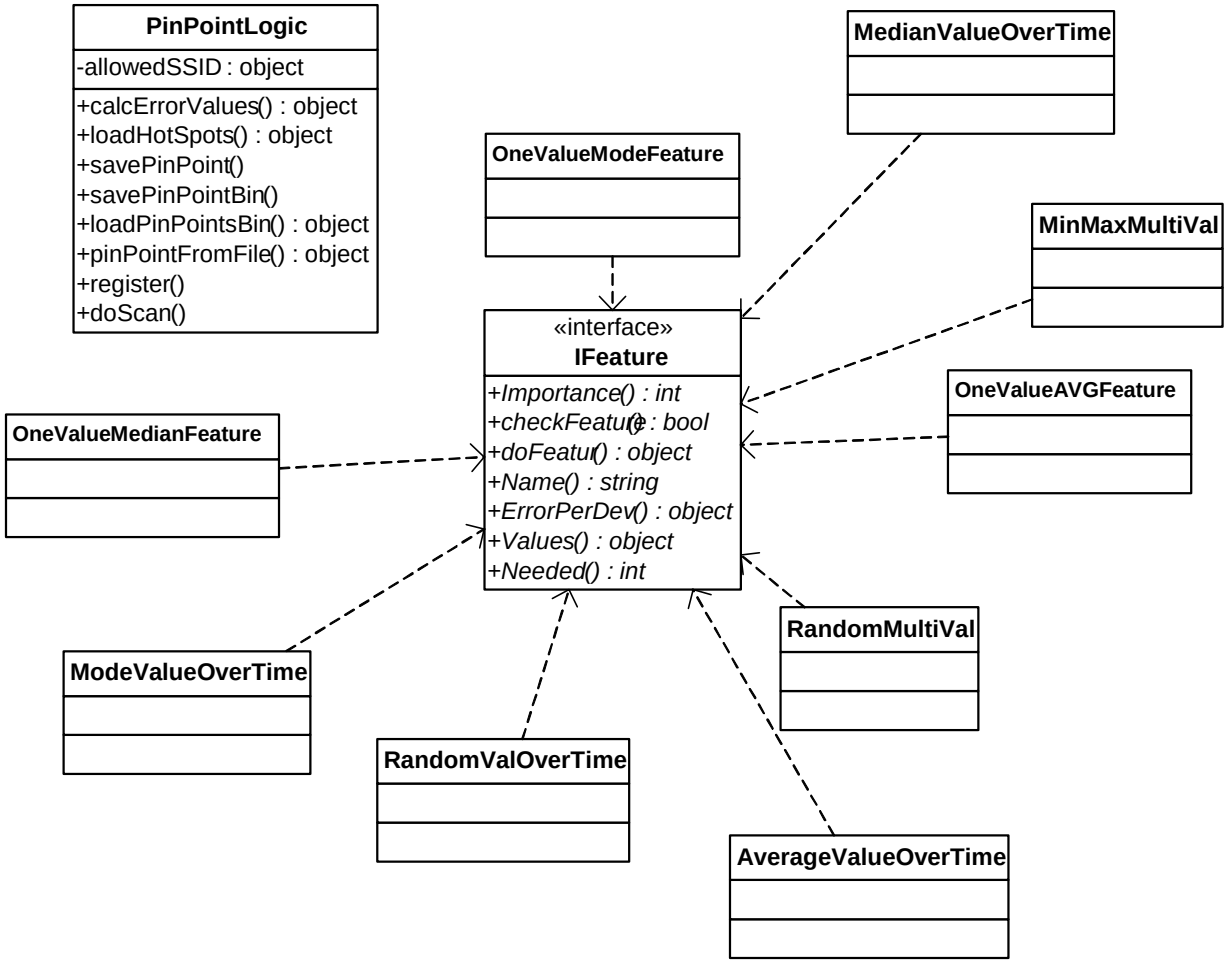
Appendix D:

Showing the flow of a the simulation Cycle Process



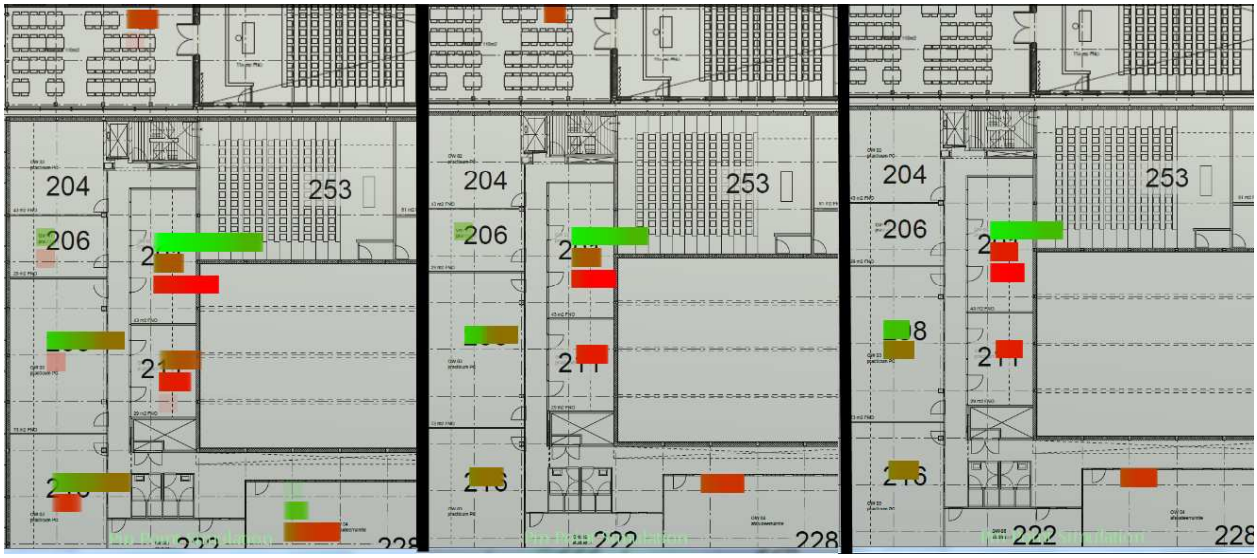
Appendix E:

The structure of the Pin Point Functions DLL, this does not show the attributes and implemented methods for each implementation of each Feature, as it is the same for each one and is later discussed in “Feature Interface and Feature Generation”.



Appendix F:

A map showing the values found over time on a map of the area during the uncorrected test. Starting at Green gradually shifting tint to Red over time. The alpha channel is the strength of the Identification. The left most picture is no threshold; the middle is a minimum threshold of 0.25 and the right a 0.5 threshold.

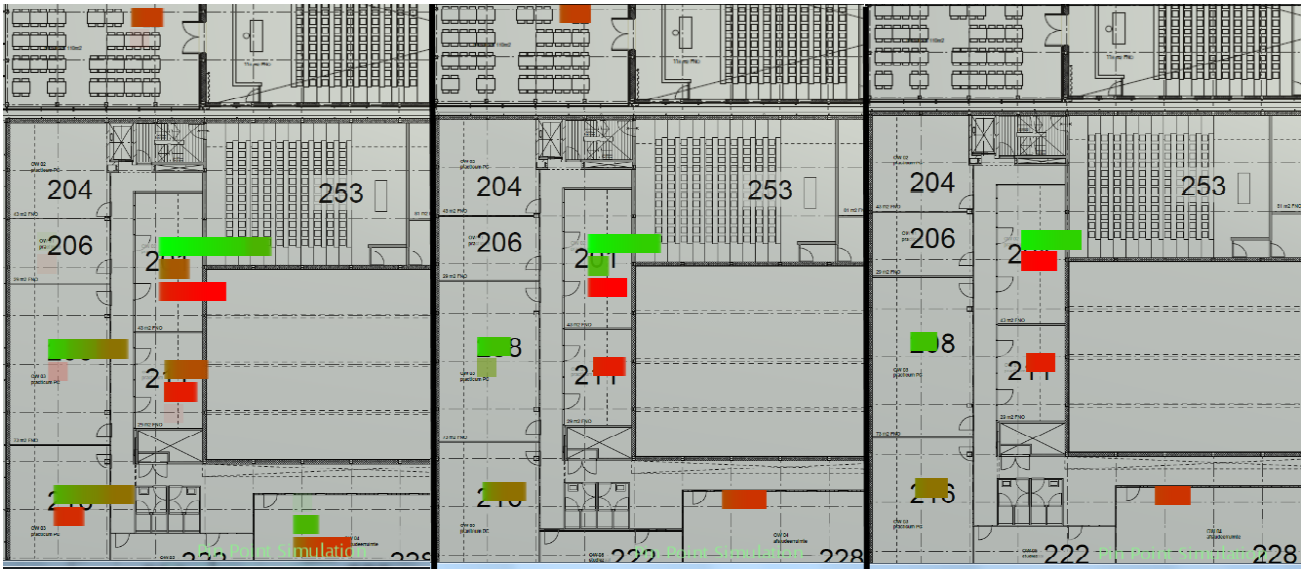


Appendix G:

A map showing the values found over time on a map of the area during the Nearest Neighbour Corrected test. Starting at Green gradually shifting tint to Red over time. The alpha channel is the strength of the Identification. The left most picture is no threshold; the middle is a minimum threshold of 0.25 and the right a 0.5 threshold.

Appendix H:

A map showing the values found over time on a map of the area during the Feature Weight Corrected test. Starting at Green gradually shifting tint to Red over time. The alpha channel is the strength of the Identification. The left most picture is no threshold; the middle is a minimum threshold of 0.25 and the right a 0.5 threshold.



Appendix I:

A map showing the values found over time on a map of the area during a test with all corrections enabled. Starting at Green gradually shifting tint to Red over time. The alpha channel is the strength of the Identification. The left most picture is no threshold; the middle is a minimum threshold of 0.25 and the right a 0.5 threshold.

Appendix J:

A map showing the values found over time on a map of the area during a test with all corrections enabled. However these results are done with only two Access Points. Starting at Green gradually shifting tint to Red over time. The alpha channel is the strength of the Identification. The left most picture is no threshold; the middle is a minimum threshold of 0.25 and the right a 0.5 threshold.