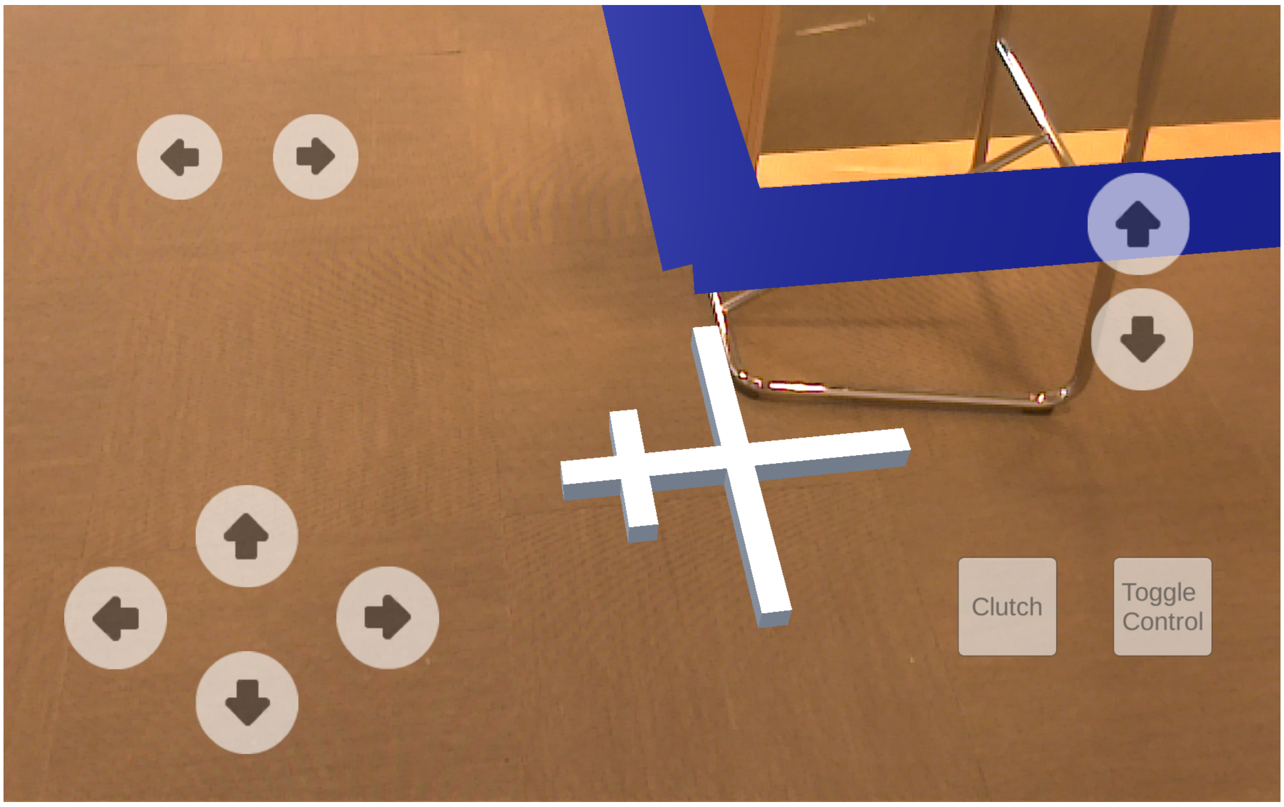
CHALMERS



Sammanstrålning av världskoordinater hos en augmented reality-redo beräkningsplattform och ett AGV-system

Bachelor's thesis in Computer Science and Engineering

JEFFREY SPÅNG
INGRID BRINKENBERG

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

Bachelor's thesis

Aligning the world views of an augmented reality enabled computing platform with an AGV system

JEFFREY SPÅNG
INGRID BRINKENBERG

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2017

Aligning the world views of an augmented reality enabled computing platform with an AGV system
JEFFREY SPÅNG
INGRID BRINKENBERG

Supervisor: Sakib Sistek, Department of Computer Science and Engineering/ Computer Technology
Examiner: Peter Lundin, Department of Computer Science and Engineering/ Network and Systems

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover:
Alignment view of the prototype, showing alignment cross and enabled controls.

Department of Computer Science and Engineering
Gothenburg, Sweden 2017

Aligning the world views of an augmented reality enabled computing platform with an AGV system

JEFFREY SPÅNG
INGRID BRINKENBERG
Department of Computer Science and Engineering,
Chalmers University of Technology

Bachelor's thesis

Abstract

The thesis aimed to synchronise the coordinate system of an augmented reality ready platform and a control system for automated guided vehicles. This attempt was done to enable augmented reality functionality to the control system. The project has been performed on site at Kollmorgen Automation, in Mölndal, and investigated the possibility to create a platform for augmented reality applications by sharing the world coordinates of two different systems. The project was part theoretical to develop a suitable method and part practical to create a testable alignment prototype. The prototype was demarcated to orientation and localisation of a Google Tango ready tablet, communication with the control system NDC8 and visualisation via the game engine Unity. The developed system for synchronisation of coordinates has been visualised to show that an adequate coordinate alignment has been achieved. The method that best met the set prototype specifications was manual localisation of the tablet in the real world. The method was used to shift the tablet's coordinates to match the coordinates in the map used by the NDC8. The finished prototype shows that there is room for further development of the system, but also that it is possible to create augmented reality visualisations of the NDC8, that correctly interact with the real world.

Keywords: AR, AGV, augmented reality, automated guided vehicle, simulation, android, coordinate systems, localisation, control system.

Sammanstrålning av världskoordinater hos en augmented reality-redo beräkningsplattform och ett AGV-system

JEFFREY SPÅNG
INGRID BRINKENBERG
Department of Computer and Information Technology,
Chalmers University of Technology

Examensarbete

Sammanfattning

Examensarbetet strävade efter att sammanstråla koordinatsystemen hos en augmented reality-redo plattform och ett kontrollsystem för förarlösa fordon. Detta för att möjliggöra nya AR-funktionaliteter hos kontrollsystemet. Projektet som utförts vid Kollmorgen Automation AB, i Mölndal,undersökte möjligheten att skapa en plattform för AR-applikationer genom att dela världskoordinater hos två olika system. Arbetet som bedrevs var dels teoretisk (för att utveckla en lämplig metod), dels praktisk (för att kunna skapa en testbar sammanstrålnings-prototyp). Den utvecklade prototypen har avgränsat sig till orientering av en Google Tango-kompatibel surfplatta, kommunikation med ett kontrollsystem och visualisering via spelmotorn Unity. Det skapade systemet för justering av koordinatsystem har visualiserats för att visa att en adekvat koordinatsamordning uppnåtts. Den metod som bäst mötte de ställda kravspecifikationer var manuell orientering av tableten i den verkliga världen. Metoden användes för att sätta surfplattans koordinater till de som korresponderar i den karta som används av NDC8. Den prototyp som utvecklats visar att det finns utrymme för vidareutvckling av systemet men även att det är möjligt att skapa "augmented reality"-visualiseringar av NDC8 som rättar sig efter den verkliga världen.

Nyckelord: AR, AGV, augmented reality, förarlösa fordon, simulering, android, koordinatsystem, unity, google tango, lokalisering, kontrollsystem.

Preface

The authors believe that automation and augmented reality are two technologies that will only become more prevalent as time goes by, and as such the opportunity to merge these fields were of huge interest. To work on a project that could potentially lead to further advancements in these fields is both a privilege and a pleasure.

The authors would like to thank Sakib Sistek for the support and encouragement throughout the thesis. Kollmorgen Automation for the opportunity, and Fredrik Ludvigsson for all his time and knowledge.

Jeffrey Spång and Ingrid Brinkenberg, Gothenburg, June 2017

Contents

1
Introduction

1.1 Background

Kollmorgen is a world-leading provider of vehicle automation kits that combine a complete range of hardware, software and navigation technologies for automated guided vehicles (AGVs). Such an automation kit is the NDC8: a generic and scalable system that suits all types of AGVs regardless of size or complexity.

The NDC8 system is continuously improved and further expanded upon; the emergence of augmented reality computing platforms opens up for a wide new range of applications for the AGV control solution. Kollmorgen believes that the properties of Google Tango may provide new functionality to the NDC8 platform.

Google Tango technology gives a mobile device the ability to navigate the physical world in a similar manner as humans do. It brings a new kind of spatial perception to the Android device platform by adding advanced computer vision, image processing and specific vision sensors.

1.2 Purpose

As a step to innovate their product Kollmorgen would like to investigate the possibilities of implementing augmented reality technology to the NDC8 platform. A prerequisite for such applications is that the world views (coordinate reference frames) of the two systems are aligned.

1.3 Aim

In accordance with the wishes of Kollmorgen, this thesis aims to develop a solution that synchronises the coordinate system of the NDC8 with that of a Google Tango enabled device's.

The goal is to create a prototype application for a Google Tango enabled android device that uses shared objects such as walls or vehicles in both systems to illustrate that the synchronisation of two coordinate systems have been aligned.

1.4 Question formulations

The thesis will aspire to answer the following questions:

- How can two separate coordinate systems be aligned?
- How can the accuracy of the synchronisation be tested?
- How can the synchronisation of two coordinate systems be shown suitable for AR-application?

1.5 Delimitation

In this thesis only software development will be utilised in order to run test cases and to achieve communication between the Google Tango and the NDC8 platform. The software and hardware used is limited to preexisting systems provided and suggested by Kollmorgen.

Development platforms are restricted to Android and Windows using Unity and Visual Studio with C#.

1.5.1 Prototype specification

The prototype must have the functionality to adjust and change the position of virtual objects to align them with the real world without causing any alterations to the coordinate of said objects. For this thesis, alignment will refer to the act of adjusting the augmented world to match up both visually and logically with the real world and synchronisation refers to maintaining that alignment in run-time. Multiple approaches to synchronising/aligning two different coordinate systems will be considered, however only one will be implemented and tested. The solution that will be developed will be that which best meets the following criterion:

- The alignment method will utilise inherent Google Tango technology.
- The solution must be able to be developed within the given time frame of the thesis.
- The solution must be able to be adequately tested in respect to visualisation of environment data such as walls.
- The solution should utilise the game engine Unity.
- The application should provide a platform for further AR-functionality to the NDC8 system

User interfaces will only be developed to give the user the functionality needed to showcase the platform with little to no focus on design or ease of use.

2 Methods

The entirety of the thesis will take place on site at Kollmorgen. The work process is divided into five phases (see fig. 2.1): Sourcing, Development, Testing & Reviews, Adjustments, and Documentation.

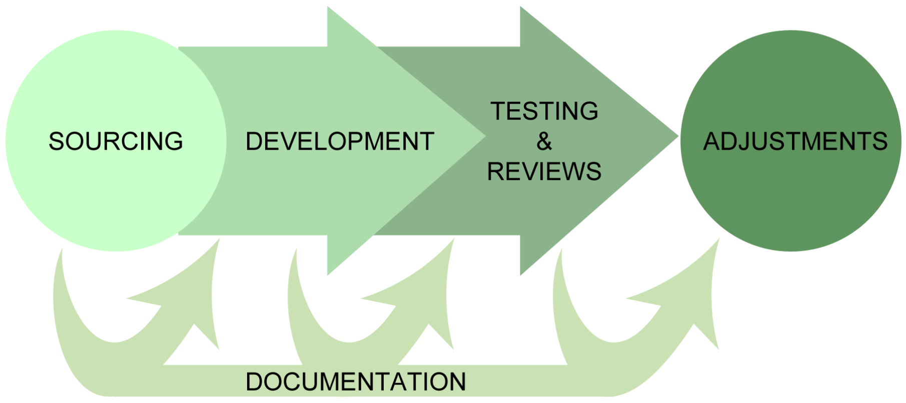


Figure 2.1: The intended work flow of the thesis (see fig. B.1 for a complete Gantt schedule).

During the sourcing phase information needed to develop the alignment system will be gathered and assessed with regard to how they can be integrated into a augmented reality computing platform.

The development phase aims to create a platform that enables augmented reality applications to be developed for the NDC8 system. The testing & reviews phase takes place during the second half of the development phase and consists of additional iterative development with the intention to define and acquire the functionality that is useful for the platform.

Documentation, including report writing, will be a continuous process from start to finish of the project.

In the final phase of the thesis: adjustments, the prototype and the accompanying thesis report will be improved upon in consideration with discoveries from previous phases.

2.1 Predicted tasks

The thesis gives rise to a number of initial tasks that need to be addressed:

Firstly, the localisation of the Google Tango enabled device must be initialised at a pre-determined and fixed position. This localisation will be based on the inherent map of an area that the NDC8 platform utilises. Once localisation has been achieved the coordinate system of the Tango will be linked to the corresponding coordinates of the NDC8 system.

Secondly, localisation of the Google Tango will be done by utilising certain markers in the real world (for example a wall or doorway) which will be used to align a virtual object in order for proper positioning to be reached. This will make the development kind of controller for alignment a necessity.

Furthermore, in order to prove that an acceptable synchronisation of coordinate systems has been achieved there needs to be a set of tests or some kind of visualisation that can show the final result.

2.2 Development model

The thesis is part software development to visualise a method of synchronisation and part research on how to align two coordinate systems; to reach the set goal the agile model will be used. This decision was made to handle changes in demand as efficiently as possible, and as the authors were meant to be part of a team at Kollmorgen, where scrum is the software development approach being used.

2.3 Tools

A Google Tango enabled tablet (see fig A.1) provided by Kollmorgen will be used to test and validate the solutions developed. The game engine Unity will be used to perform visualisation tests after the synchronisation has been made. Visual Studio 2015 will be used for software development. A complete NDC8 system will be provided by Kollmorgen, in addition to a simulation environment of the system. Physical AGV:s will also be at disposal if needed.

2.4 Testing and verification

As the prototype is created to illustrate a proof of concept for coordinate synchronisation only the accuracy of the application will be tested. The test of the alignment solution will be based on how well virtual objects are visualised in comparison to the real world. In the accuracy tests the alignment of the virtual floor plan overlaying their physical counterparts will be examined. The overlay will be measured, and an offset within a few centimetres will be considered acceptable.

3
Theory

3.1 Augmented reality with spatial awareness

The application addressed in this thesis visualises coordinate synchronisation through augmented reality. Kipper [1] describes augmented reality (AR) as a technology where real-world and digital information are blended. Furthermore, the developed application fulfils all attributes, introduced by Azuma [2], that are commonly accepted to define an AR-application. These decree that an augmented reality system:

1. must combine real and virtual information,
2. is interactive in real time, and
3. operates and is used in a 3D environment.

A requirement of creating a functional augmented reality application is accurate virtual representation of the real world and by extension how virtual data interacts with it. This is because an augmented reality experience is not believable if the real world data and virtual data are not properly aligned. For this to be possible all objects need to share the same coordinate system (further explained in 3.2) [1].

3.1.1 Enabling an AGV system with AR-functionality

One type of virtual information that can be combined with real world information is data from an automated guided vehicle (AGV). An AGV is a machine that moves and navigates through either following paths in the floor, usually set out by markers or wires, or through vision via lasers or magnets [3].

As noted by Ullrich [4] automated guided vehicles are nowadays widely used in industry and production related areas, primarily as a means of transporting goods and materials. To automate typical manual tasks AGVs need: 1) systems that govern navigation, 2) hardware to support the intended tasks of the AGV, and 3) software to run the AGV. A system, which bundles all these functions into one kit is the NDC8 control platform [5].

There currently is no connection between an augmented reality computing platform and the NDC8 control system so, as previously mentioned, correct coordinate alignment needs to be solved. Since the logic of an AGV is handled by control systems the coupling of AR-technology to an AGV should be through said systems to get access to necessary data, such as AGV position. The NDC8 platform utilises layout maps, similar to a buildings floor plan, through which the position of the AGV in the real world at a given point in time can be registered, and used to synchronise with an AR-device [6].

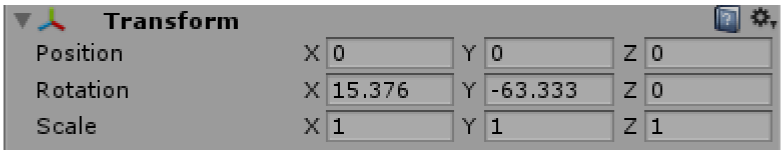
3.1.2 3D visualisation using Unity

To visualise the positional data and the map of the NDC8 platform in augmented reality they need to be transformed into 3D data. A way to interpret raw data into a format that can be used for this purpose is to utilise a game engine, to create and manage 3D objects, one such system is Unity.

At docs.unity3d.com [7] Unity is described as a cross-platform component-based game engine with support for C# scripting, which has two concepts of hierarchy. The first concept is inheritance, as commonly seen in object-oriented programming languages such as C#. The second concept is parenting: where every instantiated object in the game engine can have a parent as well as any number of children.

Two other important concepts are *Components* and *GameObjects*: Components are described as "the nuts and bolts of objects and behaviours in a game". A component can be anything from a 3D model to a C script. Components are gathered in containers which are GameObjects. GameObjects do not have any behaviour of their own and is only defined by its attached components. For example a GameObject representing an AGV would need at least one 3D model component and one script component that handles its behaviour [8].

In addition, all objects in Unity have a *Transform* component which contains a position, rotation and scale in the 3D-space (see fig. 3.1). The parenting concept enables the programmer to apply changes in the Transform class to every child object with a parent by only accessing the parent object [7].



Position	X	0	Y	0	Z	0
Rotation	X	15.376	Y	-63.333	Z	0
Scale	X	1	Y	1	Z	1

Figure 3.1: The Transform component as used in Unity.

For the purpose of this thesis a game object called "alignment plane" has been created, and it refers to a GameObject in Unity that is a parent to every other visualised object in the game world. This plane enables the developer to transform and adjust the position of the AR-experience so that it matches up with the real world. All object adjustments are made in accordance with a left-hand coordinate system, which is used in Unity (as seen in fig. 3.2).

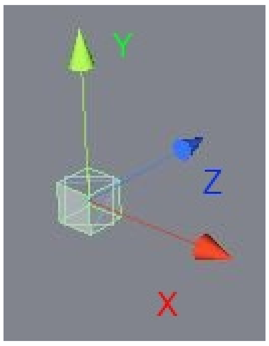


Figure 3.2: Orientation of a left-handed coordinate system. Image by Unity [9].

The game engine itself is based on a game loop that allows every active object to carry out any orders given by the program through an update method that is run every frame. The update loop runs as many times as the performance ,of the running device, allows per second. Therefor the developer has to make as much logic as possible depend on the time passed since last frame, this allows the application to become frame independent [10], [11].

When creating an application that uses Unity all objects and accompanying components are organised within container structures called scenes. An application may run entirely within one scene, or multiple scenes may be used together in a similar way as different levels of a game [12].

3.1.3 Google Tango device

While 3D representation of data is necessary for an augmented reality application, the choice of platform to visualise the 3D objects is equally important. A platform that can be used is Google Tango: an augmented reality platform for android devices with a certain set of sensors required for an augmented reality experience. These sensors include a depth sensing camera and additional motion sensors (fully shown in figure A.1) [13].

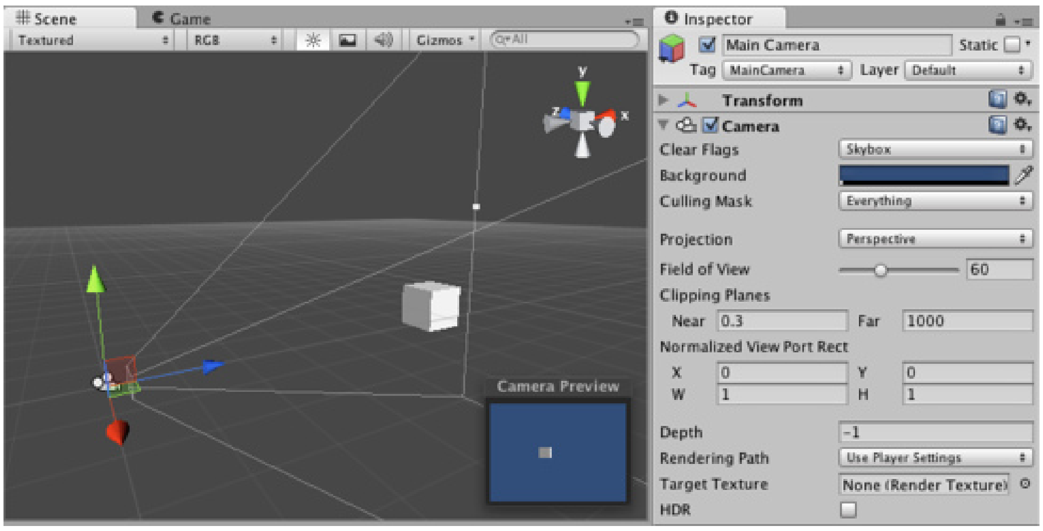
The sensors enables the unit to register details about its environment and its position in that environment. The position data, which is saved every time the unit has a new position, is available to the developer as a data structure that can be simplified as a struct PoseData (Listing 3.1). In the struct a quaternion orientation defines the rotation of the device and a 3D vector translation defines the position relative to the starting position of the Tango service, i.e. the starting position and rotation will always be 0 on all axes. [14].

```
struct PoseData {
    double orientation[4];
    double translation[3];
}
```

Listing 3.1: Code snippet of how the data of a tango pose is represented.

3.1.4 Unity with Google Tango

Unity has support for augmented reality via Google Tango, this enables the developer to use a tablets spatial awareness and depth sensing abilities in Unity applications. In this thesis the tablet is used as an augmented reality camera for the computation platform. The *Camera* component (shown in fig. 3.3) in Unity is a device through which the game world is rendered. This means that physically moving or rotating the tablet also moves the game engine camera in the same manner [15], [16].



```
Main Camera                     Static
Tag  MainCamera     Layer  Default
Transform
Camera
Clear Flags            Skybox
Background
Culling Mask           Everything
Projection             Perspective
Field of View          60
Clipping Planes
  Near  0.3            Far  1000
Normalized View Port Rect
  X  0                 Y  0
  W  1                 H  1
Depth                  -1
Rendering Path         Use Player Settings
Target Texture         None (Render Texture)
HDR
```

Figure 3.3: A Unity GameObject with a Camera component attached. The camera has a customisable view port which can be seen as the white lines drawn outwards from the camera. The camera preview in the bottom right shows what the camera will see in run-time. Picture by Unity [17].

3.1.4.1 Tango platform related challenges

To correctly combine real and virtual information with the Tango platform there are two major challenges that need to be solved. These challenges involve motion tracking of the device running the application and the fidelity of the depth camera on the Tango device.

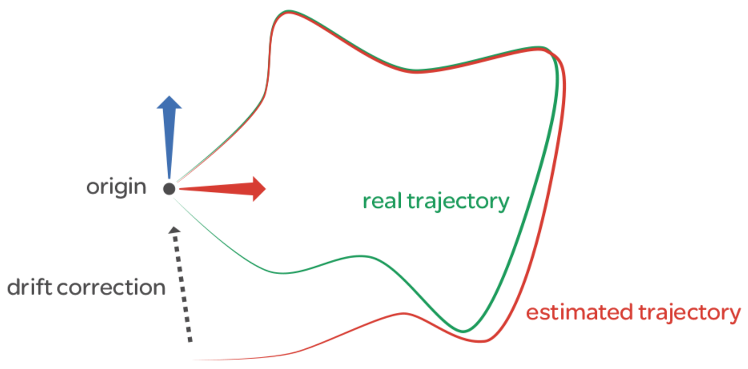


Figure 3.4: Given a starting point the red line illustrates an estimated trajectory of the Tango enabled device whereas the green line shows the real trajectory at run time caused by drifting. Picture by Google [18].

Current Google Tango enabled devices all suffer from an error within its motion tracking called drifting (see fig. 3.4). Drifting is a small measurement error that occur during runtime of a Tango application which during it's lifetime builds up and produces a discrepancy between position in the virtual world versus the position in the real world [19].

The fidelity of the depth camera further complicates this problem since it is only reliable at short range and even then might have measurement errors, which makes it harder to, through the depth camera, use the environment to compensate for drifting [20].

3.2 Methods for aligning the Unity game world with the real world

All 3D objects, physical or virtual, have their own position and orientation in the space they occupy. Alignment refers to the adjustment of these relative positions and orientation of one or more objects [14], [21].

For this thesis the case of coordinate alignment is considered the same. If one system with certain coordinates has the corresponding coordinate points in another system then they are seen as aligned (herein also referred to as synchronised). There are numerous ways of achieving coordinate synchronisation, the most common ones are described in the following sections.

3.2.1 Manual alignment

In this report manual alignment refers to visual comparison of a physical object in the real world with its virtual counterpart, without the help of built in alignment-algorithms of a computing platform. Like Viola and Wells III [22] the use of an object model with a known pose (coordinate transformation) to predict an image, which then is compared to a real image. Manual alignment is facilitated by having an object, such as a strip of tape, at a pre-measured position in the real world and a virtual object of the same shape at the corresponding position in the Unity game world. This can then be used as an alignment helper by adjusting the alignment-plane by hand so that its virtual marker overlays the real object.

3.2.2 Automatic alignment and synchronisation

3.2.2.1 Plane fitting

In contrast to manual alignment one easily implemented automatic alignment method is, as described by Hulik, Spanel, Smrz *et al.* [23], adjusting the y-position of the alignment-plane to a plane identified as the floor by the Tango unit. The floor can be defined as the plane with the lowest y-position among all planes with a y-position lower than the tango unit itself.

Similar to floor plane fitting the Tango device can also identify intersections of walls, i.e corners. This can be used to match unique corners of the Unity world and real world to each other and adjust the alignment-plane so that both corners have the same position relative to the Tango unit [23].

3.2.2.2 Area Learning

As drifitng is an issue with the Google Tango platform, any alignment may suffer from a lack of accuracy by this phenomenon. A more complex solution to correct errors from motion drifting is area learning. By remembering visual features of the environment a Tango enabled device can self-correct from drifting due to being able to notice when a key feature that it has already seen at one position has a new position adjacent to its previous position. This means that the device has probably drifted slightly and that the drift can be mitigated by adjusting the device position so that the old and new position of the key feature matches up. Visual features that are commonly used for area learning are corners, edges and any visually unique object in an environment [18].

3.2.3 Aligning with external tools

There are several other tools that can assist in alignment that depend on components or devices not necessarily accessible for every Tango enabled device or developer. Devices that have built-in magnetometers can use the magnetic rotation compared to magnetic north to rotate the game world so that virtual north always corresponds to the direction magnetic north [24].

Another way to achieve less drifting and need for alignment could be to make use of external sensors such as laser sensors or more accurate motion tracking devices to keep track of a users position instead of using the built-in motion tracking of the Tango platform. This would also remove the need for Tango-enabled devices and only require the device running the application to have a gyroscope to handle device rotation so that the user can change the direction that the camera is facing [25], [26].

4 Implementation

4.1 System overview

The developed prototype used to align the world views of the Unity 3D world space and the NDC8 system is based on manual alignment. In order to achieve a prototype that allowed synchronisation between the systems, communication between them had to be addressed. Another issue that had to be taken into consideration was how to visualise the alignment.

These problems were addressed by creating a three-part communication system consisting of a Unity application, a data parser and the NDC8 system (illustrated in figure 4.1). The part that parses NDC8 data, called a passthrough application, is a Windows application and communicates with the Unity application over TCP/IP from a separate computer. Map and vehicle data from the NDC8 platform is formatted by the passthrough application and then transmitted to the Unity application in the Google Tango device where it can be visualised in Augmented Reality.

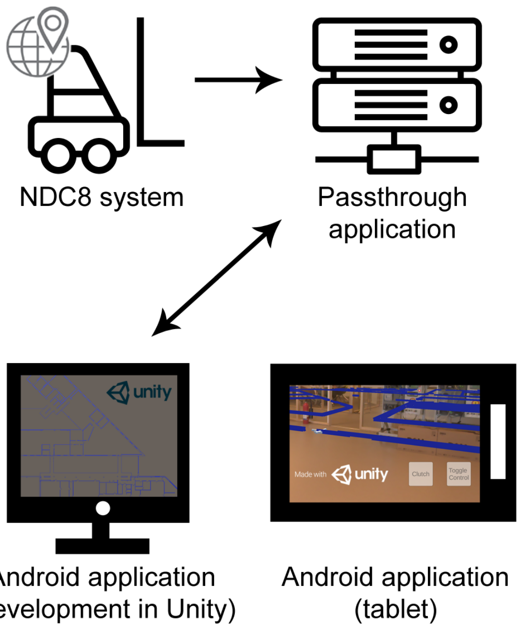

NDC8 system
Passthrough application
Android application (development in Unity)
Android application (tablet)

Figure 4.1: Communication between the three different systems was achieved via creation of a passthrough server, allowing AR-visualisation on a Google Tango device. Graphics elements from Freepik [27], [28] and Flaticon [29], [30].

4.1.1 Passthrough application and data flow

The passthrough unit was made to serve both as a formatter and a server in order to receive and transform the control system coordinates to be Unity compatible. In the NDC8 system the measuring unit is millimetres while Unity uses meters. Since the ground plane in Unity is on the XZ-axis we also needed to transform the 2D Y-coordinate to correspond to the 3D Z-coordinate. Otherwise our application would visualise data from the floor to the roof instead of across the room.

The passthrough application requests data from the NDC8 system through a Kollmorgen developed API and relays it to the Unity application in real-time. Such behaviour was critical since that data consists of vehicle information, like position on map, that needs to be updated as soon as possible. This so that the visualisation does not lag behind more than necessary compared to the real position of the vehicle.

In addition to time dependent data, persistent data such as maps are not continuously transmitted but can be sent upon request from the server terminal. The Unity application only listens for persistent data if it does not already have access to it locally; all data sent to the Unity application is recorded and saved on the running device. The decision to save persistent data locally on the device was to allow the application to function in an offline setting if it has already been connected to a saved map once.

4.1.2 Visualisation of data

Map data consists of two 2D points representing the start- and end-position of a wall. The visualisation of this is done by using the Unity LineRenderer component which can take multiple points and render a continuous line between them.

Vehicle data is represented by using a 3D model (see fig. 4.2) attached to an Unity game object, which allows the rendered position of the 3D model to be manipulated.

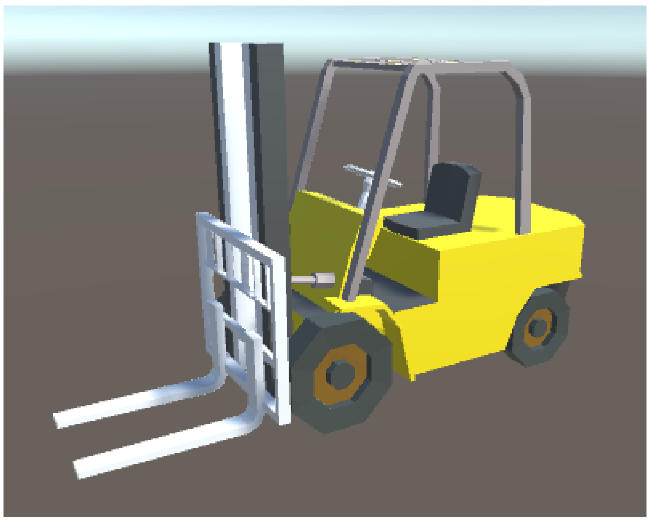


Figure 4.2: 3D-model used for vehicles.

Since vehicle data from the NDC8-system is only updated when the vehicle reaches a new segment on the map there was a need to use linear interpolation between positions to make visualised 3D models move smoothly. Because of this behaviour the visualisation of vehicles is always done with one segment delay compared to the NDC8 position. When the Unity application gets an updated vehicle position from the NDC8 system it transforms the game object's position a small bit each frame until it has reached the new position. The linear interpolation was done with regards to the time elapsed until a new position is received. Essentially, this made it so that the 3D models of vehicles appeared to have a smooth motion even though the data received does not.

4.1.3 Unity Application structure

The application was developed with two flows in mind: the first allowing an alignment scene with previously set tablet coordinates to be loaded, and one option where the alignment scene loads with manually set positional coordinates. To achieve the sought functionality four scenes were included in the prototype: start (fig.4.3), top down view (fig. 4.4), layout manager (fig. 4.6a), and alignment (fig. 4.6b).

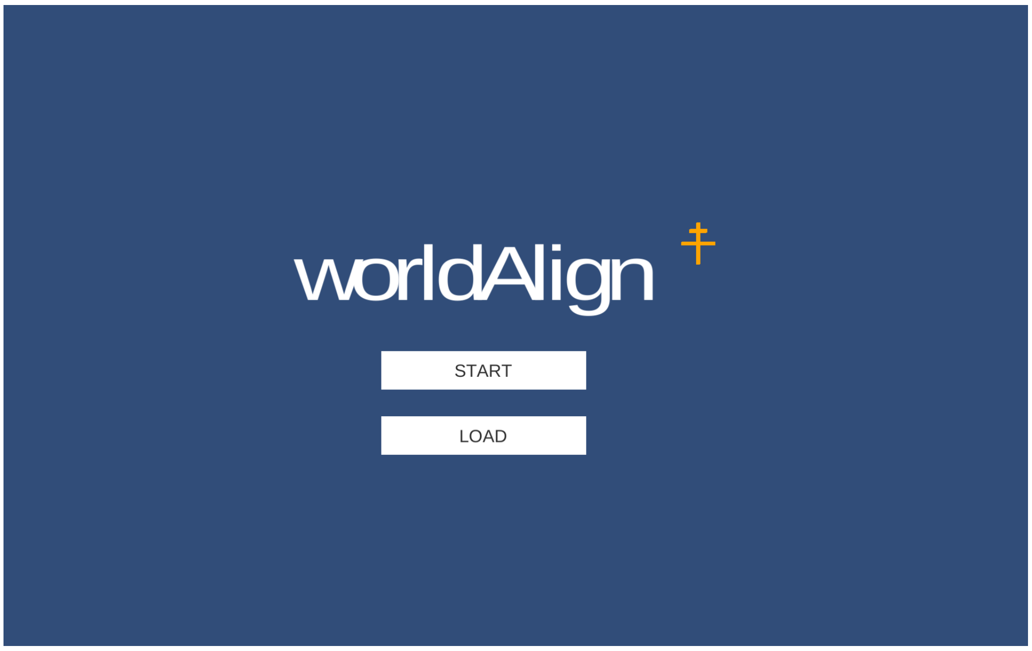
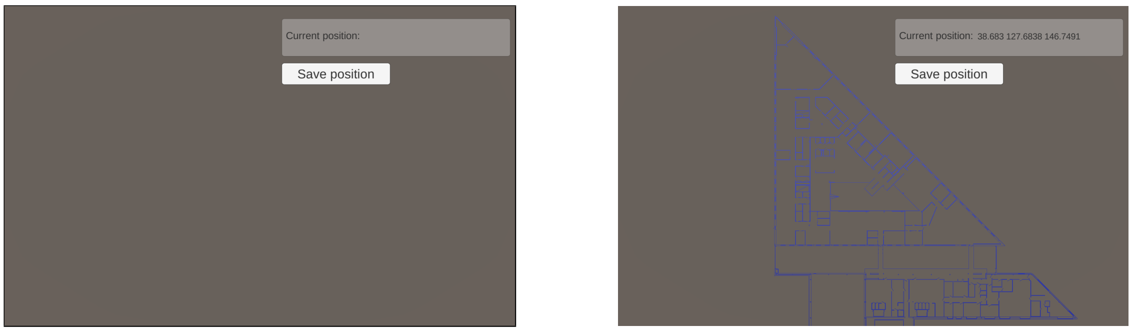


Figure 4.3: Start scene of the application. From here two interactive buttons allows the user to choose between starting a new manual alignment or loading a previously saved alignment.



(a) Scene without passthrough data. **(b)** Scene with received map data.

Figure 4.4: Top down view of the application.

Image 4.4a shows the scene visible after a user has pressed start in the starting scene, which contains an empty canvas, a panel with text and an interactive button. In this mode the application will wait for floor plan data (a layout) from the passthrough server, and once the data has been received the layout will be rendered (as seen in fig. 4.4b).

Once the layout has been rendered the user can choose their current position in the layout, by pressing once on said position. This will cause a cross-shaped alignment object to appear on the pressed position (see 4.5) and the coordinates will be registered and shown in the upper right corner. If the position is satisfactory to the user he/she can choose to press the "save position"-button, which will save the position and name of the layout as an XML-file in the format shown in listing 4.1, prepending new entries in the XML-file. The data will be saved locally on the Android device and the layout (to be used in aligning the plane to physical objects) will be set as the current layout in the XML-file.

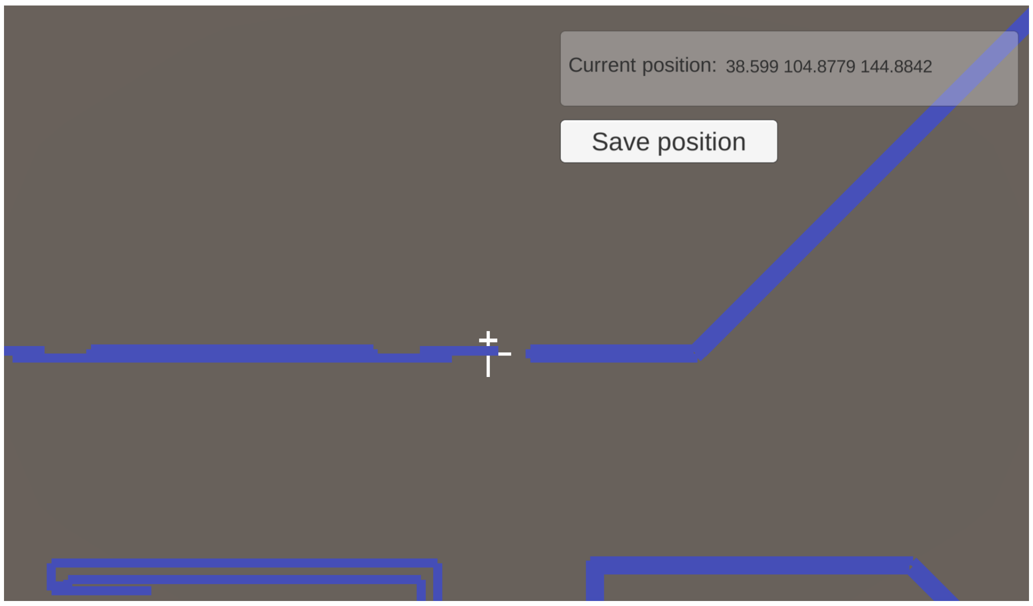


Figure 4.5: Placement of alignment object to set position.

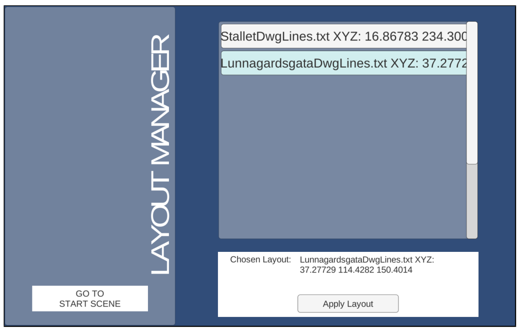
```
<Layouts>
    <Layout>
        <Name>LayoutName</Name>
        <Position>X Y Z</Position>
    </Layout>
    <CurrentLayout>
        <Name>LayoutName</Name>
    </CurrentLayout>
</Layouts>
```

Listing 4.1: Layouts are saved according to this specific format, and the current layout is also registered

Following saving of the layout the scene will change to the manual alignment scene (fig. 4.6b). The alignment scene utilises the same layout and alignment cross as the top down view but allows the viewer to see the objects in 3D, overlaying the real world. From this scene additional manual adjustments can be performed on the alignment plane (further discussed in section 4.2).

The other way to initialise the manual alignment is via the layout manager (fig. 4.6a). This scene dynamically creates interactive buttons corresponding to "Layout"-nodes in the saved XML-file. Both name and position coordinates are pulled from the save file and shown as button content. However, if there is no save file, then no buttons are created.

On the other hand, if there is a layout saved, after choosing a specified layout that layout will be rewritten as the current layout in the XML-file, and the corresponding coordinates will be set as the tablet's initialisation coordinates (the importance of initialisation is discussed in section 4.2.1).

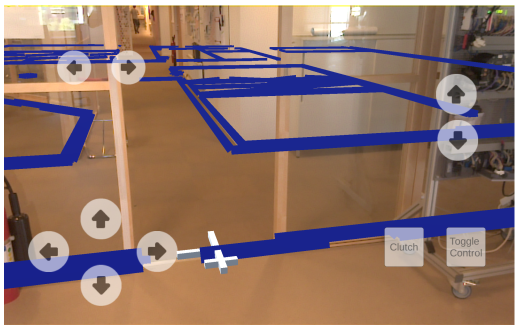


StalletDwgLines.txt XYZ: 16.86783 234.300

LunnagardsgataDwgLines.txt XYZ: 37.2772

Chosen Layout: LunnagardsgataDwgLines.txt XYZ: 37.27729 114.4282 150.4014

(a) Layout manager scene.



(b) Alignment scene.

Figure 4.6: a) Layout manager scene, used to initialise the Google Tango device with a specific layout and position. b) Alignment scene where manual adjustments are possible.

4.2 Manual alignment

Manual alignment is dependent on two concepts: initialisation and adjustment of a virtual map onto the real world. This section explains the reasoning behind and practical implementation of these ideas.

4.2.1 Initialisation

The flow of the application was deemed necessary to allow the user control of where to initialise the position of the Google Tango enabled tablet. As the Google Tango initiates at position (0, 0, 0) in the Unity world coordinates the alignment plane had to be shifted to achieve the desired starting position. As such the initialisation was the first hurdle to overcome to align the two coordinate systems.

The choice of initialisation method was determined by evaluating different kinds of methods and approaches to localisation. While the most convenient solution would be to utilise automatic alignment such as plane fitting, the hardware in the tablet used during the development of this solution did not make this feasible. This due to the fact that the range and accuracy of the built-in sensors was not good enough to eliminate the need for manual alignment, even after automatically fitting the alignment plane to the floor.

Neither was it found possible to utilise the built-in compass to rotate the alignment plane due to not being able to access it through the Android API. Instead the authors came to the decision to manually set the initialisation point, based on the floor plan of a building, where it is possible to deduct the real position of the tablet in comparison to the map given.

4.2.2 Plane adjustment

Using the parenting concept in Unity two planes were created, on which the virtual world is placed upon. The two planes were created with different hierarchy in respect to the camera of the tablet, in order to obtain properties to alleviate the manual alignment. One plane, used for positional alignment, is not a parent of the camera, whereas the other plane used for rotational alignment is a parent of the camera. The rotational plane was added so that the alignment plane could be rotated without affecting the relative position of the camera compared to the alignment plane.

When initialising the position of the tablet it is likely that the coordinates aren't in full synchronisation with the real world with the virtual world being offset into either the x-, y- or z-axis respectively. Because of this a function to decouple the plane from the camera, which allows the user to freely move and rotate the tablet to better align the virtual objects, has been developed. This is accessible to the user via the "Clutch"-button (see fig. 4.7); to couple the plane to the camera again the user have to push the button once more.

In addition to the clutch control the user can also access buttons to move the alignment plane to better overlay the real world counterparts. These buttons are accessible via the "Toggle Control"-button located next to the clutch button. Pressing the button activates eight buttons that move the plane in different directions (all visible in fig. 4.7).

In the upper left corner of the alignment scene two buttons, that control the alignment plane in Z-axis, are placed. In the upper right corner the controls for moving the alignment plane in Y-axis are placed. Finally in the lower left corner to move the plane in Y and X axis are found. Using all of the described buttons a user can shift the virtual plane to fully overlay the real world, and achieve coordinate alignment.

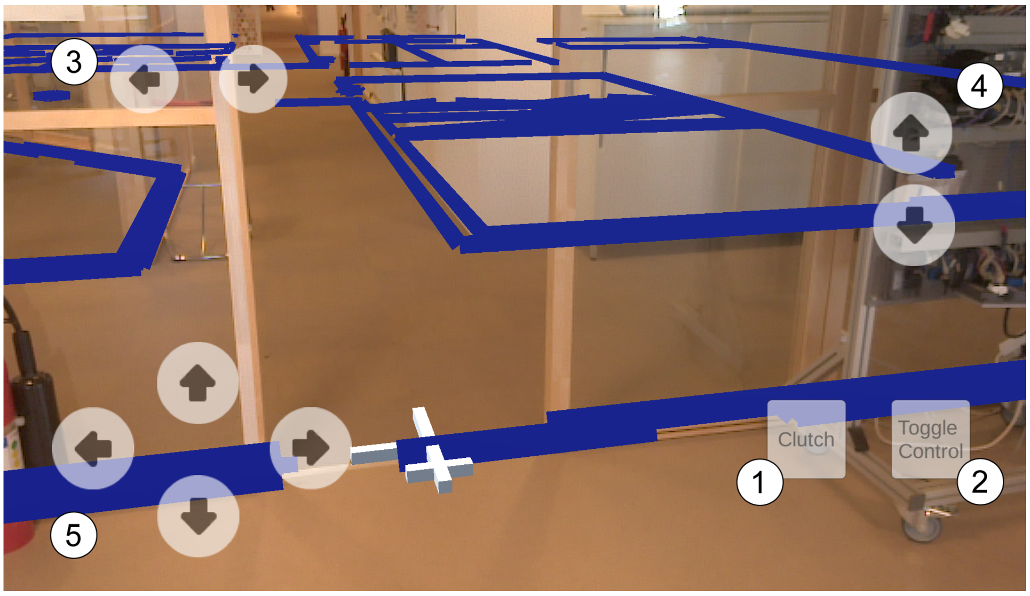


Figure 4.7: Alignment scene with controls to move the plane for positional alignment. Number denotations: 1) Clutch-button. 2) Toggle to show/hide control buttons. 3) Z-axis plane adjustment buttons. 4) Y-axis plane adjustment buttons. 5) Y- and X-axis plane adjustment buttons.

4.3 Test application

A movement tracking application was constructed that visualises the movement of the Google Tango tablet as a pink trail and overlays that movement on a floor plan of the location were the test was done (see fig. 4.8). In addition, the application prints out the distance from the starting position as a 3D-vector as the first three numbers under "Camera position". The fourth number is the average movement speed that the tablet had during the test. This was added in case that the movement speed had any effect on the motion tracking.

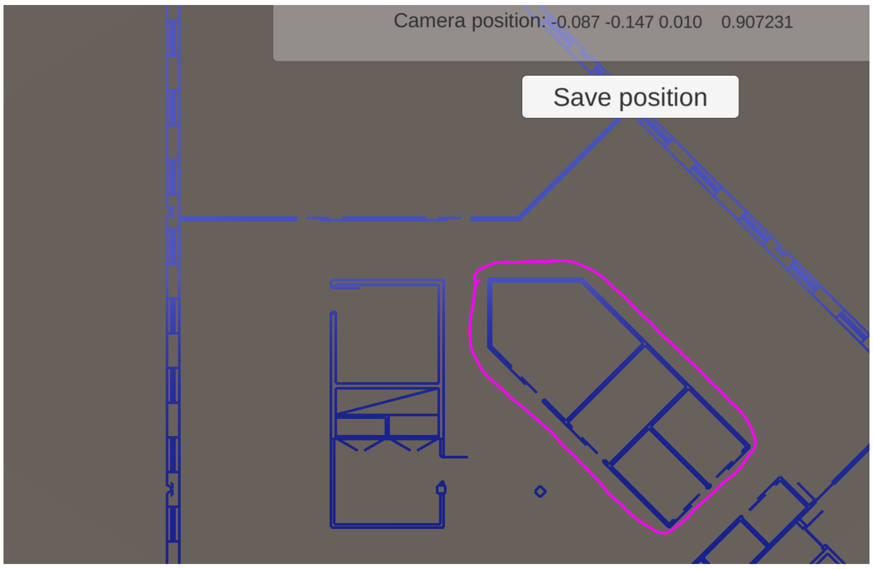


Figure 4.8: Screenshot of test application. The blue lines are walls and the pink path is the movement of the device running the application

4.3.1 Measurement of drifting error

To correctly measure the error from drifting with the test application a test was constructed. The test consists of walking in a closed loop and measure any differences of the Google Tango tablet's position with regards to its starting position. When the loop is closed its position then should be the same as the starting position. The test starts when the tablet is moved from the starting position and ends when it has returned to the starting position in the real world. The measured error is the distance between the starting position vector (0, 0, 0) and the camera position vector when the device is returned to its starting position.

5
Results

5.1 Coordinate synchronisation

The solution developed is based on manual alignment of a virtual world with the real world. The alignment in itself is done in a Unity application using map data sent to the application from the NDC8 control system via a passthrough application that parses the data.

In conjunction with the set specifications and delimitation the prototype created uses virtual 3D representations of a 2D map to align with their real world counterparts (in this case walls). The virtual representations are set in a Unity game world and located in planes that the user can manipulate in three axes to adjust the alignment.

Initial localisation of the device running the Unity application is integral to the prototype as its starting position is always set to (0, 0, 0) on application start up. Two modes to set a proper starting position the Unity application was created. One mode which receives data from the passthrough application and allows the user to choose a position on a map. The other a "load" mode which loads an already existing starting position, removing the need for initial alignment.

After setting the starting position alignment of the 3D map with the real world is possible. The developed manual alignment method consists of two kinds of controls. One through buttons in the user-interface that moves the alignment plane in a given direction when pressed. The other via the "Clutch"-mode from the Google Tango API, which utilises rotational positioning

Communication between the passthrough application, running on a Windows desktop, and the Unity application, running on an Android tablet is made possible through TCP/IP sockets and works satisfactory over a wireless local area network. Communication over a Wide Area Network has not been tested.

The developed prototype is dynamic and the Unity application only renders data sent from the passthrough application. This makes the solution modular with regards to using it together with systems other than NDC8.

5.2 Testing

Two types of tests were done during this thesis where one was based on the testing application described in the Implementation chapter and the other was done during development and by using the application.
The latter consisted of making sure that the virtual world was properly aligning with the real world and that one virtual meter was the same as one meter in reality. This was done by simply aligning virtual walls to the real world and walking around them to see if there were any major discrepancies more sizeable than the errors that can be caused by faulty measure. No such discrepancies were noticed, thus the application was considered to be correctly scaled.

After performing a series of tests with the testing application in the test environment it was concluded that after walking at an average pace of 1 m/s for about 30 m the discrepancy between the starting and ending position was 26 cm at average with the lowest measured being 15 cm and the highest being 42.5 cm. See fig. C.1 for all measurements.

One event which was revealed during these tests was the complete loss of positional tracking (see fig. 5.1). This resulted in a broken AR-experience where the alignment needs to be reset in order for the experience to be satisfying.

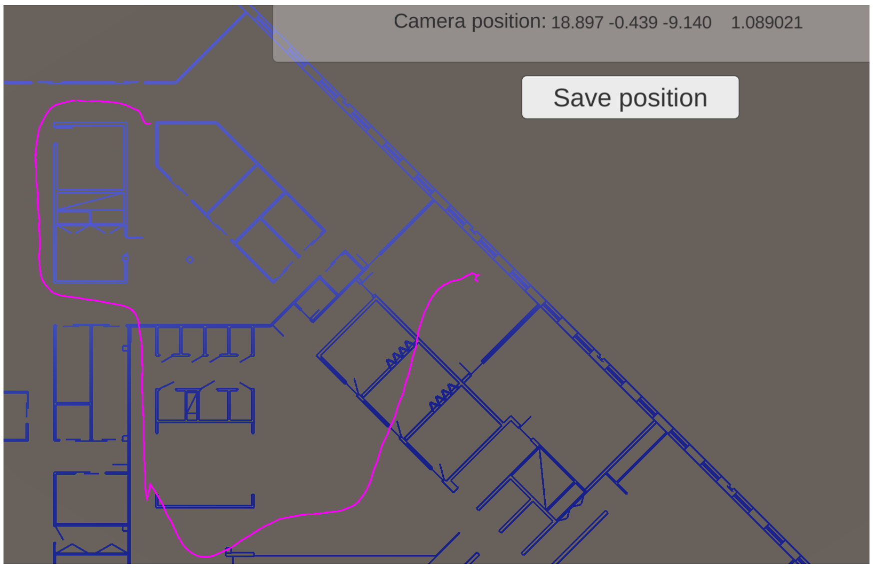


Figure 5.1: Pink path is the movement that the tango device estimated it had taken when in reality it moved in a closed loop. Exact moment when loss of tracking occured can be seen in the bottom left part of the image

6 Conclusion

6.1 Resume

A method for synchronising the coordinate systems of a Google Tango enabled tablet and Kollmorgen's NDC8 control system has been developed. As the biggest limitation for creating a method to align these systems lies in the hardware of the tablet the final prototype relies on manual adjustments of a plane that holds the game world in the Google Tango. To achieve coordinate synchronisation the plane is adjusted to overlay the real world using virtual representations of physical objects, through which a user can orient the plane.

To show that synchronisation between the coordinates perceived in the Google Tango and those derived from a map in the NDC8 system were shared, the alignment application was tested. The test aimed to illustrate the level of synchronisation accuracy, essentially if the game world of the Google Tango enabled tablet occupied the same coordinates as those used by the NDC8.

6.2 Discussion

At the beginning project it was thought that the decision of how to implement a synchronisation application would take a long time and much research. Partly because the authors believed that there would be numerous ways of implementing coordinate alignment using the Google Tango device. This assumption was proven somewhat incorrect as the authors realised that inherent sensors could not provide sought alignment functionality. In addition it was not deemed realistic to complete a prototype that utilised external sensors in the given time frame. This led the authors to decide to develop a manual alignment solution early on.

As a consequence the research part of this project was more directed towards the functions of the Google Tango platform, rather than discovering multiple ways of aligning its coordinate system with the AGV control system. Overall the planning of the project was not too far off in comparison to the execution. However, the development phase did take longer than anticipated and thus led to a much shorter testing and reviews phase than what was planned.

That being said the authors believe that the prototype discussed in the report does fulfil the aim of the thesis. The prototype shows that alignment is possible and that coordinate system synchronisation has been achieved. This because the visualised data is being represented in a position in the real world corresponding to the same position on the virtual map.

6.2.1 Strengths and weaknesses

While the developed solution was shown to achieve some level of synchronisation a few weaknesses of the application were found. The most glaring issue is that of the initialisation of Google Tango coordinates to move its coordinate system to the one used by the NDC8. By choosing manual initialisation there is the introduction of errors from two sources: 1) the user can only approximate the physical position of the tablet in the real world, 2) the coordinate initialisation in the application is mostly based on visual evaluation from the user if the coordinates in the NDC8 map are not previously known. Due to the errors, the initialisation may not provide a perfect alignment, at which point the user must adjust the game world to overlay the real world. This also introduces errors as the only way of combating overlay offset is if it is large enough to be viewed in the application. This does not give a satisfactory measurement of error, which also implies that the tests are not optimal either.

Additionally, the alignment in itself is not without fault. An automated solution would in theory be faster than the current prototype, if the identification of planes and markers was sufficiently accurate.

On the other hand one of the biggest strength of the prototype lies in its modularity; the entire system is very adaptable to changes. For example it is easy to replace the current passthrough application with a different solution, as long as it provides the right type of input required by the prototype. The passthrough currently only interprets data from teh NDC8 control system but can be further developed to format any 2D or 3D data. Since the prototype utilises Unity there is also the possibility of using the alignment in any augmented reality device that has support for the game engine. Furthermore, it implies that development on Google Tango's side would not necessarily negatively affect the prototype, it may, however, cause the need for a software patch to keep the application working with a newer Google Tango version.

6.2.2 Environmental aspects

When it comes to software development and its relation to environmental issues the direct connection is not always easy to see. For this project the developed prototype in itself may not have obvious environmental impact. However, it is a facilitator for future software to be developed that in turn could decrease the resource use regarding implementation of AGVs.

For example, using AR for virtual demonstrations of AGV behaviour instead of having physical demonstrations reduces the need for having any kind of hardware being produced in advance. By using virtual representations of AGVs one could identify eventual installation problems, or perhaps get a better picture of how numerous automated guided vehicles would act in a real production environment.
These suggestions could possibly not only reduce resources in the sense of time spent by application engineers planning how to install AGVs, but also lower the need for travels to location (thereby cutting emissions).

Seen from a developer's stand point the modularity of the created prototype could provide a longer longevity of the software, which is inline with sustainable software development. If the need to replace the platforms running the prototype application is lowered, there could also be a lesser need to utilise the finite resources used to created the devices.

6.3 Further development

The project gave rise to many problems that needed solving and numerous questions that have been to some capacity answered. However, new questions and problems also arose during the process of creating an alignment prototype.Some of the biggest issues that needs to be addressed with the prototype presented in this report is discussed in the following sections.

6.3.1 Motion correction algorithm

While a test has been developed to illustrate measured errors of the alignment additional tests to increase the statistic significance need to be performed. If after further testing a pattern in motion drift is found, it could be possible to use an algorithm to counter some of the drift, resulting in a reduced need to realign the game world after extended movement.

6.3.2 Hardware progress

With current hardware, the writers decided that fully manual alignment was the most convenient and functional method for alignment. This, however, is subject to change as hardware progresses and built-in sensors become more precise with longer range. This progress could justify a change from the current manual alignment method to a new automatic one.

6.3.3 Improved client-server data flow

To make the application more reliable and easier to use the current communications implementation would need to be revised. One such behavioural change could be to remove any need to use the server terminal to send data manually. This could be done by automatically have the Unity application ask if the server has access to any data that the application does not. This flow could also be simplified by removing any hard-coded IP and to instead use broadcast messages to find the IP-address of a tablet currently running the application.

6.3.4 Expand AR-functionality

Currently the application has no interaction with the real world other than the fact that it can overlay data to it. One addition that would make the AR experience more immersive is to implement ambient occlusion. Ambient occlusion is a technique that calculates how much light an object is getting and by having that in the application one could "hide" virtual objects behind real ones. This would make it so that the Unity Camera would only render virtual objects that are not blocked from view by real world objects.

Bibliography

[1] G. Kipper, “Chapter 1 – What Is Augmented Reality?”, in *Augmented Reality*, 2013, pp. 1–27, ISBN: 9781597497336. DOI: 10.1016/B978-1-59-749733-6.00001-2. [Online]. Available: http://www.sciencedirect.com.proxy.lib.chalmers.se/science/article/pii/B9781597497336000012 (visited on 29/03/2017).

[2] R. T. Azuma, “A Survey of Augmented Reality”, *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355–385, Aug. 1997, ISSN: 1054-7460. DOI: 10.1162/pres.1997.6.4.355. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/pres.1997.6.4.355 (visited on 18/04/2017).

[3] G. Ullrich, “The History of Automated Guided Vehicle Systems”, in *Automated Guided Vehicle Systems*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 1–14. DOI: 10.1007/978-3-662-44814-4{_}1. [Online]. Available: http://link.springer.com/10.1007/978-3-662-44814-4_1 (visited on 21/05/2017).

[4] ——, “Modern Areas of Application”, in *Automated Guided Vehicle Systems*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 15–96. DOI: 10.1007/978-3-662-44814-4{_}2. [Online]. Available: http://link.springer.com/10.1007/978-3-662-44814-4_2 (visited on 22/05/2017).

[5] ndcsolutions.com, *Proven platform for AGVs and mobile robots | NDC Solutions*, 2017. [Online]. Available: http://ndcsolutions.com/building-agvs/?gclid=Cj0KEQjwmIrJBRCRmJ_x7KDo-9oBEiQAuUPKMuE64hE8haRARd61ziaQNwweWJRXtzD4JwO3pqFxyi0aAgIv8P8HAQ#parts-ndc-solutions (visited on 22/05/2017).

[6] ——, *Bring out the best in your AGVs*. [Online]. Available: http://ndcsolutions.com/wp/wp-content/uploads/2017/01/35200-005J-NDC-Solutions.pdf (visited on 01/06/2017).

[7] docs.unity3d.com, *Unity - Manual: The Hierarchy window*, 2017. [Online]. Available: `https://docs.unity3d.com/Manual/Hierarchy.html` (visited on 25/04/2017).

[8] ——, *Unity - Manual: Using Components*. [Online]. Available: `https://docs.unity3d.com/Manual/UsingComponents.html` (visited on 02/06/2017).

[9] ——, *Unity - Manual: Cameras*, 2017. [Online]. Available: `https://docs.unity3d.com/560/Documentation/Manual/CamerasOverview.html` (visited on 21/04/2017).

[10] ——, *Unity - Manual: Execution Order of Event Functions*, 2017. [Online]. Available: `https://docs.unity3d.com/Manual/ExecutionOrder.html` (visited on 18/04/2017).

[11] ——, *Unity - Scripting API: Time.deltaTime*. [Online]. Available: `https://docs.unity3d.com/ScriptReference/Time-deltaTime.html` (visited on 02/06/2017).

[12] ——, *Unity - Manual: Scenes*, 2017. [Online]. Available: `https://docs.unity3d.com/Manual/CreatingScenes.html` (visited on 25/04/2017).

[13] developers.google.com, *Tango Tablet Development Kit User Guide | Tango | Google Developers*, 2017. [Online]. Available: `https://developers.google.com/tango/hardware/tablet` (visited on 20/04/2017).

[14] ——, *What Are Tango Poses? | Tango | Google Developers*, 2017. [Online]. Available: `https://developers.google.com/tango/overview/poses` (visited on 30/05/2017).

[15] ——, *Tango Developer Overview | Tango | Google Developers*, 2017. [Online]. Available: `https://developers.google.com/tango/developer-overview` (visited on 18/04/2017).

[16] docs.unity3d.com, *Unity - Scripting API: Camera*, 2017. [Online]. Available: `https://docs.unity3d.com/ScriptReference/Camera.html` (visited on 30/05/2017).

[17] ——, *Unity - Cameras*. [Online]. Available: `https://docs.unity3d.com/401/Documentation/Manual/Cameras.html` (visited on 02/06/2017).

[18] developers.google.com, *Area Learning | Tango | Google Developers*, 2017. [Online]. Available: `https://developers.google.com/tango/overview/area-learning` (visited on 30/05/2017).

[19] ——, *Motion Tracking | Tango | Google Developers*. [Online]. Available: `https://developers.google.com/tango/overview/motion-tracking` (visited on 30/05/2017).

[20] T. S. Kalyan, P. A. Zadeh, S. Staub-French and T. M. Froese, "Construction Quality Assessment Using 3D as-built Models Generated with Project Tango", *Procedia Engineering*, vol. 145, pp. 1416–1423, 2016, ISSN: 18777058. DOI: `10.1016/j.proeng.2016.04.178`. [Online]. Available: `http://linkinghub.elsevier.com/retrieve/pii/S1877705816301850` (visited on 24/04/2017).

[21] T. Atkins and M. Escudier, *A Dictionary of Mechanical Engineering.* Oxford University Press, Jan. 2013, ISBN: 9780199587438. DOI: `10.1093/acref/9780199587438.001.0001`. [Online]. Available: `http://www.oxfordreference.com/view/10.1093/acref/9780199587438.001.0001/acref-9780199587438` (visited on 22/05/2017).

[22] P. Viola and W. M. Wells III, "Alignment by Maximization of Mutual Information", *International Journal of Computer Vision*, vol. 24, no. 2, pp. 137–154, 1997, ISSN: 09205691. DOI: `10.1023/A:1007958904918`. [Online]. Available: `http://link.springer.com/10.1023/A:1007958904918` (visited on 18/05/2017).

[23] R. Hulik, M. Spanel, P. Smrz and Z. Materna, "Continuous plane detection in point-cloud data based on 3D Hough Transform", *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 86–97, 2014, ISSN: 1047-3203. DOI: `http://dx.doi.org/10.1016/j.jvcir.2013.04.001`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S104732031300062X` (visited on 15/06/2017).

[24] J. R. Blum, D. G. Greencorn and J. R. Cooperstock, "Smartphone Sensor Reliability for Augmented Reality Applications", in, Springer, Berlin, Heidelberg, 2013, pp. 127–138. DOI: `10.1007/978-3-642-40238-8{\_}11`. [Online]. Available: `http://link.springer.com/10.1007/978-3-642-40238-8_11` (visited on 02/06/2017).

[25] A. Pagani, J. Henriques and D. Stricker, "SENSORS FOR LOCATION-BASED AUGMENTED REALITY THE EXAMPLE OF

GALILEO AND EGNOS”, *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLI-B1, pp. 1173–1177, Jun. 2016, ISSN: 2194-9034. DOI: `10.5194/isprs-archives-XLI-B1-1173-2016`. [Online]. Available: `http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLI-B1/1173/2016/` (visited on 02/06/2017).

[26] HCL Technologies, *HCL Technologies Ltd Files Patent Application for Tracking Position and Orientation Using External Sensor for Augmented Reality Applications - ProQuest.* [Online]. Available: `http://search.proquest.com.proxy.lib.chalmers.se/docview/1368684792?pq-origsite=summon` (visited on 02/06/2017).

[27] Created by Ibrandify - Freepik.com. (2017). Location icons Free Vector, [Online]. Available: `http://www.freepik.com/free-vector/location-icons_908187.htm` (visited on 07/06/2017).

[28] Designed by Freepik. (2017). Selection of different pointers in flat style Free Vector, [Online]. Available: `http://www.freepik.com/free-vector/selection-of-different-pointers-in-flat-style_963571.htm` (visited on 07/06/2017).

[29] Icon made by Revicon from www.flaticon.com. (2017). Server free icon, [Online]. Available: `http://www.flaticon.com/free-icon/server_101334` (visited on 07/06/2017).

[30] Icon made by Vectors Market from www.flaticon.com. (2017). Forklift free icon, [Online]. Available: `http://www.flaticon.com/free-icon/forklift_117578#term=forklift&page=1&position=13` (visited on 07/06/2017).

A

Appendix 1

A.1 Google Tango tablet

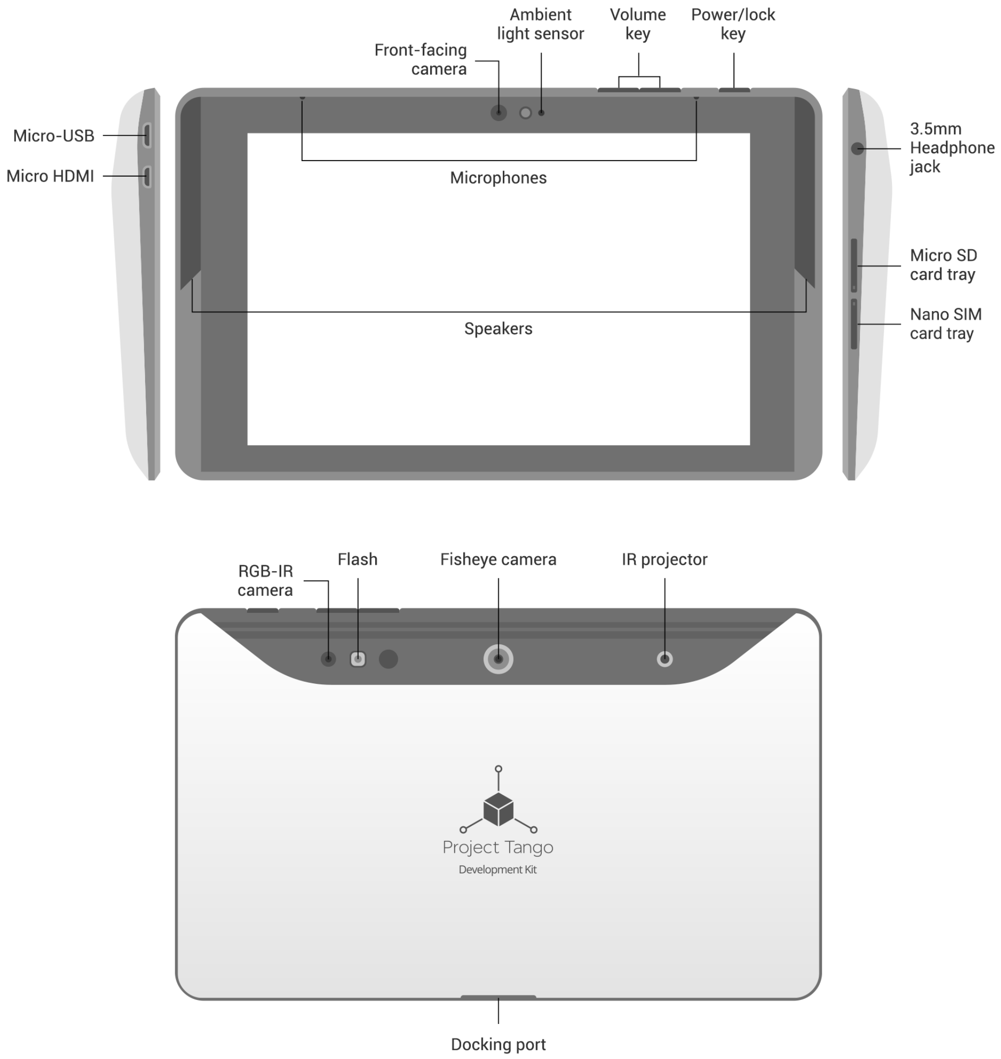


Figure A.1: Specifications of the Google Tango enabled tablet. Picture by Google [13].

B
Appendix 2

B.1 Gantt schedule

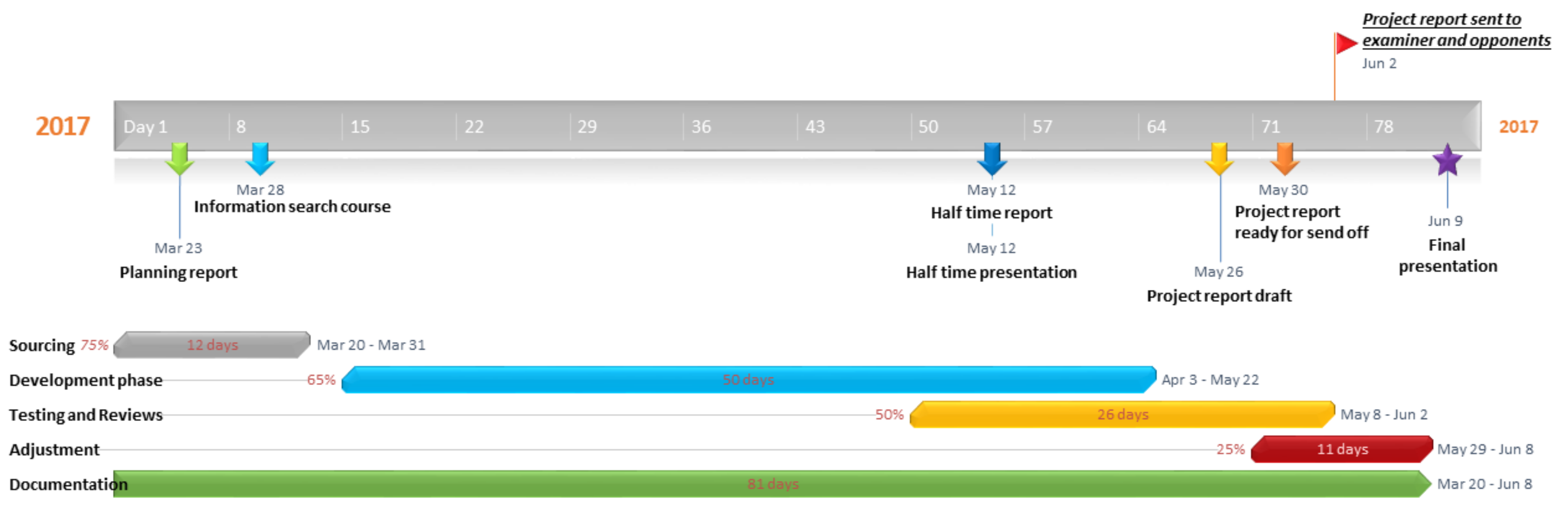
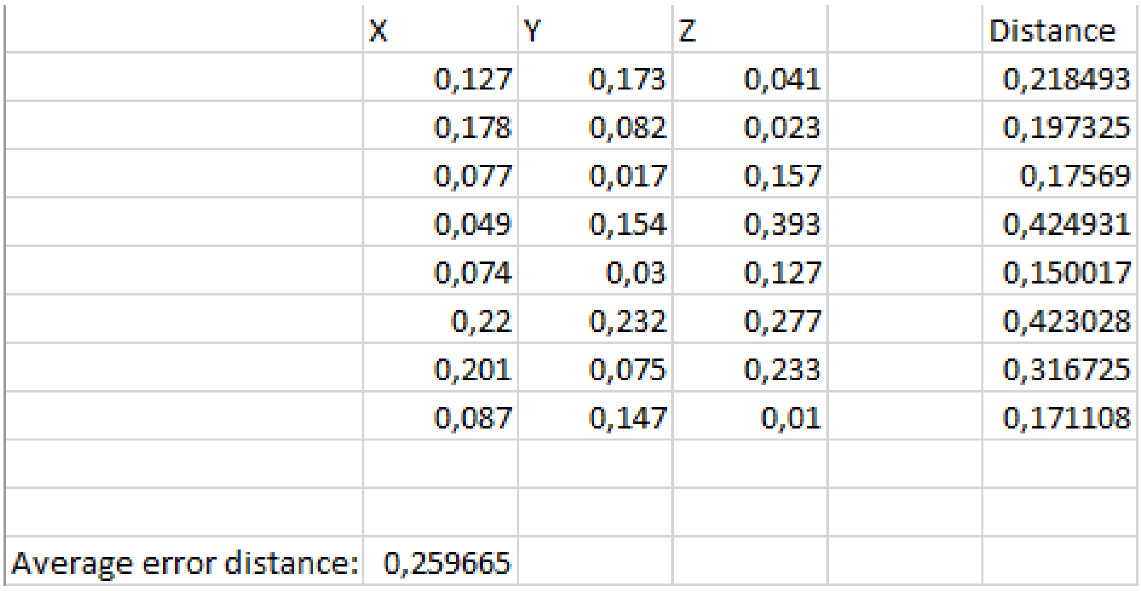


Figure B.1: Original time plan for the thesis

C

Appendix 3

C.1 Measurement data



	X	Y	Z		Distance
	0,127	0,173	0,041		0,218493
	0,178	0,082	0,023		0,197325
	0,077	0,017	0,157		0,17569
	0,049	0,154	0,393		0,424931
	0,074	0,03	0,127		0,150017
	0,22	0,232	0,277		0,423028
	0,201	0,075	0,233		0,316725
	0,087	0,147	0,01		0,171108
Average error distance:	0,259665				

Figure C.1: Table of the collected test data.