Bachelor thesis
Computer Science

Radboud University

Augmented reality as a general indoor and outdoor navigation solution

Author:
Jeroen van Voorst
s4620593

First supervisor/assessor:
dr. P.M. Achten
`P.Achten@cs.ru.nl`

Second supervisor:
dr. P.W.M. Koopman
`pieter@cs.ru.nl`

Second assessor:
MSc T.J. Steenvoorden
`T.Steenvoorden@cs.ru.nl`

June 25, 2018

Abstract

This thesis proposes a system that combines indoor and outdoor navigation underneath one system using augmented reality. The system uses highly customizable points of interest to determine the route. Then it computes the direction of the target poi on the route and representing this direction in AR. This direction is updated according to the user's current location. The idea is that a visualisation in the real world through AR directions may provide a solution for accurate navigation indoor and outdoor. By using real time footage of the live world, navigation may become less dubious and more straight-forward than the use of maps or signs.

Contents

Chapter 1

Introduction

Navigation is a subject that has been researched for decades now. Still, combining indoor and outdoor navigation while also visualizing the navigation path effectively remain to be noticeable problems. Whereas GPS works outdoors, this signal is too weak to function accurately indoors. Hence, another method is needed for indoor navigation. In order combine indoor and outdoor navigation a solution is needed that links the two underneath a general system while also presenting an effective visual path. Physical maps leave many things to be desired. One has to memorise the way by glancing over the map again and again and is very prone to making a wrong turn.

Combining indoor and outdoor navigation through a potential hybrid navigation system -combining more than one navigation system with each other- and providing correct real world embedded visual aids through augmented reality is the proposed solution that this thesis brings forward. Augmented reality essentially means adding perceptual digital alterations, like sounds, smells or visuals to reality. In this thesis, AR only extends to the visual senses and does not concern other sensory augmentations. This means that AR in this context means rendering a digital graphic across observed reality to form a composite view. Reality is augmented with additional rendered 'fake' images or 3D-objects. Using this solution, users don't have the problem of having to map the navigation to the real world. Augmented reality can make it as clear as day where the user has to go next using lines, arrows or even labels. Therefore, the main research question of this thesis is: How can an augmented reality system be used as a general solution for indoor and outdoor navigation?

First this thesis proposes a general system that combines indoor and outdoor navigation without the problems of mapping an artificial map to the real world. Then a prototype of the system is represented which makes use of AR and defined points of interest -POIs- to guide a person from one

POI to another along a shortest route. Using the POIs locations, one's current location, the heading of the device and bearing of the target POI, a direction towards a target POI can be created in AR. By ensuring that the POIs are connected only when there is no obstacle between, the direction can be used as a accurate means of navigation. Furthermore, the administrator can set specific objects or labels to be shown at those POIs. The label or object will then be shown to the user through augmented reality at the location of the POI. The system is highly customisable due to POIs and current location approaches being changeable. This system is then useful for, for example: supermarkets, festivals, open house activities or walking routes.

To tackle this problem there are several subproblems that need solving:

- scenario limitations
- the computation of the shortest route
- navigation inside a structure
- navigation outside a structure
- visualisation of this route

The system is realized in a prototype using json formatted POIs, GPS for the current location outside and accelerometer and magnetometer sensors of the device the prototype was build on. The realized prototype works with the exception of indoor tracking of the current location .

Chapter 2

Related Work

This thesis combines three concepts: indoor navigation, outdoor navigation and augmented reality. The most interesting researches for this paper were in the subjects of indoor navigation methods and augmented reality. There has been done a lot of research into indoor navigation (bluetooth, wifi, Rfid, NFC) but in order for them to work with closely positioned POIs high accuracy is needed.
Locata[12] is a navigation system which claims to have Real Time Kinematic (RTK) GPS accuracy levels. RTK GPS uses signals and a fixed location to determine one's position. This has an accuracy of 4 cm or less [17]. The system uses a network of ground base transmitters (LocataNet) and single frequence L1 tranceivers (LocataLites) to achieve this. Pirzadaa et al [22] conducted a thorough research on the accuracy and capabilities of various other indoor navigation techniques. From this list an alternative appropriate indoor navigation method might be chosen for the system. For example, Ubisense(2005) has 14 cm accuracy and is accurate enough to implement with the POIs.

Augmented reality combined with navigation is also explored in a thesis about directional indoor navigation (Sung et al) [23]. Sung made a prototype for indoor navigation based upon QR code POIs. The user has to scan the QR to get a direction arrow shown on the screen to the destination QR or POI. No path is created and obstacles might be encountered. Another difference with this thesis is the broadness. The system proposed in this thesis can be used everywhere and is not solely dependant on QR code scanning for navigation. Furthermore, the system discussed in this thesis combines outdoor navigation with indoor navigation in a general framework.

Kasprzak et al [16] also implemented a feature based indoor navigation solution with AR. When looking at specific objects in the building the user will be given navigation instructions based on their route. These routes are text based instructions and are fetched/matched. However, feedback suggested that “instructions to change in continuously, like in GPS systems.” Instead, the above system provides an augmented arrow on top of the feature recognized object. If the user looks away the direction is no longer real time shown. In the system of this thesis the directional arrow is consistently updated.

Another interesting system is the one described by Brush et al [15]. They introduce a navigation system that relies on activities. It uses ‘trails’ to navigate a user back to where they came from. These trails are the activities like “take 50 steps north” or “go up 2 elevator levels”. These activities are automatically generated. This is different than giving directional info but could be implemented as a navigation system as well as a follow up to this thesis. Interesting to note here is that at the University where this thesis has been conducted, a start-up company has been experimenting with this as well. They had placed QR-codes all over the campus. You can scan them or manually select one from a list on your phone on their app and it will give in text form activity based instructions, like go down the stairs on the right.

Hile et al[14] have implemented outdoor navigation with augmented reality on phones. Here they used landmark points (location with a latitude and longitude at a certain landmark). When arrived at those points the system shows users images of the landmark with augmented direction arrows. These stationary images are pulled from a database containing photo’s that are made of the landmark at specific GPS points.

Chapter 3

The augmented reality navigation system

In this chapter, the proposed system will first be explained in theory. After this, it will be explained how it can be applied using various existing techniques. Then lastly the thesis will focus on how the realized prototype functions.

3.1 Proposed system

3.1.1 System's architecture

The system is dependant on the points of interest(POIs) that the administrator of the system sets up. These POIs have a latitude(lat), longitude(lng), neighbours and a floor all defined by the administrator. All POIs must be reachable from each other; there are no isolated POIs. POIs can be both inside and outside, as long as they have the required fields.

Each of those fields play a crucial part in the system for outdoor and indoor POIs:

- The neighbours field is required so that the system knows what POI's are connected without an obstacle in between.
- The latitude longitude fields are there for determining a user's position and arrival at a POI.
- The floor field is required for indoor POIs to distinguish floors from each other and determine stairs or elevators. Outdoors this can be a default base floor value.

Using the POI information, the system can then compute the shortest route and navigate the user from one POI to another.

A result of this particular scenario is that it's limited to the administrator's demands. It is only possible to navigate between POIs that the administrator of the system has set up. Navigation across two arbitrary latitude longitude pairs is therefore not possible.

3.1.2 Inner workings

Route computation

The user travels between POIs from source POI to destination POI. The path between can only consist of a connection between POIs: the shortest path is simply a list of POIs the user has to visit in order to get to his destination. The system must therefore calculate the optimal shortest path. To get the relation of the POIs the system takes the neighbours and the appropriate latitude and longitudes and computes the distance between the POI and its neighbours.
From the above a graph is made with each vertex a POI and each edge a connection between the POI and one of its neighbours. The edges have a weight which is the shortest distance between the two POIs. From the system's architecture there can be no obstacles between a POI and its neighbours and therefore, the edge can be created and represent a clean path. Using a search algorithm (A* in this thesis), the graph can be traversed from one vertex to another using minimal weight and thus minimal distance. Every vertex visited is added to the route to create the route list.

Navigation

The route is computed which gives the POIs from the source to the destination. The latitude and longitude of the POIs can be determined from the administrator. Important here is that every POI is reachable by travelling a path of POIs without obstacles in between them. Not every POI needs to have a description, but can simply be made as a navigation point.

As a prerequisite the current location can be accessed as well through for example GPS for outdoors and Locata or QR based navigation for indoors. An optional field can be made for POIs that tell whether a POI is indoor or outdoor. This way the system can switch between accessing current location methods. Then, through converting the latitude and longitude to a location object, the bearing can be calculated. This bearing [7] must be the bearing between two locations along the shortest path. There are two kinds of north, the magnetic north and the true north. The magnetic north [18] is the point where a magnetic arrow points to under the influence of the earth's magnetic field. The true north [19] is the direction to the north pole of the earth from any location along the meridian.

A device that uses a magneto and accelerator meter will determine one's heading with the hardware compass in degrees east or west of the magnetic north. If the device computes the true north, the bearing is according to convention in east or west degree of the true north. This thesis will use east. In most if not all real case scenario's the hardware compass will return the heading of the magnetic north and the bearing will return a degrees of the true north. In order to let both be compatible with one other the system has to turn the magnetic north heading into a true north heading. This can be done calculating a 'magnetic declination', which is the angle difference in degree between the magnetic north and the true north on a horizontal plane.

Using this one can compute the direction of the target POI by first converting the heading of the device to the appropriate north and then update the offset to represent one's bearing to the destination. This way, the direction arrow will point towards the destination POI instead of north.

Having this direction the system then navigates the user from POI to POI until the destination POI is reached. There aren't any obstacles between the user and the target POI, therefore the user can simply follow the arrow's direction on the screen to get to the next POI. The system leads the user from POI to POI, which means that at each POI the bearing to the next POI on the route needs to be computed. The direction of the arrow also needs to be updated. This is done by checking the user's current latitude and longitude in 5 equal intervals between every source and target poi. If the user is within a meter of the target POI then the POI target is updated to the next POI on the route. This goes on until the user has arrived at the destination.

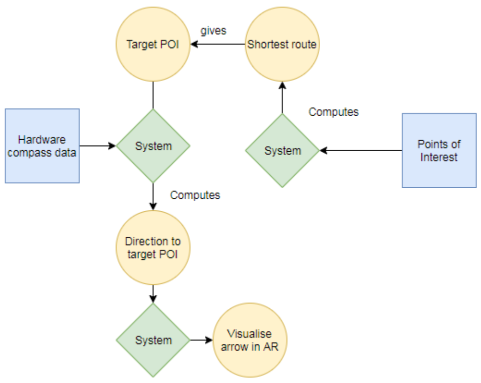


Figure 1. The System's architecure

Figure 1 shows a general flow chart of the system's architecture.

The system also has to detect whether the user has deviated from the path and let this be known to the user. Using the fact that the POIs are neighbours and neighbours are defined as straight line distances due to those being the shortest distances, then the system can calculate a cross-track error [8]. A cross-track error is the distance of the current location to the line between the target and source POIs. Then the administrator of the system can determine if this distance is bigger than 10 meters, show a notification that the user is going the wrong way.

Labels, 3d objects and the compass can be displayed as a helping service for the user. POI descriptions, navigation indicators like arrows or appropriate icons are examples of interesting AR objects at POI coordinates.

3.1.3 Scenario sketch

Three kind of scenario's are possible: only inside POIs, only outside POIs or both. Say there's a person named John, who wants to navigate from one POI to another. Then there are the following three sample scenario's. They may change depending on the navigation system used (for example John might have to scan a QR-code inside to enable the system to get his current location and update the arrow).

Supermarket with multiple floors

John is standing at the entrance of a supermarket. Using the system that the supermarket made available to him, he opens the system on a device (say phone), opens the correct application, selects the "entrance" POI from a scroll-down list of available POIs and selects "The meat department" as the destination POI. The device computes the route of POIs from the POI with the description "entrance" to the POI with the description "The meat department". The device opens its AR view showing a directional arrow towards the first POI on the route. John follows the arrow's direction until the end of the lane, the arrow 'updates' and changes direction. John turns around to follow this new direction following the arrow once again until he reaches a stairway. The the arrow's updated with a message: "The next POI is on the next floor please go up the stairs." John follows the device's instructions and once upstairs follows the arrow's direction once again until he arrives at his destination. He does this until he sees the label with "The meat department" on his device. The device notifies John he has arrived.

An outdoor festival

John is standing at the entrance an outdoor festival. He opens the system provided by the festival on a device. He then selects the "entrance" POI from a list of available POIs and selects “The main stage” as the destination POI. The device computes the route of POIs with source the POI with the description “entrance” to the destination POI with the description “The main stage”. The device opens its AR view showing an directional arrow towards the first POI on the route. John follows the arrow’s direction until he faces stage 2 which is in the way. The arrow ‘updates’ and changes direction, leading to a next POI that will lead him around the stage. Once John is around the stage by following several updates of the arrow, he follows the arrow one more time and finds his destination. He sees an Main Stage 3d font on his device and the device notifies John he has arrived.

Open house of a University

John has arrived with a bus at “Erasmus Building”. However, he wants to go to the “Huygens Building” . He opens the system provided bym the Radboud University on his device and selects the “Erasmus Building” POI from a scroll-down list of available POIs and selects “Huygens Building” as the destination POI. The system does the same as in scenario 2 but John accidentally goes the wrong way and ends up two buildings away from his destination. The system detects this deviation from the path. It will check whether a new POI lies between the destination and John. Depending on whether there exists such a POI the system will either update the arrow to this POI or otherwise keep pointing towards the current target POI. A warning message will be shown that John is not following the arrow. John notices this and finds his way to the Huygens building and wants to go inside. John sets his current location “Huygens Building” and destination “HG.00.300”. The system will continue to update the John’s position. It will also keep track of the target POI Navigation implementationusing John’s actual position. John finds his way as in scenario 1.

Comparison

There is common ground to be found in these three scenarios as well as several differences. In all three cases John fundamentally does the same: set an source and destination POI and walk according to the augmented arrow. However, in the background the app does have to switch from current location fetchers. Another difference is that in scenario 2 additional ’navigation’ POIs will have to be defined to get John around the stage. Furthermore, in the third scenario a deviation from the path has to be found and corrected.

3.2 Proposed Prototype

3.2.1 Platform choice

The proposed system is compatible with every AR device that has an accelerometer and magnetometer sensors. These sensors are necessary for directional calculations and should therefore be present in the chosen device. Furthermore, the device needs to be able to locate the device at any point in time during the navigation process. The manner of locating the device is of no importance, as long as the latitude and longitude are provided with one meter accuracy. This is due to the arrival check and path deviation detect.

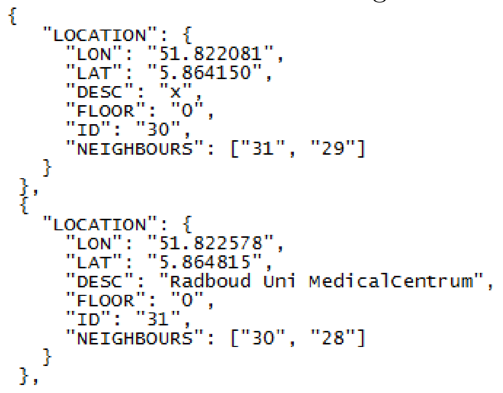
The arrival check is done by consistently checking the updated location of the user and whether the user is within one meter of the target POI. This checking is done by calculating the distance of the user's location to the target POI. If so, the POI is set to arrived and the arrow updated to the next target POI.

The compatibility of the device with AR means that the device must be capable of running a service or development kit that renders a digital graphic across observed reality to form a composite view. In this thesis, AR only extends to the visuals and not other sensory augments. Reality is essentially augmented with additional rendered 'fake' images. A direct consequence of this definition is the need for the device to display the observed environment. Therefore, the device will need a camera.

3.2.2 Design and algorithm approaches

Shortest route computation

The POIs' information can be given to the system using for example json:



```
{
   "LOCATION": {
     "LON": "51.822081",
     "LAT": "5.864150",
     "DESC": "x",
     "FLOOR": "0",
     "ID": "30",
     "NEIGHBOURS": ["31", "29"]
   }
 },
 {
   "LOCATION": {
     "LON": "51.822578",
     "LAT": "5.864815",
     "DESC": "Radboud Uni MedicalCentrum",
     "FLOOR": "0",
     "ID": "31",
     "NEIGHBOURS": ["30", "28"]
   }
 },
```

Figure 2. The POI list in json format

For computing a shortest route the system uses the A* algorithm. In the context of a weighted graph, A* is with the correct heuristic complete, optimal and admissible. This makes it suitable for the weighted graph of the system. A* has access to the POIs coordinates so that it can be used for A*'s heuristic.

A* needs an admissible heuristic in order to always compute the optimal path. Admissible means that the cost that the heuristic estimates is not higher than the actual lowest possible cost. The heuristic in this thesis' context is the shortest distance between the current location and the destination. If the POIs are near to each other, which is usually the case in all scenario's, then the taking this distance using the latitude and longitude straight line connection is acceptable; the error margin is small.

There are two ways to get this distance: the Manhattan distance and the Euclidean distance. The Manhattan distance [21] determines the shortest distance in a constant one dimensional plane where one can only move vertical and horizontal, whereas the Euclidean distance [20] has no such limitations and can determine the shortest distance in every dimensional plane. Therefore, the Manhattan distance in this thesis context is not as effective as the Euclidean distance. Because the neighbours are defined to have no obstacles between them, a 'straight line distance' will work as the shortest possible route. However, the longitude depends on the latitude, because the longitude is smaller at the top of the earth say north pole, than the equator. This can be corrected by multiplying the longitude by the cosine of the latitude.

Computing the shortest route between floors is done by linking the two graph's -that result from two floor POIs- together by a 'stair' POI. A stair POI is a POI that has a neighbour that has a different floor value. An additional edge is created between those two POIs with a constant weight for stairs that can be edited in accordance to the size of the stairs. This so that the Euclidean distance that is calculated based on latitude longitude coordinates is still in scale with the stair weight.

If the area is too large (say larger than a city) and the error margins become noticable, the Haversine Formula [8] can be used instead to calculate the great circle distance, which is the shortest distance between two points on a sphere. However, since the earth is not entirely a sphere the Haversine is at least as accurate as 3m in 1km. In such large areas there is no need to compute the shortest route between floors, because buildings as large do not exist, let alone a POI with a stair neighbour that has a distance of more than a city.

Navigation implementation

Now that the shortest route list of POIs together with the corresponding latitudes and longitudes is available, two things are needed to set the arrow.

Firstly, the heading is needed of the device. The heading can be extracted from the hardware compass (accelerometer and magnetometer). The convertion of the magnetic north to true north can be done by looking up the magnetic declination on the World Magnetic Model 2015 Declination models [11] and converting it to a 360 degree model instead of the -180/180 model. This goes for all degree operations if the formula returns a value in a -180/180 model. The convertion could happen by for example mod operation $f(x)$ +180 mod 360.

Secondly the bearing of the two POIs is needed for the system. The bearing can be calculated using the formula $bearing = atan2(sin(\Delta q) * cos(q1) * cos(p1) * sin(q1) - sin(p1) * cos(q1) * cos(\Delta q)$[8]. where $p = (p1, p2)$ and target location $q = (q1, q2)$ in latitude longitude respectively and Δq is the difference between the longitudes.

Now the offset of the arrow can be calculated from the user's current position, be it POI or current location. This offset is updated with respect to the repeated computed heading of the device and current location. This way the arrow will always point to the target POI no matter where the device is faced or placed. For the outdoor current location updates GPS can be used. For indoors, one of the navigation methods described in [22] can be utilized or a simple qr code based system can be implemented.

The system also has to detect path deviations using the cross track error distance. This distance can be calculated using the formula: $cte = asin(sin(d13) * sin(b13 - b12)) * R$ where d13 is the distance between the source POI and the third point of the line, b13 is the bearing between those POIs, b12 is the bearing between target POI and source POI and R is the earth's radius (6.371 km [6]). So for this calculation another bearing between the current third point location and the source POI is computed using the method above.

Visualization

The visualization of the arrow and other objects in augmented reality have to be rendered. For this an 3d object is needed, which can be created with a program like Blender [5]. The object must then be rendered (for example using a rendering program like openGL [9]).

Once the 3d objects can be rendered the position of the objects need to be determined. Labels could for example show the POI's description and be rendered at the specific latitude and longitude of the corresponding POI. The arrow itself however is required to be at a fixed position in the view so that if the user moves or moves the devices the arrow's position isn't affected. Otherwise, if a users walks away, the arrow will become smaller and further away from the user. Also, a fixed arrow helps the user to immediately know where to look at the screen. However, despite of being in a fixed position on the screen, the arrow must still rotate with the device's orientation so that it will always point to the correct direction. Details about visualization implementation are very much device and SDK dependent.

Customization

The customization of the system lies in the POI customizations and delivery as well as the 3d objects and the navigation method. The 3d objects and POI's locations, descriptions, and neighbours are all decided by the administrator of the system. Custom labels can be loaded when looking at a specified coordinate or by looking at a QR code. For example for a supermarket the meat department may have a meat pictograph hovering above its location. Another example would be to display the target POIs position as a (distant) dot. Lastly, the system only needs to have an accurate current location of the user. The approach to this is open to all kinds of algorithms as long as it provides the current location. For outdoor this can be GPS or RTK-GPS for example. Indoors, one of the methods discussed by Pirzadaa et al [22] might be chosen.

3.3 Realized prototype

3.3.1 The prototype

The prototype is made for a Samsung galaxy s7 using android's ARCore SDK [2]. ARcore is an SDK made by google to enable developers on android to build augmented reality applications. ARCore requires android 7.0 and is only able to run on specific devices[3]. I've chosen the samsung galaxy s7 because I've had some experience in working with android studio and the samsung galaxy s7 was the only device available which could run ARCore. Another reason was ARCore is just a few months old and I wanted to test the extend and limits of ARCore.

The shortest route computation using A*, floor differentiation, arrow positioning and outdoor navigation POI updating all work as intended.The shortest route contains a list of all POI's in such an order that the action of travelling to each POI will lead the user to their destination in the shortest possible time.

The objects are made using Blender and rendered using openGL and I used the premade objects of ARCore's compass objects [4]. Then in the AR view and environment ARCore generates its view of the world by detecting visually distinct features. It uses the position and orientation of the device camera relative to the world and the features to create anchors to place objects at. The fixed position of the arrow is realized by first placing a compass base at a fixed anchor compared to the camera and then placing the arrow's base on the middle anchor of the compass's base. Now ARCore tracks the entire compass as a fixed position in the real world view. The arrow however, needs to be able to rotate according to the bearing. This is done via a rotation pose consisting of the principals of quaternions according to the Hamilton convention [13]. A quaternion is a forumla in the form $w + ax + by + cz$ where x, y, z and w are real numbers and a,b,c are unit values. For example in a rotation quaternion the x, z, y are vectors that represent the three Cartesian axes x y and z. The pose rotation used by ARCore uses as quaternion [10]:

$$x = k.x * sin(theta/2)$$

$$y = k.y * sin(theta/2)$$

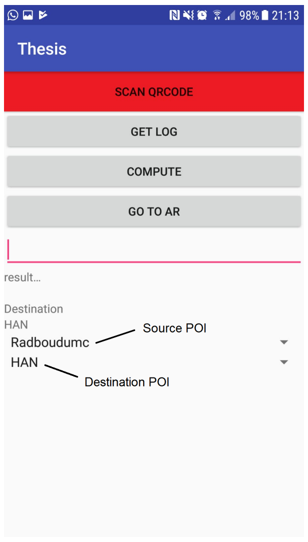
$$z = k.z * sin(theta/2)$$

$$w = cos(theta/2)$$

where theta is the angle rotated counter-clockwise about a unit-length axis k. This means that a rotation over axis y will result in the following quaternion parameters:

$$x = sin(theta/2), w = cos(theta/2), y = z = 0$$

The arrow is rotated in the compass base in the horizontal(x) plane of the x,y,z space by the calculated direction angle.



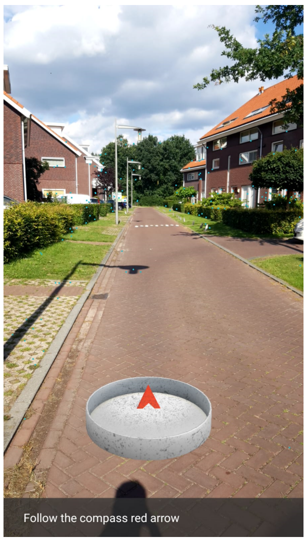


Figure 3.1 The interface Figure 3.2 The direction

Figure 3.1 shows the start up interface. Here there are two spinners: the one on top being the source POI and the one underneath being the destination POI. Once both have been set one of the buttons above can be pressed. The "get log" button is there for me to extract the path into a log on my device. 'Scan QR' reads optional QR POIs and 'compute' computes the route between the set source POI and the target POI. 'Go to AR' should only be pressed after computation, otherwise the arrow will point north. If pressed after compute then the compass base and arrow are constructed (figure 3.2) where the arrow points to the first POI on the path. The blue stips on figure 3.2. are the features recognized by ARCore.

Outdoor latitude and longitude updates are fetched from Google's locationmanager [1] and using this location the distance to the target location is calculated. When this distance is below a meter, the POI will marked as visited and removed from the current route. The next POI on the route will then be set active and the bearing will be recalculated. This results in an updated directional angle and hence an update to the rotation of the arrow.

3.4 Test results

There were three situations: indoor, outdoor navigation and the navigation with the two combined.

First I tried to test the app inside, but since I had no time to implement an accurate indoor navigation method to my prototype. Instead I opted for loading a map into the prototype with POIs scattered all over it. These POI have x,y pixel coordinates instead of real latitude and longitude coordinates. This required the json to have two addition fields namely the x and y pixel coordinates of the POI. The distance could then be calculated the same way as over latitude longitudes. In order to have an appropriate distance estimation, the map needs to be on scale. The prototype was then able to compute the graph and the shortest route. The arrow worked but could not be updated once the next POI was reached due to no active tracking of the user position. Therefore, the prototype only worked correctly on outside POIs. However, the systems fundamentals should work inside as well as long as the app has access to the users current location.

Then, I tested the outdoor navigation on simple routes of a max 5 POI long (all POIs where latitude/longitude pairs of corners of streets) and the arrow straightly pointed me from POI to POI with sometimes flipping around for a second before reconstructing itself. This was due to ARCore being unable to find fixed positions of objects when it cannot recognize any points or poses to latch onto. So when staring at a blank white ground or wall, ARCore has no idea where to place objects because it has no references to the real world. So in case of very blank environments or limited view through the camera (at night for example) the compass and arrow disappear. As soon as ARCore can relate itself to the real world again the compass is reconstructed and reset to point to the correct POI.

Lastly, there was the combined navigation. While using at least two POIs indoors in the route and at least one outdoors, it worked until the arrow had to be updated indoor. GPS wasn't accurate enough indoors and therefore the needle updated either too soon or too late causing wrong navigation. Therefore it did not work as intended.

Unfortunately, there was another noticeable limitation to ARCore: there was no way to render an object at specific latitude longitude in the real world. Objects could only be rendered through user interaction. The user has to press a white node on a created plane and only then will the specified 3d object be shown at that point. Due to this limitation I could not make use of labels and objects for POIs.

Another problem is the accuracy of the hardware compass. The heading has an inaccuracy of about 15 degrees in either direction and strong magnetic fields may influence the readings.

Chapter 4

Conclusion and Future work

The proposed system is a general system for a navigation solution on AR devices. However, the preconditions are that the current location and compass data can be determined. The realized prototype confirms that the system works outdoors. The system fundamentally works the same indoors as outdoors, but for the way to determine one's current location. Hence, the realized prototype hints that it should be possible to navigate indoors the same way as outdoors. The amount of customization in this system is high as well, but the system can only be used in for travel between user/administrator established POIs.

The realized prototype proves that it is possible to combine the system with a stock device for outdoor navigation and indicates that it will be possible for indoor navigation as well. However, it is far from done. Distance indicators, labels or 3d object rendering at specific locations and most importantly an indoor system implementation are still unrealized either due to SDK limitations or time. Future work and research could be done into the system's combination with indoor navigation systems as Locata or activity based systems.

The arrow system may also be upgraded by hard-coding the edges of the POIs as line objects in AR and then map these to real coordinates. Then when a POI is reached the new line to the next POI will be created on for example the ground. This requires, however, accurate latitude longitude object rendering which ARCore does not have. Therefore, comparing different SDKs -for example an system implementation using the IOS side variant ARkit- might provide more insight on what features are needed and what platform provides the best AR navigation experience and why.

Bibliography

[1] Android's locationmanager. https://developer.android.com/reference/android/location/LocationManager, last accessed on 01-06-2018.

[2] Arcore. https://github.com/google-ar/arcore-android-sdk, last accessed on 4-06-2018.

[3] Arcore supported devices. https://developers.google.com/ar/discover/supported-devices, last accessed on 04-06-2018.

[4] Arcore unsupported compass. https://github.com/inio/arcore-android-sdk/tree/unsupported-geo-oriented, last accessed on 03-06-2018.

[5] Blender. https://www.blender.org, last accessed on 4-06-2018.

[6] Earth's factsheet. https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html, last accessed on 03-06-2018.

[7] Location android. https://developer.android.com/reference/android/location/Location.htmlbearingTo(android.location.Location), last accessed on 01-06-2018.

[8] Location formula's. https://www.movable-type.co.uk/scripts/latlong.html, last accessed on 04-06-2018.

[9] Opengl. https://www.opengl.org, last accessed on 4-06-2018.

[10] Pose rotation. https://developers.google.com/ar/reference/java/com/google/ar/core/Pose#Pose(float[],%20float[]), last accessed on 04-06-2018.

[11] The world magnetic model. https://www.ngdc.noaa.gov/geomag/WMM/DoDWMM.shtml, last accessed on 04-06-2018.

[12] Jas Barnes, Chris Rizos, M Kanli, and A Pahwa. High accuracy positioning using locata's next generation technology. 06 2018.

[13] W. (n.d.). Hamilton. *Elements of Quaternions (Cambridge Library Collection - Mathematics)*, volume 103. Cambridge: Cambridge University Press. doi: 10.1017/CBO9780511707162.

[14] Harlan Hile, Ramakrishna Vedantham, Gregory Cuellar, Alan Liu, N Gelfand, Radek Grzeszczuk, and Gaetano Borriello. Landmark-based pedestrian navigation from collections of geotagged photos, 01 2008.

[15] A J. Bernheim Brush, Amy K. Karlson, James Scott, Raman Sarin, Andy Jacobs, Barry Bond, Oscar Murillo, Galen Hunt, Michael Sinclair, Kerry Hammil, and Steven Levi. User experiences with activity-based navigation on mobile devices, 12 2010.

[16] Sebastian Kasprzak, Andreas Komninos, and Peter Barrie. Feature-based indoor navigation using augmented reality, 07 2013.

[17] Dinesh Manandhar, K Honda, and S Murai. Accuracy assessment and improvement for level survey using real time kinematic (rtk) gps. 2:882 – 884, 1999.

[18] US Department of Defense. "magnetic north." dictionary of military and associated terms. 2, 2005. `https://www.thefreedictionary.com/magnetic+northl`, last accessed on 4-06-2018.

[19] US Department of Defense. "true north." dictionary of military and associated terms. 2, 2005. `https://www.thefreedictionary.com/true+northl`, last accessed on 4-06-2018.

[20] Vreda Pieterse and Paul E. Black. "euclidean distance", in dictionary of algorithms and data structures [online]. 2006. `https://www.nist.gov/dads/HTML/euclidndstnc.html`, last accessed on 04-06-2018.

[21] Vreda Pieterse and Paul E. Black. "manhattan distance", in dictionary of algorithms and data structures [online]. 2006. `https://www.nist.gov/dads/HTML/manhattanDistance.html`, last accessed on 04-06-2018.

[22] Nasrullah Pirzada, M Yunus Nayan, Fazli Subhan, Mohd Fadzil Hassan, and Muhammad Khan. Comparative analysis of active and passive indoor localization systems. 5:92–97, 12 2013.

[23] Hyun Sung and Sung Hyun Jang. A qr code-based indoor navigation system using augmented reality. 06 2018.