Mapping the underworld

Alexander J. M. van der Meer

July 19, 2016

Summary

This bachelor assignment contributes to the P.I.R.A.T.E. project. The project's goal is to create an autonomous pipe inspection robot for small diameter gas distribution mains. The goal of this thesis is to implement a method to create 3D reconstructions of the pipe's interior. A circular structured light sensor constructed in previous work was used in combination with odometry data from a wheel encoder and orientation data is obtained with an IMU (inertial measuring unit). All measurements are logged and 3D reconstructions are performed offline using MATLAB. All measurements are done on 125 mm outer diameter PVC pipes. Reconstructions were done for a 90 degree elbow piece, a T junction, and a small obstacle in the pipe. The incorporation of orientation data into the reconstruction is shown on a large radius 90 degree corner.

Preface

I would like to thank M. Reiling for his time and effort that have helped me to improve the system. The constructive feedback of E. Dertien was always appreciated. Special thanks goes out to my parents for having made this endeavour financially possible.

Alex van der Meer
Enschede, June 2016

Contents

1 Introduction

1.1 Context

In-pipe inspection of gas pipes is currently done on relatively small sections of pipe using an endoscope approach. Inspection from the surface is also done. In sewer pipes inspection is currently done by having a camera mounted device go through pipes and an operator manually looking at the recordings. Commercial systems are available for large pipes. The current ways of doing inspection are considered to be both expensive and time consuming.

In 2006 the research project P.I.R.A.T.E. was started at the university of Twente. The aim of the Pipe Inspection Robot for Autonomous Exploration (PIRATE) project (3) is to develop an autonomous robot platform for in-pipe inspection of small diameter, low pressure (urban) gas distribution mains. This network consists of smooth PE, PVC and grey cast iron pipes. It is already capable of operator controlled exploration but the autonomy is currently being developed.

In order to assess the quality of the pipe, detailed information on the condition of the network is needed. This means that dents, obstacles and other unwanted artefact's should be recognized and located. To do this, quantitative information on current network diameter and consistency is necessary. In order to navigate autonomously through the pipe, upcoming junctions have to be detected and recognized. To get the information that is needed for these tasks the robot needs a form of sensing the geometry in front of it. At the moment of writing, (June 2016) the robot only has a camera mounted to the front and back. In 2009 research was done by E. Drost on such a sensing technique (4). The result was a functional structured light sensor. The sensing technique consists of a laser projecting a circular pattern to the front of the robot, creating a cone structure. A camera then makes images of the projection in the pipe. Anomalies alter the shape of the perceived projection. Using triangulation the information in the image can be mapped to 3D points. This sensor was too large to be implemented in the pipes of the target diameter. In 2010 the following was done by T. Mennink (12). A way to detect and classify an upcoming junction was presented and the size of the sensor was further reduced but still not enough to fit the pipes. In 2014 the sensor was improved and reduced in size by M. Reiling (13), now it does fit the target diameter pipes. This work produced the working sensor but did not use it to actually perform a mapping of the pipe interior geometry. This thesis focuses on using this sensor to create a 3D mapping of the in-pipe environment.

1.2 Problem statement

The goal of this assignment is creating a 3D reconstruction of the pipe's interior. The reconstruction should be represented in a way that puts emphasis on anomalies in the pipe. In this way an operator can see the locations and shape of irregularities in the pipe. The reason for doing this is that this model could be used as input to automatically detect anomalies and junctions. This would serve as input to autonomous motion. Just as important, in this way an operator would only look at marked locations to save precious inspection time. The automatic artifact detection is outside the scope of this research. This work will show the reconstruction of a 90 degree bend with two sleeve lengths, a T junction, a small obstacle in the pipe and a larger radius 90 degree corner piece. Since the goal is to map the underground pipe interior, the name of the thesis is 'Mapping the underworld'. This was inspired by a large research project by the same name where pipes are being mapped by scanning from the surface (Hamilton).

1.3 The approach

The monocular structured light vision sensor developed by M. Reiling in 2014 (13) has been used as input to get info on the 3D shape of the pipe. The sensor is aligned in the middle of the pipe along the pipe axis. The sensor is attached to a cart on wheels to ensure the sensor stays in the same position (it is not attached to the current PIRATE robot). In all experiments where a straight section of pipe is mapped the position is determined using odometry only. The odometry is done using the information of an encoder on a wheel of the cart. In one experiment a long turn is mapped, here the orientation input is from an Xsens IMU (inertial measurement unit). The input from the encoder, the processed output of the sensor and the IMU data are logged and processed offline after measuring. MATLAB is used to generate a 3D reconstruction from each experiment.

1.4 Report outline

An analysis of the design contraints and state of the art are described in chapter 2. It includes an analysis of the pre-existing structured light sensor, how it works, it's limitations and how the output should be converted to geometric information. In chapter 3 the implementation is discussed. This covers the experimental setup and the realised hardware and software. In chapter 4 the experiments are treated. These consist of a description of the measurement setup, why it is relevant and images of the resulting 3D reconstructions. The report ends with a conclusion and recommendations for future work in chapter 5.

2 Analysis

In this chapter first an overview is given of what is done in related state of the art research. This will be followed by an analysis of the used structured light sensor to understand how it works and the way that the output can then be translated into 3D coordinates will be shown. Then the requirements on the output of this work are given.

2.1 State of the art

An important feature of the PIRATE robot is that it can move through various junctions in small pipes. Some robots were found that can operate in smaller pipes then the PIRATE robot. In a paper from 1999 by K. Suzumori et al. (15) a robot is described that can take elbow joints and move vertically in 25 mm pipes (only metal pipes due to use of magnetic wheels). In a paper from 2015 by T. Takayama at al. (16) a robot can move through elbow's in 20 mm pipes. Both robots have in common that no qualitative inspection is done and there is no aim for autonomy, only a camera feed is available. Since the PIRATE robot aims for autonomous inspection, information on defects and on the environment in front of the robot are needed.

The preferred method of inspecting pipes is an NDT (non-destructive testing method). This robot does NDT, since it's objective is to lower current inspection costs. In an overview of existing use of robotics in gas and oil pipe inspection from 2016 the following main methods are discussed (14). The use of eddy currents, MFL (magnetic flux leakage), ultrasonic inspection, optical inspection, and tentacle sensing. From these methods eddy current and MFL only work in metal pipes. This research also aims at inspecting PVC and PE pipes, so these methods are not investigated. Tentacle sensing is done by extended sensors that are in physical contact with the wall. This generates additional friction and increases the force and traction required by the robot, which makes it more suited for larger pipes and robots. Ultrasonic inspection can be used to create a 3D mapping using time of flight of sound waves. However since ultrasonic sensing needs a coupling medium (liquid) it is unsuitable for use in a live gas distribution mains. This leaves optical inspection as the option to use.
When looking in the IEEE database for relatively small diameter pipe inspection (below 250 mm) from 2010 onwards, the most used sensing approach was visual inspection. No mention of tactile sensing or ultrasonic sensing was found and little on using eddy currents.

Optical inspection can be done in various ways. The most straightforward method is recording camera footage without processing. But more can be done. Image processing can be done to recognize defects, also stereo vision can be used to add depth information and laser range scanners can be used to get 3D information about the pipe.

2.1.1 Image processing and Stereo vision

A study done by in T. Wu et al. in 2015 (17) and a a study from 2011 done by Yamashita et al. (18) are both examples of the use of image processing for defect detection. High resolution images are taken of an conical mirror to get omnidirectional input. They show results of algorithms that can detect corrosion and small cracks. This provides valuable, qualitative information on the state of the pipe wall. Downsides are size and power use. The conical mirror makes the sensor bulky (but no size is given). The high resolution that is needed combined with the heavy processing make for high power use. This approach alone is not enough for the PIRATE since information on the frontal situation is needed. But an optimized version of this method could act as an aid to a different main sensing method.
Stereo vision is used in a paper on sewer pipe inspection from 2015 (9) to determine 3D information on defects in the pipe (not a model of the entire pipe). They detect cracks, holes and obstacles. The sensor is relatively large and is used in 250mm pipes. With the small size of cur-

rent camera's this method could be an option but power consumption is a challenge compared to a structured light approach.

2.1.2 Structured laser light

Another approach to obtain information on the geometry of the pipe is using structured laser light. This is used the most in recent pipe inspection systems. When looking at the laser pattern choice several options are available. In 2011 Lee et al. demonstrated a robot 'MRINSPECT' which uses a line pattern is projected in front of the robot (10). This belongs to the MRINSPECT pipe inspection project started in 2005. Full geometric information requires the pattern to be rotated. They are able to detect upcoming junctions for navigation purposes but this approach is not suited for defect detection since not all geometric information is available in one frame. And high frame rates are unwanted due to power used.

A different approach is using a circular laser pattern, this was more often used in literature. This can be done as a circular pattern projected orthogonal to the pipe surface as was done in the studies done by in T. Wu et al. in 2015 (17) and done by Yamashita et al. in 2011 (18) which were mentioned before. Here a conical mirror is used to film an omnidirectional image. Both sources combine this with the image processing mentioned earlier. As was mentioned there, this approach gives great detail on the condition of the pipe wall but it cannot be used as the only sensing method as no frontal information is available.

A circular pattern can also be done as a cone projected in front like the one used in this thesis (described in the work of M. Reiling in 2014 (13). The same approach was done in was also done in 2007 by O. Duran et al. (5). In 2016 a calibration method of the cone method is shown by Zhu et al. (19). Both sensors are substantially larger implementations than the structured light sensor used in this research. Since a pipe is cylindrical, only a circle is enough to map an entire cross-section. The main advantage of this method is the low processing power since only a circle has to be located per frame. The amount of frames and speed of motion together determine how much of the geometry of the pipe is mapped. A downside is that the geometry behind an obstacle cannot be mapped. In the work of O. Duran et al. (5) the method is extended by not only looking at the geometric information that can be deducted from the laser light but also the intensity information of the concerned pixels was used. Here neural networks algorithms from artificial intelligence are used. Defects are detected using fusion between the intensity data and geometric data from the observed laser pattern. By adding this fusion they increased their detection rate and were able to detect all their test cases, existing of cracks and obstacles. This is outside of the scope of this research but it could be an addition to the existing method.

2.1.3 Pipe reconstruction

This research is about 3D mapping the inside of a pipe. Of the sources mentioned above only in the work of Yamashita et al. in 2011 (18) a full 3D reconstruction of the pipe was made. They also add the texture info retrieved from images and map it to the pipe model. This is valuable for an operator but it does significantly increase power use. In the work done by Lee et al. in 2011 on the MRINSPECT robot (10) a 3D map is made of the pipe network as a line graph, so no information on the pipe's shapes but information showing how the network is configured. This is done using IMU information and odometry information. The same approach will be taken, but with the geometric information added.

2.2 The structured light sensor

In the first section the concept of how the sensor is built and operates is explained. After this a derivation is given for the step from sensor output to 3D coordinates. This will be followed by

a note on the how the processing step from camera image to sensor output data is done. At the end the effects of misalignment are shown.

2.2.1 The concept and specifications

The sensor uses a laser to project a circular light pattern in front of it. This is then recorded by a camera. This happens in a dark environment to simplify extracting the pattern from the image that is taken. The laser stays at a fixed angle. Using triangulation the place where the pattern is captured on the image can be related to 3D coordinates. A cross section of the laser is shown in figure 2.1 . The laser is directed at an diffractive optical element (DOE) and then reflected using a mirror. The DOE still passes some through the middle but this is absorbed by a beam dump. The circular laser pattern is used since only one line of illumination is required to map the entire pipe circumference, reducing acquisition times. This method is also future proof since lasers are becoming more efficient and camera's are becoming smaller and require less power. The camera has a maximum resolution of 1280x960 pixels but is used at 640x480 pixels to reduce the computational load. As discussed in the requirements section it's accuracy has been shown to be 0.35 mm. The sensor has as longest and widest dimensions 80mm x 31 mm.

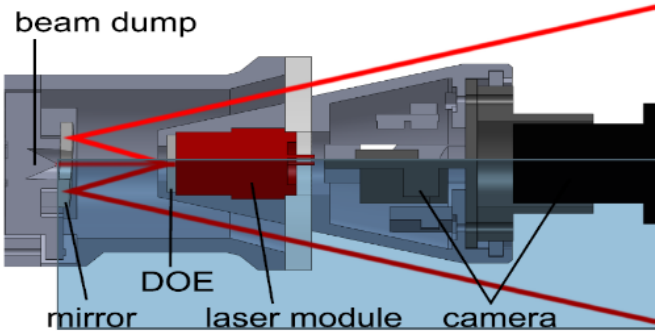

Figure 2.1: A cross section of the structured light sensor, clearly showing the path the laser takes. (13).

2.2.2 Calculating 3D coordinates from images

The camera and projector are modelled using the pinhole-camera model. This relates image coordinates to world coordinates and greatly simplifies calculations. Lines are drawn from objects straight towards 1 focal point, and note is taken where the lines cross an image plane. The model does not include lens distortion but during calibration distortion parameters can be measured, then the distortion can be compensated for. For an extensive description on the modelling the reader is referred to the thesis of M. Reiling (13).

This results in the representation given in Figure 2.2. It shows a cross section of the pipe and sensor with relevant distances. The camera focal point is shown as O_c. The camera image plane is perpendicular to figure and can be found at the focal distance f_c away from O_c. The distance r_c is the distance in pixels from the centre of the image to the point where the laser pattern is found. The laser has O_p as the point the beam originates from. The projector is modelled in the same way as the camera with an image plane and a focal distance but the plane is virtual and so it can be chosen anywhere. To ease calculations it is chosen at the same f_c of the camera. Since the laser is constantly at angle α, r_p a constant value. The values of t_z, α and f_c can be obtained during the calibration method described in the thesis of M. Reiling (13).

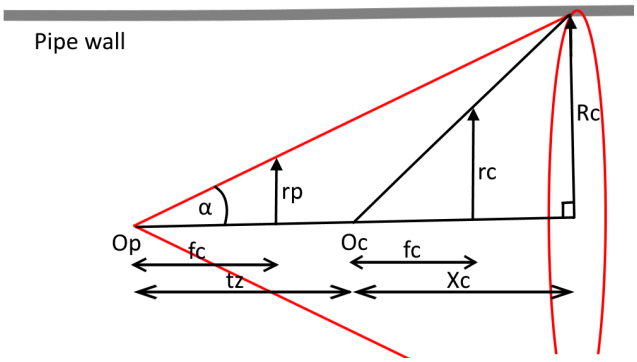

Figure 2.2: A schematic drawing a side view of the laser beam reaching the pipe wall and being filmed by the camera. Op and Oc are the focal point of the projector and the camera. Fc is the focal distance.

The values of interest in this figure are R_c and X_c. In the camera frame coordinate system, along the pipe's axis is x, vertically up is z and the sideways direction is y. When a point In the pipe is at a radial angle of θ around the pipe's axis the following hold for it's coordinates:

$$x = X_c \tag{2.1}$$
$$y = cos(\theta) * R_c \tag{2.2}$$
$$z = sin(\theta) * R_c \tag{2.3}$$

Here a derivation is made for obtaining R_c and X_c. Using trigonometry:

$$r_p = f_c \cdot tan(\alpha) \tag{2.4}$$

Using triangle ratio's this holds:

$$r_c / f_c = R_c / X_c \tag{2.5}$$
$$X_c = R_c \cdot f_c / r_c \tag{2.6}$$

Also using triangle ratio's:

$$r_p / f_c = R_c / (t_z + X_c) \tag{2.7}$$

Combining (2.6) and (2.7) gives:

$$R_c = (r_c \cdot r_p \cdot t_z) / (f_c \cdot (r_c - r_p)) \tag{2.8}$$

Combining (2.6) and (2.8) results in

$$X_c = r_p \cdot t_z / (r_c - r_p) \tag{2.9}$$

In the equations for both R_c and X_c the term $r_c - r_p$ is important. In stereo vision this is called radial disparity, depth is related to the difference between points where the image plane is reached. It can be seen there will be division by zero when $r_p = r_c$ but this will not occur since r_c only approaches r_p at X_c approaching infinity. Wat can also be seen from looking at figure 2.2, is that r_c becomes larger if an object is closer and smaller when it is further. This means that inside a pipe with a closed wall, the r_c can never become less than the value that represents the pipe wall.

2.2.3 Obtaining the radial distance, r_c from camera images

The process of retrieving the radial distance from camera images has been implemented and explained in the work by M. Reiling (13) therefore I will shortly touch on it here. First some filtering is done to put emphasis on the laser light. The program outputs 360 values for the radial distance until the laser pattern is detected which means 1 value per degree. Per value a virtual line is drawn from the centre outwards and the program calculates which pixels are relevant for this line. The light intensity values are available for all these pixels. Then a Gaussian distribution is fitted to these values creating sub pixel accuracy.

2.2.4 Effect of sensor location and orientation within the pipe

The location and orientation of the sensor is of great influence on the correct operation of the mapping. In figure 2.3. the effect on the model is shown when the sensor is in the middle of the pipe, but tilted upwards. Sideways it is properly centred. What can be seen is that when the sensor deviates from the axis of the pipe a part of the circle can get allot further from the situation without deviation. This results in reduction of accuracy since there is more distance per pixel. When the sensors orientation deviates or the position is of centre, this can be compensated for mathematically in the model, if this deviation is known. In this thesis the deviation is assumed to be zero and no compensation is therefore considered. In figure 2.3. it can be seen what happens when there is a deviation. The real line of motion is following the real pipe. However the assumed line of motion is that in line with the camera. In this case this produces the tilted tube shape, which has an ellipse shape where the sideways diameter is correct but the vertical diameter is larger than reality. Deviation will never create a smaller diameter.

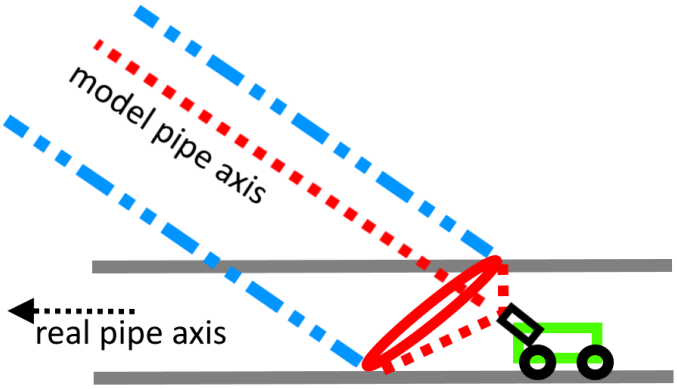

Figure 2.3: A schematic drawing showing the effect of a tilt upwards in the camera's orientation relative to the pipe, when being centred. The horizontal shape represents the real pipe, the tilted larger shape represents the pipe when being modelled.

2.3 Requirements

The requirements are devided into a section on the requirements that were already set in previous work and the ones for this research.

2.3.1 Requirements from previous work

Operating conditions

By the constraints from gas companies as mentioned in the 2014 PHD thesis by E. Dertien (3) the method used should be able to map pipes from 57 mm unto 118 mm inner-diameter. (63 and 125 mm outer-diameter). This work will be focusing on a mapping system for a 125 mm

PVC pipe since this allows for simpler prototyping and all concepts also apply to the smaller pipes.

Speed
Based on a combination of practical and financial reasons described in the work by E. Dertien (3) the desired speed of motion of the final robot is as first priority 4 cm/s and as second priority 8 cm/s. This means the sensor vessel has to be able to maintain this speed.

Resolution
The accuracy of the geometric information that can be obtained from the structured light sensor is stated to be 0.35 mm in the work of the master thesis by M. Reiling (13), here ways to improve this accuracy are also mentioned. In this thesis there will be little emphasis on absolute accuracy, since an uncalibrated sensor is used. The 3D reconstruction should however be able to show abrupt changes in surface height in the mm range.

Power and processing
For the PIRATE robot, power and processing load is an important consideration. The robot should be able to process on an embedded platform in real time. The work done in this thesis is focused on a proof of concept of the 3D mapping capabilities and does not focus on minimizing power use and processing load. The sensor vessel will be fed by a power outlet and the processing will not be done on an embedded target but on a laptop instead.

2.3.2 Requirements for this research

3D reconstruction
In the reconstructions that are made, an operator should clearly be able to spot obstacles and be able to spot and recognize the type of mapped junctions. The reconstruction should put emphasis on situations deviating from a clean empty straight pipe. Although no automatic detection of anomalies is done, a step up is provided to it.

Resolution
In the depth direction along the pipe axis, every 3 mm or less a mapping should be made. This means that at the first priority speed of 4 cm/s a mapping has to be made at least 13.3 times per second. For 8 cm/s a mapping has to be made at least 26.6 times per second. The resolution of the position sensing information should be lower than 1 mm. This resolution determines how subsequent mappings are connected together and has to be less than the 3 mm previously mentioned. For the orientation information a resolution of at least 1 degree is required.

3 Implementation

3.1 Overview

This section will provide an overview of the implemented sub-components of the system and how they are interconnected. In figure 3.1. such an overview is shown. It shows that there are three forms of inputs originating from a sensor vessel. The sensor vessel is a cart with wheels which has a wheel encoder, an IMU and a the structured light sensor attached. These three all gather and transfer data to a laptop, and do this independently. The data from all three is also logged independently. The laptop is running linux. This is done since the processing of the images from the camera requires linux based video drivers (v4l2). A box is drawn around the imaging sensor and the processing to put emphasis on the fact that this part was previously created and reported on in 2014 by M. Reiling (13). After an experiment finishes the logged data is processed in Windows on the same laptop. The log files are then imported into MATLAB. In MATLAB the data is combined into a 3D reconstruction of the pipe.

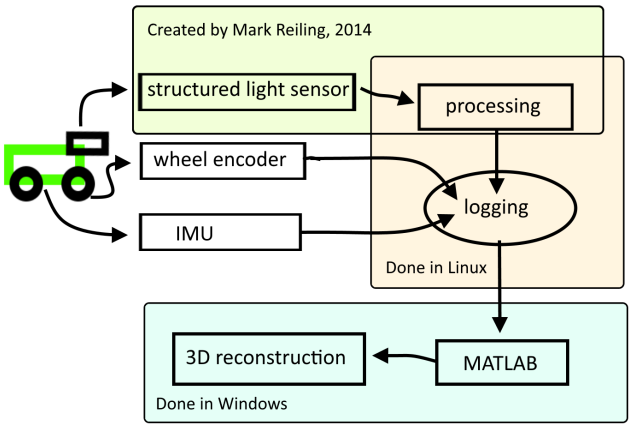

Figure 3.1: A schematic drawing showing the flow of information from measurements to a 3D reconstruction of the pipe. There are two shaded area's representing operating systems in which measurements took place. One shaded area shows what part of the implementation that was created by M. Reiling in 2014 (13).

3.2 The sensor vessel

The sensor vessel is chosen to be a cart since it is easy to keep the structured light sensor in the same place relative to the pipe walls. The cart has been 3D printed at RAM for a different purpose by a different person than the author. The cart has been modified by the author to adopt to the current situation. The cart was already equipped with a motor that incorporates an encoder. The cart is shown in figure 3.2. It shows the IMU from Xsens in the centre and the imaging sensor positioned on the front. The IMU communicates with the laptop through USB. The camera is also connected and powered through USB to the computer. During testing the contacts of the control board reduced in quality and were secured using tape.

Control over the vessel

The presence of a control board is indicated in figure 3.2. This board is also used for each subsection of the P.I.R.A.T.E. robot. It is designed as a slave node on a bus. Documentation can

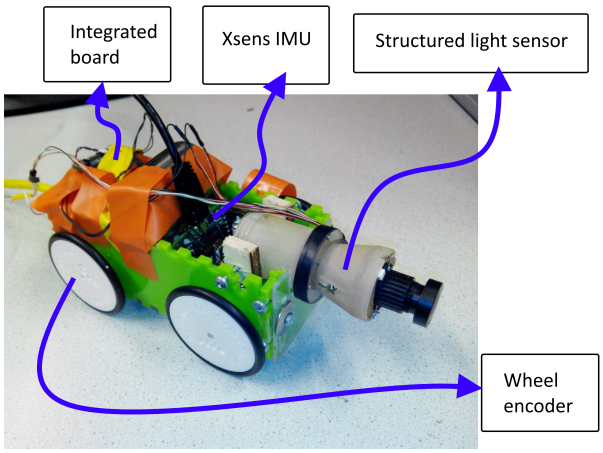

Figure 3.2: This cart is used to model the P.I.R.A.T.E. robot during the measurements and as such contains all sensors.

be found in the work of E. Dertien (3). The code is also documented there, but additions have been made by M. Reiling but this is not documented. Only minor changes were made by the author. At the back of the cart there is an Ethernet socket. The board, the motor and the laser are powered over Ethernet. The board is used to obtain encoder data. Also the board controls the motor of the cart and the power of the laser in the imaging sensor. The board communicates using RS485, by way of a CAT5 cable with an Arduino ATmega that is connected through USB to the laptop and also through USB to a nanoKONTROL2 control panel by KORG. See figure 3.3.

Figure 3.3: This image shows the KORG control board connected to a casing containing an Arduino ATmega, this is connected to the sensor vessel with the yellow Ethernet cable and to the laptop using usb.

With the connection to the computer the wheel encoder data is logged. The pre-existing code running on the AtMega is described in (3), to this code some important changes were made by the author. The KORG control panel can be used to set the motor speed and turn the laser on and off. The controller is used to move the cart during experiments. The sensor vessel was measured to achieve maximum speeds of 12 cm/s.

In order to be able to align the structured light sensor to the centre axis of the pipe, It's attachment to the cart should be adjustable. For this purpose 4 additions were made to the cart. They are shown in figure 3.4. a) shows a top view. Here it is visible that a large screw is placed through

the bottom of the cart with a wooden plateau on it. It is adjusted in height by turning it. Sideways two pieces of wood hold the sensor and can be adjusted by turning the screws. b) shows a peace of perplex with slits in it. It is attached through bolts with rings to the cart, in this way it's height can be adjusted.

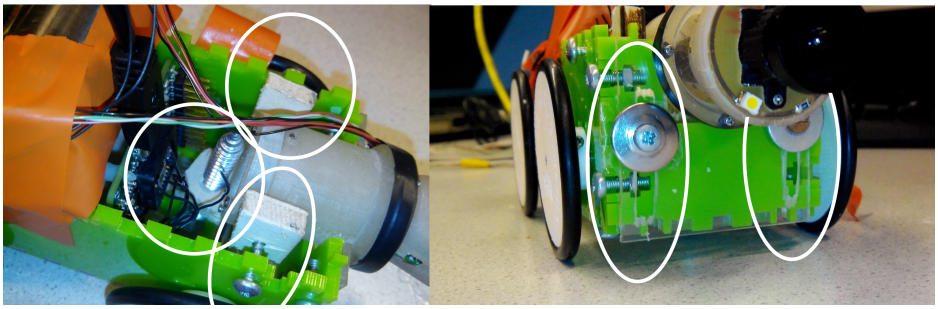

Figure 3.4: Shown in white are the methods for adjustments to the camera position

3.3 Obtaining and logging the data

This section discusses in detail what sensors are used to obtain data and how the data is obtained.

3.3.1 Structured light sensor

The sensor's laser is powered and controlled through the control board. The camera is powered separately through an USB connection to the laptop. The processing is written in c++. The processing outputs a MATLAB file containing the radial distance r_c of where the laser is detected relative to the image centre, with 360 values per frame. The program was altered to add timestamps.

The processing was originally done on an embedded target, the Avero Gumstix board. There it worked on 15 frames per second. The processing was migrated to a laptop running Ubuntu in order to allow for easier debugging and to receive all the data in one system. Due to unknown reasons the frame rate dropped to 2.08 fps. With 0.48 seconds per frame. A requirement was to do measurements at least at 4 cm/s. Another requirement was to have 3 mm in between frames or less. The speed is less relevant for the goal of creating and evaluating a 3D reconstruction of the pipe. Therefore the speed of measuring was lowered. This means a speed of 3/0.48 = 6.25 mm/s is desired. The motor's gears where not build for these speeds and generate too little force for constant speed. The cart is therefore moved by pulling or pushing the cables. Since this does not happen at a constant speed the average should be around 3 mm/s in order to stay below the 3 mm in between frames.

The sensor was aligned with the pipe axis using the adjustment possibilities. This was done by placing the cart in the pipe and adjusting until the projection looked like circle perpendicular to the pipe axis. Some small vertical tilt was still visible.

3.3.2 Odometry

For odometry an option is using wheel encoders. This has high precision but has the drawback of slipping effects. Another option is visual odometry as done in a pipe environment in 2011 by P. Hansen et al. (7). It does not have the slipping problem but it needs a well illuminated environment and has a higher computational load. Since the structured light system only works in the dark, this is not suited and a wheel encoder will be used. The wheel encoder that is used is the same as the one in the P.I.R.A.T.E. robot, an Austria microsystems AS5055a magnetic encoder. It has 12 bit resolution in the rotation angle. This resolution is more than is needed.

Therefore in the pre-existing code the amount of steps per revolution are decreased to 1800, this is 5 times 360 resulting in 1/5 degree per increment. The following was done in order to map these increments to distance. A path of 1600 mm was laid out and the cart was driven from start to end while pressing it down to ensure no slipping effects. totalIncremets / distance = 19242/1,6 = 12026,25 ticks/m. This results in a resolution of 0.08315 mm per increment. This is well within the set requirement of 1 mm.

On the Arduino ATmega a program runs that polls the encoder status on the control board in the vessel. The AT mega then sends this info through a serial port to the Arduino IDE running on the linux laptop. The pre-existing program was altered in order to increase the speed at which encoder data is polled. The polling now happens at 49 Hz. The actual frame rate of the camera is 2.08 Hz, when the system was well synced, this is also the rate at which encoder data is required. Since this is not the case a higher sampling rate is required to minimize matching errors. During testing there were problems with the wheels slipping. This was successfully dealt with by adding a heavy piece of metal to the cart to increase traction.

3.3.3 Orientation

For the orientation input an Xsens MTi-1 Development kit is used. This sensor was chosen since obtaining reliable orientation information from an IMU is a difficult task. It requires fusion of 3 axis gyroscope, 3 axis accelerometer and magnetic sensing data. Xsens uses a secret and patented sensor algorithm that can deliver the wanted orientation angles. The processing is done on the board itself. The Xsens is placed inside the cart and connected directly to the laptop using a USB cable. The board communicates to a program provided by Xsens called the MTmanager. In this program .csv log files can be created of wanted output. Such files are subsequently imported into MATLAB. The sensor experiences some change in the angles over time but this is not a continues process. A measurement of 3 minutes when the sensor is stationary produced the following changes in angles: Roll: 0.2 degree, Pitch: 0.2 degree, Yaw: 1,0 degree. For the orientation the same holds as for the encoder, 2.08Hz sampling would be sufficient but 20 Hz is chosen in order to minimize matching errors. The rate is less than that of the encoder since the orientation will change more gradually than that of the encoder. The orientation input is only briefly used. It is used in one experiment when driving through a long 90 degree corner. The rest of the experiments are done in a straight section of pipe where the orientation is assumed constant.

3.3.4 Syncing the data

In order to fuse the data for 3D reconstruction it has to be synced. The ideal situation is to have all sensors and processing done in the same program so that syncing becomes straightforward, this was not possible in the given time frame. All data is provided with a timestamp in milliseconds. A button on the KORG controller is added to both turn on the laser and turn it off. Since this is communicated to the control board on the cart, it can be printed in the encoder output. 'laser ON; start'/'laser OFF'. The output of the processing from the camera shows an error value (100) per degree when no laser is found. In the processing output the first frame where the values cease to be 100 is noted. The time at which the 'laser ON' was printed is also noted. Some small offset due to processing times is inherent. When the cart starts moving the nearest camera frame is taken and used as the start of the measurement, the same is done for the end. In MATLAB the odometry value per frame is obtained by a linear mapping from the total number of odometry values to the total number of frames. The orientation data is synced with the rest by quickly lifting the back of the cart, causing a quick change in orientation and this warps the perceived circle changing all the r_c outputs.

3.4 3D reconstruction

In this section the following things will be discussed. First sensor data is used to create a 3D reconstruction. This will be followed by how that reconstruction will be coloured, and it will end with a method to perform a software correction to combat the errors introduced by not calibrating the sensor. All 3D reconstruction related operations were preformed using MATLAB R2015B. Due to time limitations an offline analysis was chosen. ROS (robot operating system) would have been an interesting choice when looking more at real time operation. MATLAB was chosen for its pre-existing extended functionality, of which the possibility to do calculations and plot in 3D within one program.

3.4.1 Connecting the information to 3D coordinates

Here the implemented process to go from sensor input to world coordinates will be discussed. The output of the processing of the sensor that is available are 360 values per frame, one per degree, representing the r_c value in pixels. All of the following is done per input frame. With these values a 3D point cloud is created using the method described in the 'Calculating 3D coordinates from images' section in the Analysis. The $3xn$ matrix is called P_c. The steps that are taken to transform this point cloud to world coordinates, P_w is shown in figure 3.5. The flow of information during the processing of 1 image frame is shown with red arrows.

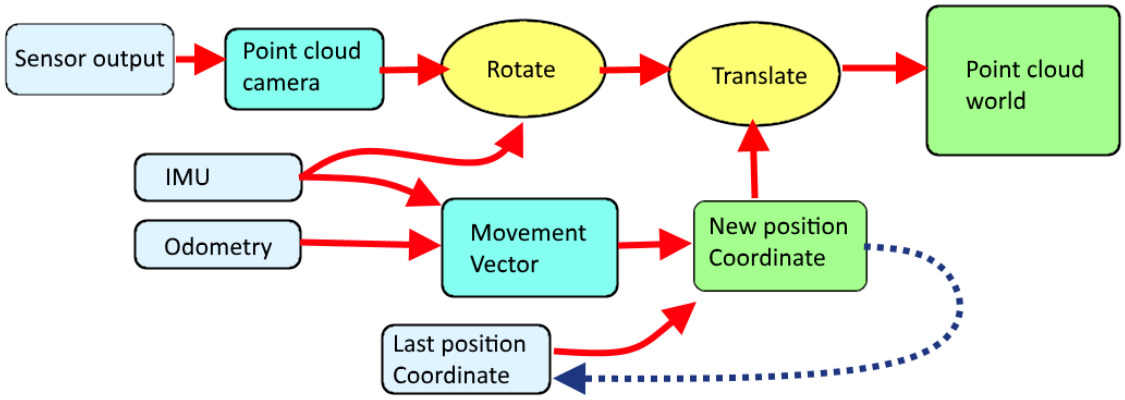

Figure 3.5: A block diagram showing the computation of 3D world coordinates using the sensor data.

To go from the camera reference frame to the world reference frame, first the point cloud is rotated. The input for this rotation are the three Euler angles obtained from the IMU, Roll, pitch and yaw. The $3xn$ coordinate matrix is multiplied by a standard rotation matrix, R. This point cloud has to be translated to the current position of the sensor vessel in the world. It is important to note that the first position coordinate for every measurement is set to $(0,0,0)$. The position is determined by a combination of inputs from the IMU, odometry and the previous position. The current orientation is combined with the change in odometry from the previous measurement point to the current measurement point (Δo) to create a movement vector, v_m. The direction in which all angles are zero is chosen to be the positive x axis. This means that v_m is calculated in the following way, where e_x is a unity vector in x direction.

$$v_m = \Delta o \cdot R \cdot e_x \tag{3.1}$$

This movement vector is added to the last position to obtain the current position. This way of calculating a movement vector in the direction of the current orientation introduces small errors in the position since it assumes the path that was taken was in straight lines from frame

to frame. A more correct but computationally more heavy method would be to use a form of curved lines that also takes the previous orientation into account.

The rotated point cloud can now be translated using the current position to obtain the desired output in world coordinates, P_w. In order to be able to translate using a matrix calculation, all coordinates used are in the homogeneous form. The translation matrix is called T.

The next equation shows the full operation to obtain the world coordinates.

$$P_w = T \cdot R \cdot P_c \tag{3.2}$$

In MATLAB the resulting P_w is plotted using the scatter3 function.

3.4.2 Applying colour to show deviations from the pipe.

Finding the 2D deviation
In order to apply a colouring some metric is required to map the colours to. For an operator the orthogonal deviation with respect to the wall of the pipe is interesting and intuitive. This information can be obtained by taking a 2D version of the point cloud in the camera reference frame. This is done by only looking at the z and y coordinates per frame. When a formula of the z, y coordinates where the pipe wall should be, is available then it is possible to calculate the distance of a measurement point to where the wall should be giving the deviation in meters. Due to a combination of the sensor not being properly aligned and calibrated, the pipe wall could be represented as an ellipse shape instead of a perfect circle. Per measurement series, one frame is chosen where only the clean pipe wall is present. To the z and y points from this measurement an ellipse is fitted using a least squares approach. (function created by R. Brown in 2007 (Brown)). For each measurement frame the deviation is calculated per point. This is done using a function that can calculate the smallest distance between a point and an ellipse, given the ellipse parameters found using the fitting function. (function created by H. Ma in 2010 (Ma)). In this implementation the reference frame to which an ellipse is fitted is chosen manually. For autonomous operation needs to find a reference itself. This could be done by monitoring for a sequence of frames that are very similar, then an operation could be performed to indicate how well this data resembles an elliptical shape. When this is the case then with high probability this a piece of empty pipe, and it can be used to use as a new reference. This process would be robust for changes in pipe diameter.

Applying a color mapping
The scatter3 function in MATLAB that is used to plot the point cloud, excepts a colour matrix, C. This matrix has the same number of columns as P_w. For each point it needs a 3 number RGB value. Obtaining the RGB values is done by taking two colours and linearly incrementing or decrementing the RBG values from the first to the second colour based on the deviation distance. The deviations that are within the pipe wall are more common and generally more interesting than deviations outside the pipe wall. Therefore 1 colour is used to map points outside the pipe and 3 colours are used for inside the pipe. For the mapping two boundaries can be set. One is the maximum distance outside the pipe until which colours should still vary, second boundary is this for the inside of the pipe. Outside the boundaries the colour is constant. This was preferred over using the maximum detected deviation as the end of the colour range, so that the type of obstacle does not influence the mapping of distance to colour. The colouring from outside inward is as follows: blue, yellow, red, green, black. The colours were chosen for their and ordered for their high visual distinctiveness. The implementation uses a twice as small distance range for the first colour (red) then for the rest. This is done because small deviations around the pipe wall are more interesting than further out, since then it already clear there is a substantial obstacle.

3.4.3 Software camera centre correction

A significant offset (to the left) was found of the circular pattern relative to the centre of the camera image. This is seen in figure3.6.

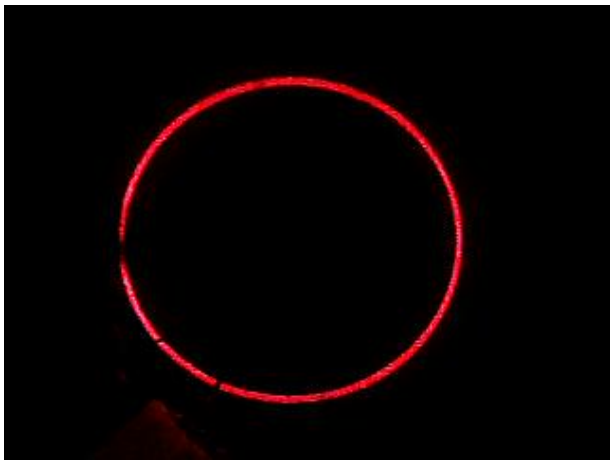

Figure 3.6: A camera shot of an empty pipe.

This means the offset is between the camera axis and the laser cone that was projected. This would normally be minimized and dealt with during calibration. Since calibration was not done due to time restraints this has a significant impact on the calculation of geometric information. It is now interpreted as if the right side of the pattern is further away than it is since a smaller distance from the centre means further away. The left side of the pattern is interpreted as closer by. When making a scatter plot of this frame this is visible. In figure 3.7. the side view (note that x is the axis of robot motion) without correction is seen. The right side is 13 cm further away than the left side while the diameter of the pipe is 12 cm. as seen in a). The sideways diameter and the vertical one are 15 cm and 17 cm as seen from the front view in b).

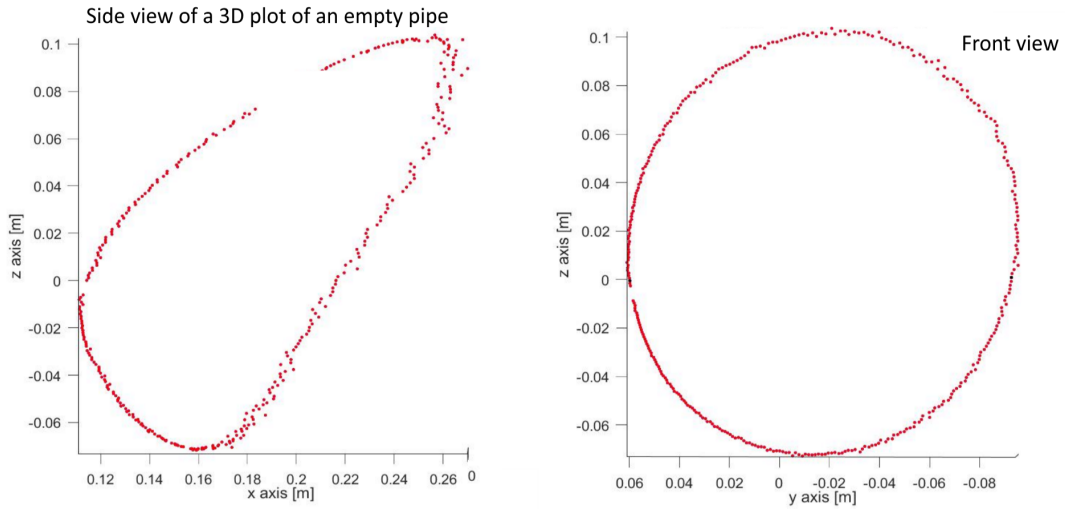

Figure 3.7: a) (left) This shows the side view without correction. b) This shows the front view without correction

For the correction it is assumed that the sensor is perfectly aligned with the pipe axis. Then the centre of the perceived circle can be used as the new point of reference to determine r_c, polar distances from that centre instead of the actual centre of the camera plane. The centre of

the circle is determined by doing an elipse fitting using a function created by R. Brown in 2007 (Brown). So an r_c input is taken and translated to what it would be with the new centre. Figure 3.8. shows how this can be done. It is a plot of the 2D camera plane where all coordinates are in pixels (not a geometric plot).|

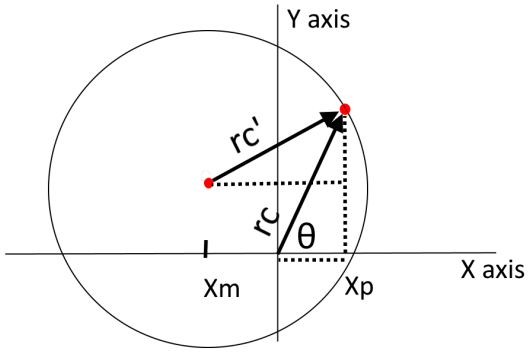

Figure 3.8: This shows a plot of the camera plane where all coordinates are in pixels. r_c' is the corrected r_c and r_c the original one.

The following equations show how to obtain the new $r_{c'}$ value using Pythagorean theorem.

$$x_p = cos(\theta) \cdot r_c \tag{3.3}$$

$$y_p = sin(\theta) \cdot r_c \tag{3.4}$$

$$r_{c'} = \sqrt{(x_m - xp)^2 + (ym - yp)^2} \tag{3.5}$$

After correction is done the sides are equally far away, as expected (see figure 3.9). The sideways diameter and the vertical one are 14 cm and 16 cm. This is closer to the true 12cm. This 'quick and dirty' correction has caused the model in figure 3.9 to show that the top and bottom of the pipe are both further then the middle. This cannot physically be true unless the pipe is significantly deformed inwards in the middle. The reason for this is that the sensor was not placed perfectly aligned with the pipe axis, it had a slight tilt downwards causing the shape of an ellipse with the vertical diameter being bigger than the horizontal one. (The reason for this is visualized in the analysis the 'Effect of sensor location and orientation within the pipe' section). When the centre of the image ellipse is taken by the correction this makes the r_c down and up equal causing them to be at equal distance of the origin, creating this warped result.

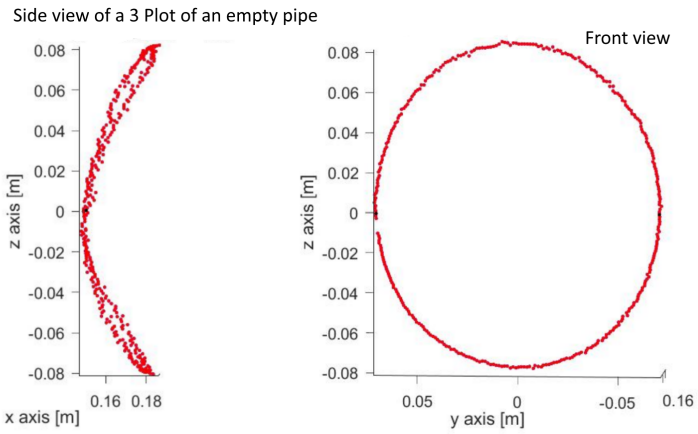

Figure 3.9: a) (left) This shows the side view with correction and b) shows the front view with correction.

4 Experiments

4.1 Overview

All experiments except one, are conducted in a Martens ecomar PVC pipe. The outer and inner diameters are 125 mm and 118 mm. Those experiments all use a straight section of 1 m length. This section is undamaged and has a circular smooth interior. These experiments are done without orientation input since the driving direction is only along the straight section of pipe. In all experiments care is taken to ensure the camera is mounted at the same position to the cart, in a way that places it in the centre of the pipe. The last experiment is done by driving through a long 90 degree corner piece made by Wavin. It has the same inner and outer diameter. In this last experiment, orientation data is used.
Since the frame rate of 2,08 Hz of the camera is lower than expected, the driving speed is too low for the motors transmission. Therefore the cart is pushed by hand resulting in a non-constant speed. This speed is mentioned per experiment.

The first two experiments are of an obstacle in the straight section of pipe. This is followed by the mapping of an 90 degree elbow piece simulating two situations of a bad connection. This will be followed by the mapping of a T-section. For each experiment a note is given on why it is relevant and details on the setup are given. The results are given mainly in the form of figure's showing the 3D reconstruction created from the measured data. After each experiment an interpretation is done.

4.2 Experiment; A block of wood

A block of wood of 2 x 3 x 10 cm is placed along the axis of the pipe. The cart is placed in such a way that the laser light is just in front of the block of wood. In this experiment the cart is not moved and only one frame of data is recorded. An image is taken from the camera showing the laser illuminated pipe, it is shown in figure 4.1, a). The output from the 3D reconstruction is visible in Figure 4.1, b). The reconstruction is viewed from the front but with a shift to the lower left. At the bottom of the pipe the points appear around 15 cm further along the pipe axis (x axis) than the rest of the ellipse shape. They also appear around 5 cm lower (on the z axis).

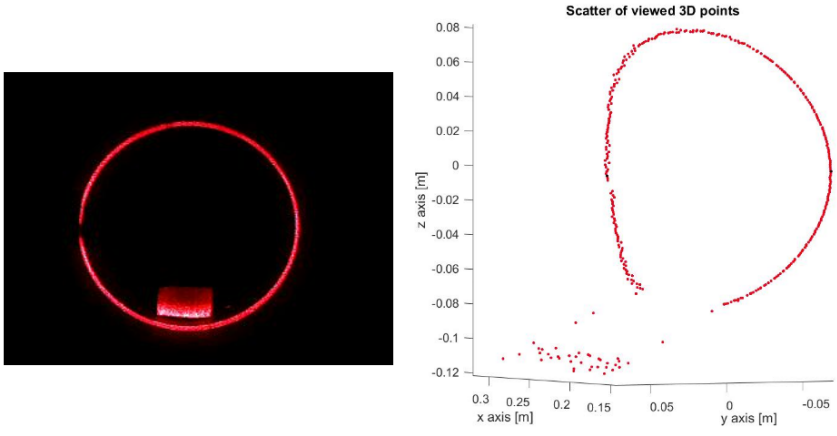

Figure 4.1: a) (left) An image taken of the current situation by the camera. b) A 3D reconstruction of the situation.

4.2.1 Interpretation

The points at the bottom of the reconstruction are clearly placed outside the pipe in 3D space. The rest of the pipe is reconstructed in the way that is to be expected from the discussion in the 'software camera centre correction' in the implementation. Regarding the bottom points, according to section 'Calculating 3D coordinates from images', in the analysis, this means that the r_c that is obtained from the camera system is smaller than that of what is to be expected for the pipe wall. When looking at the input this was indeed the case. As shown in the analysis in that same section, in case the pipe is intact it should not be physically possible to obtain this smaller r_c input. This however is based on the assumption that the processing of the image always finds the actual point where the laser strikes the pipe. The method as described in the analysis looks for the highest intensity. As can be seen in Figure 4.1, a) at the bottom of the image the reflected light that is illuminating the front of the block is more bright than the light reflected from the direct illumination of the pipe. This situation easily occurs since the surface of the pipe is not directed at the camera while a surface perpendicular to the pipe axis is directed at it. A different way of processing the image is desired. An idea is to give preference to continuing shapes. A different idea is a way to take the most outward intensity peak. The reason is that a reflection from when the laser beam strikes behind an object will have to pass through the object to reach the camera and be shown more outwards (larger r_c) which is extremely unlikely.

4.3 Experiment; Obstacle

This experiment is done in a straight section of pipe. An obstacle is placed in the pipe. The obstacle consists of a peace of pipe wall with two slits in it. One long one and one shorter one. The shorter and the longer one are 1.2 and 1.8 mm deep respectively. They are 5 mm wide. The block itself is 8 mm thick. The block is shown in figure 4.2. The experiment is done in order to see to what extent an obstacle is mapped correctly and the slits aid in testing whether the applied colouring can help to make shapes more distinct. The first frame is used as a reference for the colour mapping. The 3D reconstruction that resulted from the measurements is seen in figure 4.3.

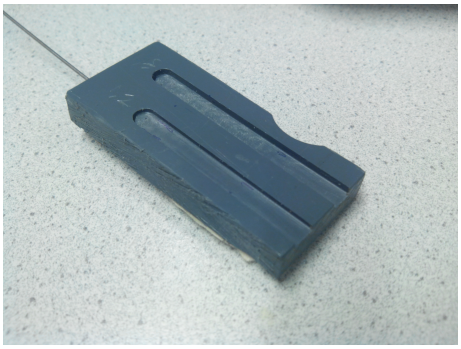

Figure 4.2: A picture of the used obstacle, showing two cuts one of 1.8 and 1.2 mm depth

4.3.1 Interpretation

The shape of the pipe is mostly yellow, indicating that the pipe shape was consistent with the reference frame that was used. The shape of the obstacle is clearly visible at the bottom of the pipe. The obstacle is proceeded by a point cloud coloured in blue, indicating the points are outside the pipe as can be clearly seen from the side view in figure 4.3, b). The points continuously go down and then abrupt start to come back up again. By the points made after the first experiment this is assumed to be caused by reflection from the frontal surface of the

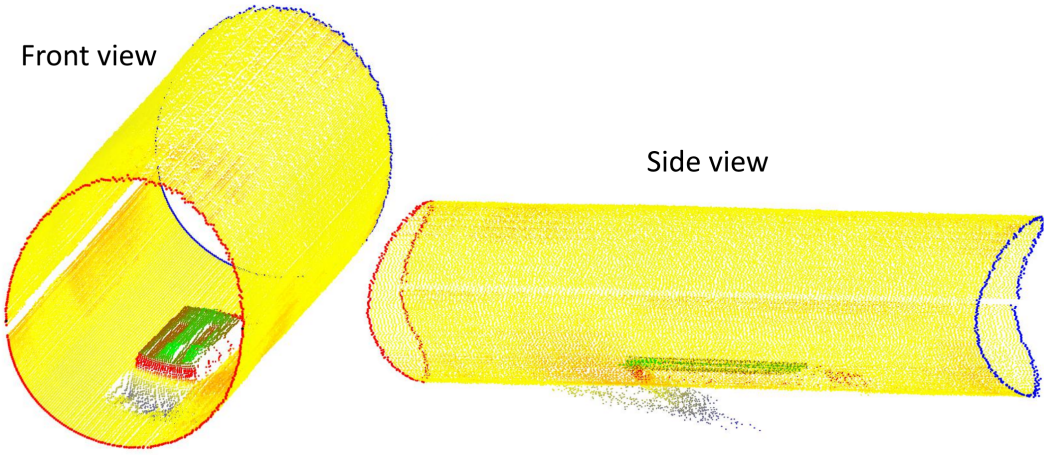

Figure 4.3: a) (left) A photo of the used obstacle. b) The 3D reconstruction of the pipe with the obstacle placed at the bottom.

obstacle. The shape of the obstacle is clearly visible and resembles the shape in figure 4.3. The colouring aids in understanding the contours of the object by making it distinct from the background (pipe wall). The difference in surface depth caused by the slits is emphasized by the red/green colour difference. From the side view in b) it can also be seen that the back of the object is represented as if it has a slope downwards. This is not the case, the ideal situation would be to show only the absence of points there.

4.4 Experiment; 90 degree turn

In this experiment the cart will drive in a straight section of pipe towards a 90 degree elbow turn. These kind of turns are common in the pipe network and are interesting for navigation purposes. The turn has a turn radius of 16 cm measured from the middle of the pipe. Since the sensor must be in the same place relative to the pipe wall, the current setup cannot be used to drive through the turn, the cart will stop when it reaches the gap within the sleeve. With the sleeve, the section that connects the straight pipe and the elbow is meant. In this experiment two gap distances within will be used. When the distance is 0 the two pipe parts are properly connected. Bad connections between sections can cause problems in the network, therefore it is interesting to see how well they can be recognized. First a long distance of 3,5 cm is used and second a distance of 1,5 cm will be used. Figure 4.4. a) shows the section of pipe to be mapped, and b) and c) are the long and short gap connection photographed from the inside. It is seen that the pipe is deepened at the place of connection. There the cart stops since else it would tilt and the sensor would not be centred.

The long gap within the sleeve experiment
The total distance travelled by the cart during the measurement was 24,6 cm. With an average speed of 2.5 mm/s. This results in an average distance between frames of on average 1.2 mm. The gap was set at 3,5 cm distance. The resulting 3D reconstructions are shown in figure 4.5 and 4.7 but first the second experiment.

The short gap within the sleeve experiment
The total distance travelled was 24,3 cm. The average speed was 2.3 mm/s. The average distance between frames was 1.2 mm. The gap was set at 1,5 cm distance. Since the cart can drive until the gap, the cart can drive 2 cm further than in the 3,5 cm case. This should produce more measurements on the pipe wall in front of the cart. Next the 3D constructions are shown. In figure 4.5. the top view is shown on the left. The pipe configuration is shown with the dotted line. The gap could be recognized in the image and was indeed 3,5 cm in the reconstruction.

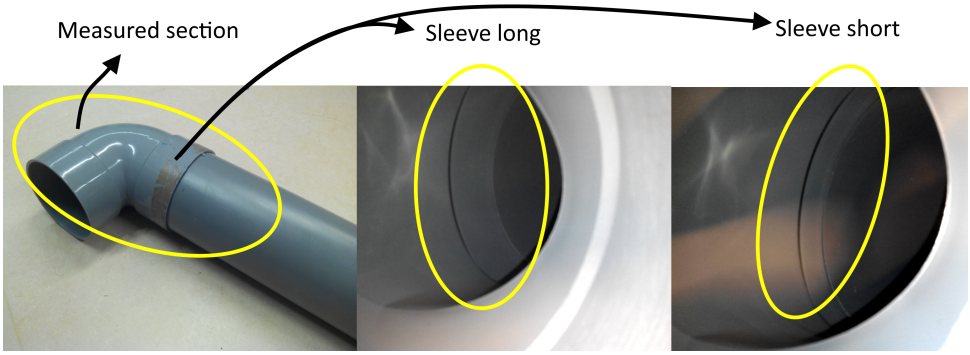

Figure 4.4: a) (left) The section of pipe to be mapped. b) The long gap within the sleeve. c) The short misalignment.

On the right the in-pipe view is shown. This shows how the gap is coloured slightly blue since it is outside the pipe. It also shows a red area where the circle ends, indicating a heightened area. In figure 4.6. the same is shown for the short gap experiment. More of the pipe wall in the corner is mapped in this case, as shown by the black colour being present.

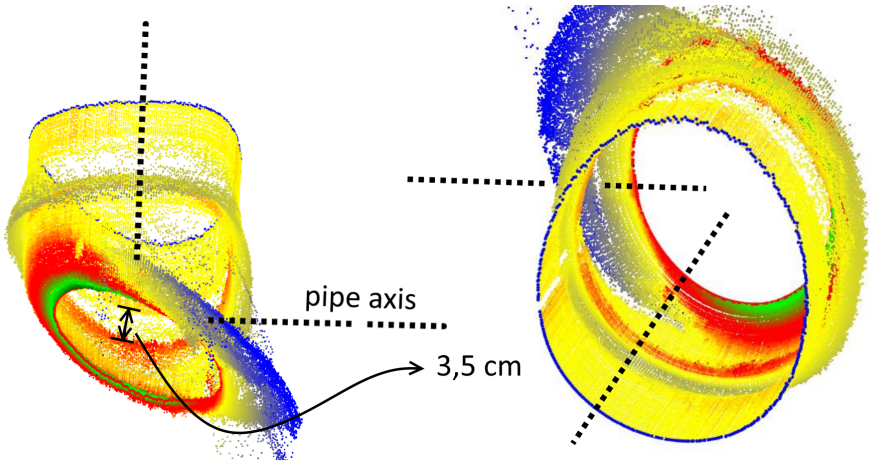

Figure 4.5: a) (left) shows a top view from the reconstruction the dotted line shows the pipe axis. The gap position and distance is shown as 3,5 cm. b) (right) Shows a view from inside the straight section of pipe.

Next in figure 4.7. and 4.8 the side views of the pipe are shown, as seen from the hole in the corner. This shows that the gap in the pipe sleeve produced points far outside the pipe wall. This view clearly shows how first there is a decrease/absence of points and then the points extend outwards to return to the original pipe wall radius.

4.4.1 Interpretation

The first impression is that the presence of a 90 degree corner can successfully be recognized. In the side view figures the difference between a long gap and a short one can easily be spotted. For the points at the gap, what is expected is the following. The normal pipe wall, transitioning into an area without points, (since the laser cannot see around a corner) followed by a section of a constant depth, if the gap is long enough. And then a short slope of approximately 5 mm followed by the normal pipe wall. The experimental results deviate from this by having a varying increasing depth followed by a a short increase in surface inside the pipe shown by red. The

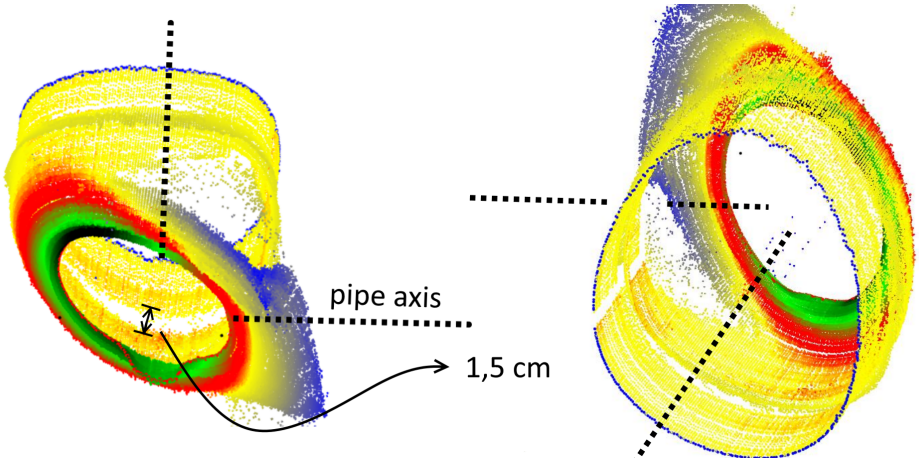

Figure 4.6: a) (left) shows a top view, with the gap distance of 1,5 cm marked, b) shows a look from inside the pipe.

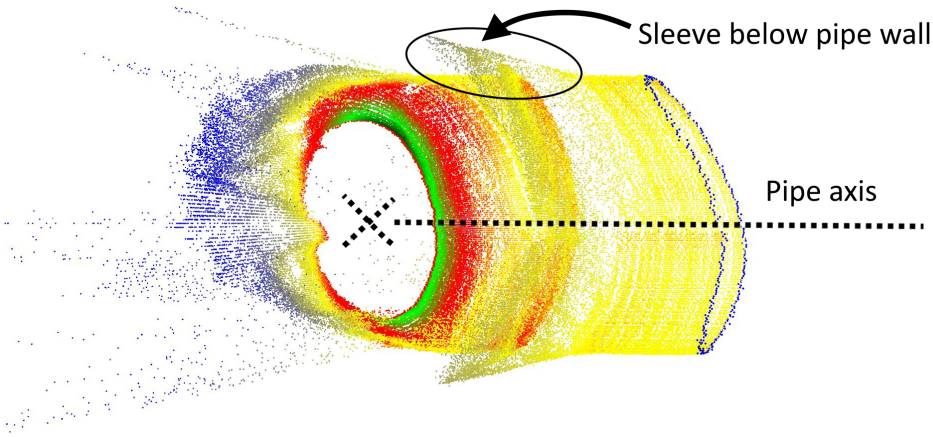

Figure 4.7: This figure shows a view of the pipe with the long gap from the side, looking into the hole of the bend. The gap can be seen to extend outside the pipe wall.

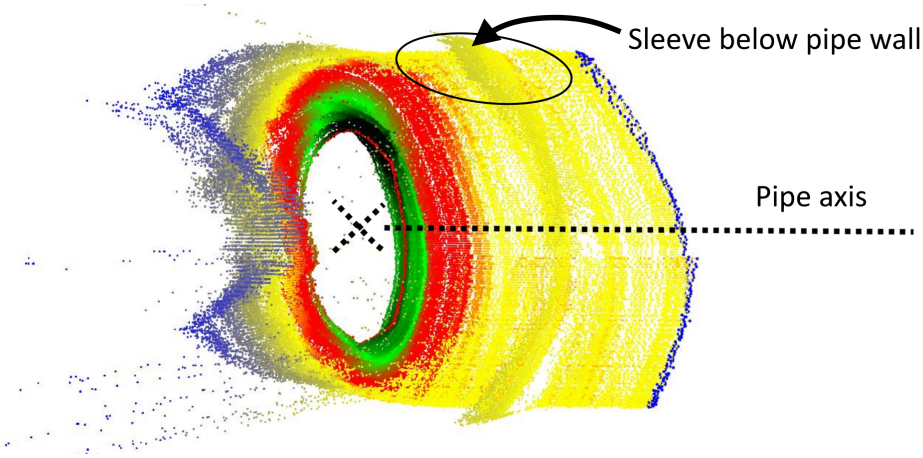

Figure 4.8: This figure shows a views of the pipe with the short gap from the side, looking into the hole of the bend. The gap can be seen to extend outside the pipe wall.

varying depth is probably caused by reflections and the cause of the red part is unknown. Since the size and location of the sleeve misalignment can be recognized from the reconstruction, this makes it probable that it could be done with automatically.

4.5 Experiment; T-junction

In this experiment the cart will drive in a straight section of pipe towards a T-junction where the cart drives into the closed end. This is a common type of junction in the pipe network. It is fabricated by Martens. Figure 4.9 shows the situation to be mapped.

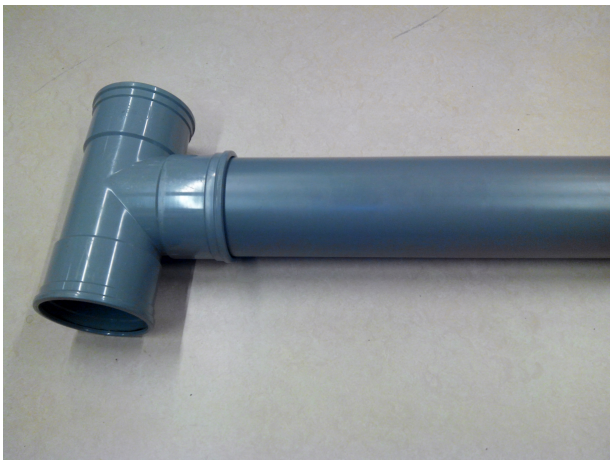

Figure 4.9: The straight section of pipe with a T-junction connected.

The measurement was done in 78 s over a distance of 41 cm, resulting in 5.3 mm/s.

The resulting 3D reconstruction that was made is shown in figure

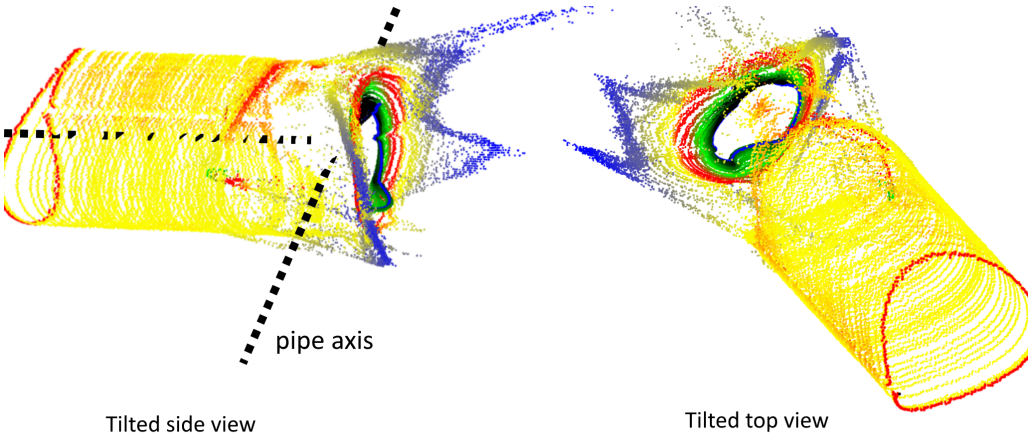

Figure 4.10: a) (left) Has a dotted line indicating the axis of the measured pipe. The left view is tilted from the side and the right view is tilted from the top.

4.5.1 Interpretation

The shape of a T junction can be found in the reconstruction, however the special relations at the end of the junction are not according to reality. The blue points indicate the walls of the junction. They shape of the junction dictates that these points should be curling inwards, instead they are observed to curl outwards. This might be caused by the sensor not being cali-

brated. Deviations do become larger when points are further away. In the left part of the figure a far stretch of points can be seen in blue, this is again thought to be caused by reflection.

4.6 Experiment; Taking a corner

This experiment is done in order to show the incorporation of the orientation data taken with the Xsens IMU into the reconstructions. A wavin PVC-a Gastec long 90 degree corner piece will be used. It is shown in figure 4.11. The pipe has the same diameter as the previously used grey pipe, 118 mm inner and 125 mm outer diameter. The measurement was done in 264 sec, the distance travelled was 90 cm, resulting in 3.4 mm/s mapping.

Figure 4.11: A photo of the corner piece of pipe that was used during this experiment.

As mentioned this experiment focuses on the orientation data and is not expected to create a decent mapping of the pipe. The cart that is used as the sensor vessle is designed to keep the camera centered in the pipe when driving straight but in a corner it cannot do this. In figure 4.12a) a camera shot is shown from within the corner, the opening at the end of the pipe is visible on the right. This shows that within the corner the pipe is only mapped on the left side of the pipe. Figure 4.12b) shows the front view of the reconstruction of one frame within the corner.

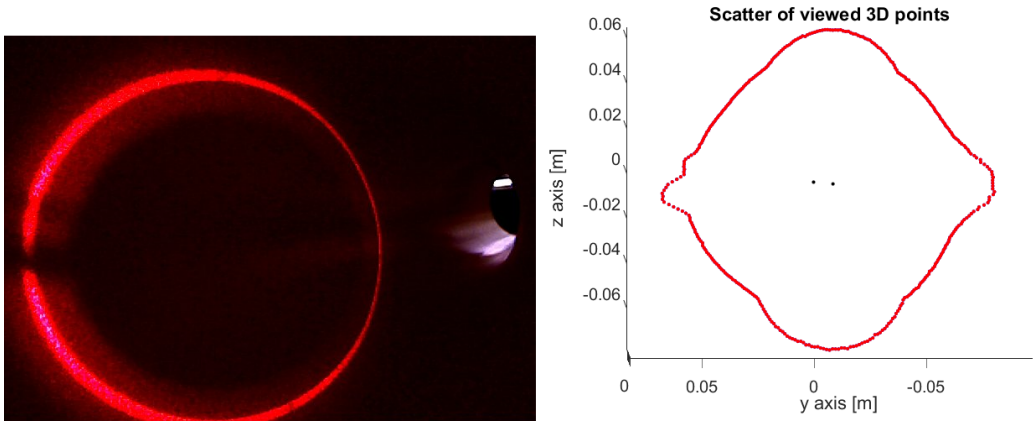

Figure 4.12: a) (left) An image taken by the camera in the corner is shown. It shows the circular pattern only on the left pipe wall. The opening at the end of the pipe is seen on the right. b) This shows the 3D reconstruction created by this image.

In figure 4.13 the reconstruction of the entire corner is shown. In a) the reconstruction uses the odometry data and IMU data together as is described in the implementation chapter. In b) A reconstruction done with the same data is shown, but here the odometry data is replaced by a linear spacing between frames. The spacing was done at a distance of 1 cm between frames. In this way the pipe is stretched out some to better see what happens to the angle information. In both figures the black lines/dots show the path that the camera took.

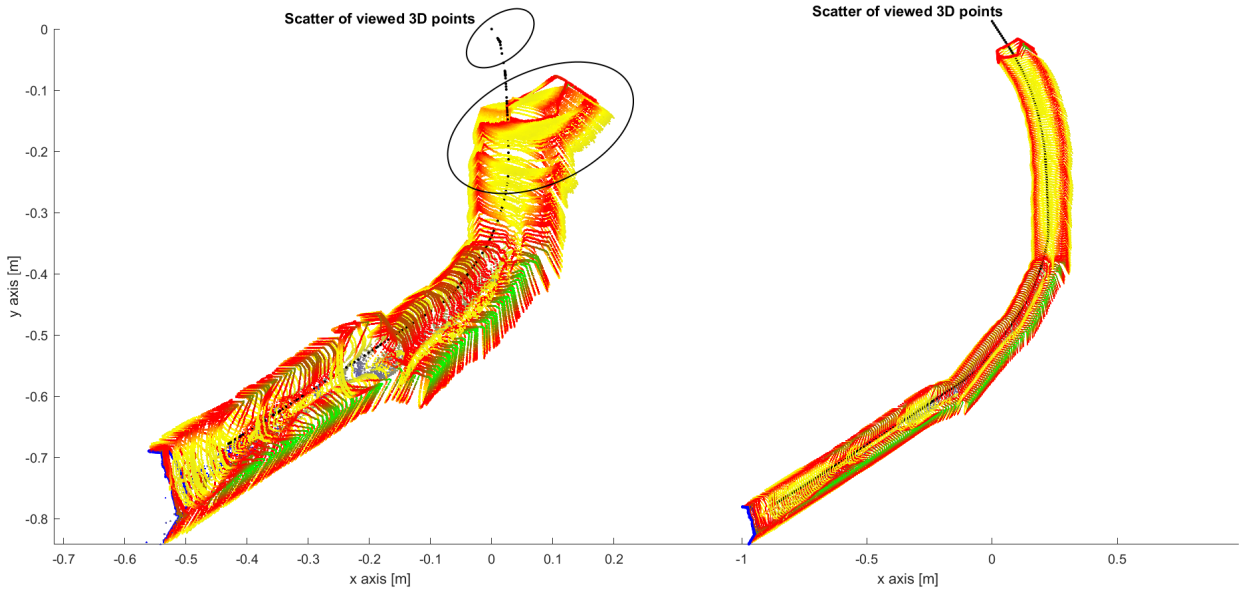

Figure 4.13: a) (left) This shows the reconstruction of the corner with odometry data included. The black line shows the path of the camera. The small circle shows the part of the camera path that is at 90 degree with respect to the start. The large circle shows multiple frames at the same location with different orientations. b) Shows the reconstruction with a linear spacing of 1 cm between frames, showing the shape of the corner

4.6.1 interpretation

The shape of the reconstruction of one frame in the corner shown in figure 4.12 b) is not as expected. The expected shape would be an ellipse that is close on the left and further from the origin (the camera) on the right. The reason why the constructed shape is not an ellipse is unclear. The set timeframe for this thesis does not permit looking into the reason for this. But to show how the cart moves, this will be sufficient.
As shown in figure 4.12 a) only the left wall of the pipe is mapped. This shows that when moving through a corner, the P.I.R.A.T.E. robot should rotate it's camera in order to map the entire pipe circumference. As mentioned before, when the exact orientation and position deviations of the camera with respect to the pipe's centre is known, the mappings can be mathematically compensated for.
In figure 4.13 a) The shape of the pipe is somewhat present but it seems like the corner is not 90 degrees but less. The small black circle in the figure that is drawn around the camera path shows some points that are in at 90 degrees with respect to the start. But it is short. The second larger black circle shows that multiple frames with multiple orientations are plotted at one location. It is impossible for this to be correct since the cart can only have one rotation at one moment. There seems to be an error in the recording of the odometry data. In figure 4.13 b) the linear spaced plot does show the shape of the corner that was used. Some deviations occurr due to the fact that the cart did not drive at a constant speed of course. This shows that the orientation can be used in the reconstruction.

5 Conclusions and recommendations

5.1 Conclusions

The methods used in this thesis successfully produced 3D reconstructions of a 125 mm outer- and 118 mm inner-diameter PVC pipe. The used concepts and methods are also expected to work in metal and PE pipes. The experiments have shown that the way of reconstructing the pipe can be used to manually recognize the following. Upcoming 90 degree elbow and T junctions, including possible misalignment of sections of pipes. It has been shown for misalignment's of 1,5 and 3,5 cm. The last experiment showed that orientation data can be successfully incorporated into the reconstruction but the current sensor vessle is not able to keep the camera centred when changing orientation, causing the reconstruction to be wrong. The model was required to clearly show abrupt changes in surface height in the mm range. The experiment with the obstacle clearly shows the 1.2 and 1.8 mm slits on the obstacle, which fulfils this requirement. The reconstruction has some clear deviations from reality. Reflections of the laser from the pipe wall onto other surfaces has been shown to lead to points that can be placed outside the pipe's dimensions. Another point is that in the reconstructions the pipe is not represented as a circular shape. This is caused by two factors, the sensor not being calibrated before use, and some deviation in the alignment of the sensor to the pipe's axis. The mapping system that is used in the final robot should be able to map with 4 cm/s or more. With the desired distance of 3 mm between mappings this resulted in a minimum mapping rate of 13 Hz. The implemented system worked at 2.08 Hz. The bottle neck was the structured light sensor, however in previous work it has been shown to be able to work at 15 Hz. This shows that the used system is able to reach this mapping speed if more debugging time is available. This has not impacted the 3D reconstruction results, they have been mapped at a lower speed in order to obtain the minimum spacing of 3 mm between mappings. The colouring of the reconstruction clearly aided in recognizing irregularities in the pipe. For the colouring a method was implemented to calculate distance deviations orthogonal to the pipe wall. This provides a step up to do automatic detection of anomalies in the pipe.

5.2 Recommendations

The three mean causes for miss-representation of reality are not having calibrated the structured light sensor, selecting reflections over the real point of projection and misalignment of the sensor with the pipe's axis. The first recommendation is to calibrate the sensor. The reflections are a problem, the solution to this problem is in the processing of the camera images. This should be investigated. An idea is to select the intensity peak corresponding to the closest object. Since reflections will most probably occur behind the point of projection of the laser. To tackle the alignment problem, pose estimation could be done in order to locate the camera position relative to the pipe. This could then be used to mathematically compensate for not being in the centre of the pipe. Having the robot not have to completely align the sensor is highly desirable for practical use. The paper by Y. Hu on pose-estimation from 2012 (8) can serve as inspiration. An improvement in order to be able to map larger objects would be to make use of an wide angle lens. During testing some obstacles were out of view.

For the goal of autonomy of the P.I.R.A.T.E. robot, anomalies have to be automatically detected and classified. The process of finding a reference to calculate whether data deviates from the pipe wall is currently done by manually choosing a reference. This should be automated. An idea is to select a sequence during real time monitoring, where the data is very similar, than it is probably the pipe wall. In order to detect upcoming junctions the work of T. Mennink provides a solution (12). Work is still to be done on the recognition and classification of defects. When this would be implemented it is and idea for the robot to start filming while illuminating the

pipe in the section where it thinks there is an anomaly. This would be valuable information for an operator. Another improvement for the operator could be making a mesh out of the current point cloud. In that way it would be easier to understand for an inspection employee and to spot irregularities. Work on this for a large tube inspection robot has been done in 2012 by A. Breitenmoser et al. (1).

An improvement for the obtaining of data would be to put all of the sensors on one master so that they can be controlled by a single program that can handle syncing of the data. This would greatly reduce the time needed to conduct an experiment. It would also reduce the frequency with which the odometry and IMU data would have to be sampled to the same one as the frame rate of the camera. Now excess is needed to reduce the effect of synchronisation errors.
Work should be done on creating a real time version of the mapping, since now it is done offline. Only in this way can it be used for navigation purposes. A SLAM approach would benefit the overall accuracy of the mapping. Also, the software has to be transformed to a different implementation that is not in MATLAB. Since this cannot be run on an embedded target.

A The implemented MATLAB code

Not all code will be shown, but only the most important code. First the main script will be discussed and this will be followed by two sections showing important helper functions.

A.1 The main script

The code that will follow is taken from the main script, coordinates5withIMU.m . Not the whole file is shown, mainly the part about reading in input files is left out. This script produces the actual 3D plot that represents the pipe, from sensor input files. First the r_c correction as described in the implementation in section 3.4.3 is applied (or not depending on the Boolean being true or false). The correction happens in a separate function, applyImageCenterCorrection(..). Then a for statement loops all frames that were recorded. It loops over the polarmaps variable, which contains the output of the processing done in Linux on camera images, which gives an array of 360 r_c values per frame. Per frame, 360 x,y,z coordinates are calculated and stored in Pframe using a function get3Dcoordinates(..). This function is shown in the next section. The next step is to rotate the points around the origin according to the current orientation of the sensor cart. Euler angles are obtained from an input log file from the IMU. From these 3 angles a rotation matrix is obtained using a function getRot(..) (get rotation) The points in Pframe are rotated using a matrix multiplication.
The if-else structure that follows is used to be able to switch between the following. Only doing a linear spacing between frames, including odometry data, and including odometry and IMU data.
When the loop is done and the colorThePipe Boolean is set to true, a colour matrix is created that contains one colour per coordinate. The matrix is created using a function called getColorMatrix(..), this function takes the deviation from the pipe wall per point as an input. The deviation is calculated by the function getDeviationFromPipeWall(..). This function will be discussed in a separate section. In the next if statement the scatter3 function is used to either plot all points with or without colour.

```
% possibly apply image center correction. The viewed circle of the
% pipe has an offset compared to the image that captures it, this
% is offset is  minimized in this way which has a large impact on
% geometrical representation
if (rcCorrectionOn)
    for (frame = startFrame:1:endFrame)
        polarmaps(frame,:) =
        applyImageCenterCorrection(polarmaps(frame,:));
    end
end

allAngles = linspace(0,2*pi,360);

% This is the starting position, and it represents the coordinates
% of the camera in the world.
positionCurrent = [0, 0, 0];
% Initialize history of camera coordinate system origen position in
% world coordinates
posHist = [
positionCurrent(1); positionCurrent(2); positionCurrent(3);];
```

```
% Total raw points, not moved as the robot moves
Praw = [ []; []; []; []; ];
Pt = [[]; []; []; []];]; % Total points of observed pipe

% This is used in order for a linear spacing of frames as xPosition
linXpos = 0;

previousODO = 0;
currentODO = 0;

for (frame = startFrame:1:endFrame)
    % Get x,y and z from one polarmap ( give a warning if frame is
    % corrupt )
    [Pframe, frameIsCorrupt] = get3DCoordinates(fc, alfa, tz,
    numberOfAngles, polarmaps(frame,:) );
    if ( frameIsCorrupt )
        error('This frame is fully corrupt, rc is only 100');
        frame
    end
    % These points will remain in the camera coordinate system. (if
    % the cart moves no depth is added, they are used for coloring)
    Praw = [Praw, Pframe;];
    % Make sure that the points where x,y,z = 0 are not plotted,
    % these are created when rc = 100;
    for ( i = 1:1:length(Pframe) )
        if (Pframe(:,i) == [0 0 0]' )
            Pframe(:,i) = [NaN NaN NaN];
        end
    end
    % Rotate points around coordinate system of camera
    if ( rotatePointCloudisOn )
        Orientation = [
        angles(1,frame), angles(2,frame), angles(3,frame)];
        R = getRot(Orientation);
        Pframe = R*Pframe;
    end

    % Concatenate frame points to all points
    % On the top a choice is made, whether only ODO info is used
    % Only a linear spacing or ODO and IMU. Here it is implemented
    if ( One_is_ODO_2isLinSpacing_3isIMU(1) )
        % This line is only used with ODO, it ads the odo input to
        % the x Coord.
        PwithODO = [
        Pframe(1,:) + odometry(frame); Pframe(2,:); Pframe(3,:);];
        Pt = [Pt, PwithODO];
    elseif ( One_is_ODO_2isLinSpacing_3isIMU(2) )
        linXpos = linXpos + frameDistance;
        Pt = [
        Pt, [Pframe(1,:) + linXpos; Pframe(2,:); Pframe(3,:)] ];
    elseif ( IMUandLinSpace )
```

```
        % Now IMU and linear frame spacing are used!
        % Update current position, by looking in wich direction you
        % travel
        movementVector = [frameDistance; 0;  0;];
        Or = [angles(1,frame), angles(2,frame), angles(3,frame)];
        R = getRot(Or);
        movementVector = R*movementVector;
        positionCurrent = [positionCurrent(1) + movementVector(1,1),
                           positionCurrent(2) + movementVector(2,1),
                           positionCurrent(3) + movementVector(3,1)];

        % Transform to homogenous coordinates for the
        % transformation.
        PframeHomo = [
        Pframe(1,:);Pframe(2,:);Pframe(3,:);Pframe(1,:)*0 + 1;];
        % Translate frame points by current position
        % Translation matrix using homegenious coordinates
        Tr = [1 0 0 positionCurrent(1);
              0 1 0 positionCurrent(2);
              0 0 1 positionCurrent(3);
              0 0 0 1 ];
        PframeHomo = Tr*PframeHomo;
        % Add frame coordinates to all coordinates
        Pt = [Pt, PframeHomo];
        % Update camera coordinate system origen position in world
        % coordinates
        posHist = [
        posHist, [
        positionCurrent(1); positionCurrent(2); positionCurrent(3);] ];
    elseif ( One_is_ODO_2isLinSpacing_3isIMU(3) )
        % Determine the distance traveled since the last frame
        currentODO = odometry(frame);
        ODOinBetweenFrames = currentODO - previousODO;
        previousODO = currentODO;
        % Update current position, by looking in wich direction you
        % travel
        movementVector = [ODOinBetweenFrames; 0;  0;];
        Or = [angles(1,frame), angles(2,frame), angles(3,frame)];
        R = getRot(Or);
        movementVector = R*movementVector;
        positionCurrent = [positionCurrent(1) + movementVector(1,1),
                           positionCurrent(2) + movementVector(2,1),
                           positionCurrent(3) + movementVector(3,1)];

        % Transform to homogenous coordinates for the
        % transformation.
        PframeHomo = [
        Pframe(1,:);Pframe(2,:);Pframe(3,:);Pframe(1,:)*0 + 1;];
        % Translate frame points by current position
        % Translation matrix using homegenious coordinates
        Tr = [1 0 0 positionCurrent(1);
```

```
                0 1 0 positionCurrent(2);
                0 0 1 positionCurrent(3);
                0 0 0 1 ];
            PframeHomo = Tr*PframeHomo;
            % Add frame coordinates to all coordinates
            Pt = [Pt, PframeHomo];
            % Update camera coordinate system origen position in
            % world coordinates
            posHist = [
            posHist, [
            positionCurrent(1); positionCurrent(2); positionCurrent(3);] ];
        else
            error('You did not enter a choice at the top,
            on using ODO, linSPace or ODO and IMU')
        end
    end;

    if ( colorThePipe )
        % Color the pipe:
        % NOTE: Only works when going straight,
        % NOTE: You have to set the cleanPipeFrame to an empty clean
        % pipe.
        % Get the deviations from the pipe wall for all datapoints
        Praw = getDeviationFromPipeWall(
        (cleanPipeFrame - startFrame) + 1,Praw);
        % Get a colorMatrix wich can be used when plotting the pipe.
        C = getColorMatrix(minDeviation, maxDeviation, Praw(4,:));
    end

    %Create a 3-D scatter plot and fill in the markers. Use view to
    % change the angle of the axes in the figure
    hold on;
    % Here the pipe is plotted, with or without colour
    if (colorThePipe)
        scatter3(Pt(1,:),Pt(2,:),Pt(3,:),1,C,'.')
    else
        scatter3(Pt(1,:),Pt(2,:),Pt(3,:),'.','red')
    end
```

A.2 The get3Dcoordinates function

This function transforms the r_c values that result from the structured light sensor his processing into 3D coordinates relative to the sensor. The code is shown below this text. It does this using the method derived in section 2.3.2. The function also checks whether an input frame is corrupted, by checking whether all values are 100. This was done since in normal operation all frames must be uncorrupted. Per point a check is done whether the value is 100, if it is, the coordinate is made to be zero. This will be changed to NaN (not a number) later in the program so that the points are not plotted. There will always be some points with the value 100 where the cables to the sensor obstruct the laser projection.

```
    function [Pframe, frameIsCorruptFlag] =
    get3DCoordinates( fc, alfa, tz, numberOfPoints, polarmap )
```

```
    frameIsCorruptFlag = false;
    corruptCounter = 0;
    thetha = linspace(0, 2*pi, numberOfPoints);
    % create 3D coordinates from 1 frame
    for (i = 1:1:numberOfPoints)
        rc(i) = polarmap(i);
        % When 100 is a value for when the view is obscured
        % (by a wire in this case)
        if ( not(rc(i) == 100) )

            % 2 dimensional
            Xc(i) = (fc * tan(alfa)*tz) /
            (rc(i) - fc * tan(alfa) );
            Rc(i) = (rc(i) * tan(alfa) * tz) /
            (rc(i) - fc * tan(alfa) );



            % 3D coordinates
            y(i) = -cos(thetha(i))*Rc(i);
            z(i) = -sin(thetha(i))*Rc(i);
            x(i) = Xc(i);
        else
            y(i) = 0;
            z(i) = 0;
            x(i) = 0;
            corruptCounter = corruptCounter + 1;
        end % End if
    end; % End for, loops 1:1:numberOfPoints

    if ( corruptCounter == numberOfPoints )
        frameIsCorruptFlag = true;
    end

    corruptCounter;

    % Construct the output
    Pframe = [x; y; z;];
end % End total function
```

A.3 The getDeviationFromPipeWall function

This function uses a clean pipe data frame and fits an ellipse function to it. Then for all data points the distance in meter to that ellipse is calculated. The code is shown after this text. The fitting and distance calculation is done in 2D, the x information which is the axis along the pipe axis is discarded to speed up calculations. An ellipse is fitted using an external function, fitEllipse(..) which was created by Richard Brown in 2007 (Brown). A loop is done over all data points. Per point the distance to the fitted ellipse that represents the pipe wall is found using a function ResidualsEllipse(..) which was written by Hui Ma in 2010 (Ma).

```
function PtWithDistance =
```

```
getDeviationFromPipeWall(cleanPipeFrame,Pt)

% This function uses a clean pipe data frame and fits an elipse
% function to it. Then for all datapoints the distance in m to
% that elipse is calculated.

% Inputs:
%   cleanPipeFrame: This is the relative frame number from the
% startFrame number where a clean outline of the pipe can be found
% in the data. This will be used to fit an elipse to, to be used for
% fitting other frames.
%   Pt:
% Pt are the 3D coordinates of all points that represent the pipe

% Outputs:
%    PtWithDistance:
%    This is the distance from the fitted pipewall to the datapoint
%    in m. This can than be used for coloring the images.

% The depth coordinate information is not used (Pt(1) or x )
points2D = [
Pt(2,((cleanPipeFrame - 1)*360 + 1):(cleanPipeFrame - 1)*360 + 360);
Pt(3,((cleanPipeFrame - 1)*360 + 1):(cleanPipeFrame - 1)*360 + 360);];
% This returns a rotation matrix and other parameters needed to plot
% the
% elipse
[Q elipsAngle a b z0 ] =
fitEllipse( points2D, 'linear', 'constraint', 'trace');

% Calculate the deviation between data and fitted pipe elipse.
for ( i = 1:1:length(Pt) )
    dataPoint = [Pt(2,i) Pt(3,i)];
    % Find the point that is closest to this point on the elipse
    % (using an externally found function at:
    % http://www.mathworks.com/matlabcentral/fileexchange
    % /27708-distance-from-points-to-an-ellipse
    [RSS, XYproj] =
    Residuals_ellipse(dataPoint, [z0(1) z0(2) a b elipsAngle]);
    % Calculate the distance between the fitted elipse and the data
    dx = XYproj(1) - dataPoint(1) ;
    dy = XYproj(2) - dataPoint(2) ;
    % Add the deviation number to the total points matrix, Pt
    Pt(4,i) = sqrt( dx^2 + dy^2);
    % Determine whether the deviation is pos. or neg., pos. is
    % inside the circle. The distances to the center are compared.
    dPoint = sqrt( dataPoint(1)^2 + dataPoint(2)^2 );
    dElips = sqrt( XYproj(1)^2 + XYproj(2)^2 );
    if ( dPoint > dElips )
        Pt(4,i) = Pt(4,i) * -1;
    end
end
```

```
% Output
PtWithDistance = Pt;
end% end total function
```

B The detailed experimental procedure

I will now explain the exact way in which data was collected during an experiment. This is aimed at a reader who wants to continue this research with the same setup. I will only describe an experiment that uses IMU data since from this the procedures for doing experiments with only odometry or only structured light sensor information can be derived by doing less actions. First the cart is placed in the pipe. The end of the pipe is covered using a round cut-out from foam. On a computer running windows (a personal laptop was used) the Arduino IDE is run and the Xsens MTmanager program is booted. In the Arduino IDE the serial monitor is used to view the output of the Arduino atMega which prints the wheel encoder value, the setpoint for the motor and a timestamp in milliseconds. The structured light sensor is connected over USB to a computer running Linux. (This was a PC at the RAM lab). The line to start the logging of the structured light sensor is put into a terminal put not started yet.
Before the actual measuring some syncing has to be done. Using a button on the midi controller the laser on the cart is turned off. Then the line to start the camera logging is run from the Linux terminal. Next the S button is pressed on the midi controller, which turns the laser back on and prints that it has done this, in the Arduino serial monitor. Now this moment will be seen in the camera processing output as going from all values being 100 (no laser detected) to varying values. Using the timestamps the data can be synced. Now the IMU has to be synced. The IMU logging is now started in the MTmanager program. The cart is quickly tilted up two times. When the cart is tilted, the angles change quickly in the IMU log file, and the camera output should produce constant values when the cart is not moving, the tilting will quickly change the distance of detecting the laser pattern resulting in changed r_c values.
Now that all necessary syncing operations are done the measuring can begin. On the midi controller the cart is made to move forward, the cart is helped to move by hand by pushing the cables connected to it. When the measurement is done the log file from the IMU is saved. The log file from the camera processing is automatically saved to a .m file. The contents are copy pasted into a new file. The contents of the Arduino serial monitor are copy pasted to a text file and saved. The odometry text file obtained from the serial monitor is imported to a Matlab variable using the build in import function which can separate the txt file into columns by looking at spaces.

Now that the experiment is done some additional actions are required for syncing. Some values need to be noted and used to derive other values. In the odometry text file, the timestamp from the moment the laser is first on should be noted. It is found by the print statement 'laser first on'. The timestamp and odometry value of when the actual experiment started should be noted by searching for when the values start to increase. The odometry value and timestamp of when the experiment ended should also be noted, this is seen by the change from increasing odometry values to constant ones. Now the camera processing output should be inspected. The frame number and timestamp from the frame where the laser is first on should be noted, the method of detecting was explained earlier. To find the frame and timestamp at which the IMU was synced by tilting the cart twice, the frame at which the first tilt starts should be found. This is done by plotting the camera processing output in a linear spacing plot and manually recognizing this moment by a large deformation of the pipe shape. In the IMU output, the moment the first tilt is done should be recognized by seeing a change from constant angles to varying angles.
From these values the following should be derived. The frame at which motion started and at which the experiment ended, the time of the last relevant IMU sample. This is done by entering the timestamp values noted before, into a script found in the HelpingScripts folder, which is found in the file structure that belongs to this research. It gives the times at which the exper-

iment started and ended, by manually looking at the timestamps in the camera log, the nearest matching frames can be selected. Now the next step is to alter the camera log .m file to insert the following at the top (where the values are from an example).

```
startFrame = 73;
endFrame = 329;
laserFirstOnFrame = 12;
motionEndFrame = 329;
```

The last step is to fill in, info in the main Matlab script so that the input files can be found. This is explained in the comments in the code.

Bibliography

[1] Breitenmoser, A. and R. Siegwart (2012), Surface reconstruction and path planning for industrial inspection with a climbing robot, in *2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, pp. 22–27, doi:10.1109/CARPI.2012.6473354.

[Brown] Brown, R. (), fitellipse.m - File Exchange - MATLAB Central. http://www.mathworks.com/matlabcentral/fileexchange/15125-fitellipse-m

[3] Dertien, E. C. (2014), *Design of an inspection robot for small diameter gas distribution mains*, Ph.D. thesis, Enschede. http://doc.utwente.nl/91336/

[4] Drost, E. (2009), *Measurement system for pipe profiling*, Msc report 003ce2009, University of Twente.

[5] Duran, O., K. Althoefer and L. D. Seneviratne (2007), Automated Pipe Defect Detection and Categorization Using Camera/Laser-Based Profiler and Artificial Neural Network, **vol. 4**, no.1, pp. 118–126, ISSN 1545-5955, doi:10.1109/TASE.2006.873225.

[Hamilton] Hamilton, M. (), Affiliated Research - Mapping The Underworld - Engineering and Physical Sciences Research Council, University of Birmingham, University of Bath, University of Leeds, University of Sheffield, University of Southampton. http://www.mappingtheunderworld.ac.uk/affiliatedresearch.html

[7] Hansen, P., H. Alismail, B. Browning and P. Rander (2011), Stereo visual odometry for pipe mapping, in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4020–4025, doi:10.1109/IROS.2011.6094911.

[8] Hu, Y., Z. Song and J. Zhu (2012), Estimating the posture of pipeline inspection robot with a 2D Laser Rang Finder, in *2012 IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 401–406, doi:10.1109/MFI.2012.6342999.

[9] Huynh, P., R. Ross, A. Martchenko and J. Devlin (2015), Anomaly inspection in sewer pipes using stereo vision, in *2015 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pp. 60–64, doi:10.1109/ICSIPA.2015.7412164.

[10] Lee, D. H., H. Moon and H. R. Choi (2011), Autonomous navigation of in-pipe working robot in unknown pipeline environment, in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1559–1564, doi:10.1109/ICRA.2011.5980503.

[Ma] Ma, H. (), Distance from points to an ellipse - File Exchange - MATLAB Central. http://www.mathworks.com/matlabcentral/fileexchange/27708-distance-from-points-to-an-ellipse

[12] Mennink, T. (2010), *Self Localization of PIRATE Inside a Partially Structured Environment*, Msc report 027ce2010, University of Twente.

[13] Reiling, M. (2014), *Implementation of a Monocular Structured Light Vision System for Pipe Inspection Robot PIRATE*, Msc report 016ram2014, University of Twente.

[14] Shukla, A. and H. Karki (2016), Application of robotics in onshore oil and gas industryâĂŤA review Part I, *Robotics and Autonomous Systems*, **vol. 75, Part B**, pp. 490–507, ISSN 0921-8890, doi:10.1016/j.robot.2015.09.012. http://www.sciencedirect.com/science/article/pii/S0921889015002006

[15] Suzumori, K., T. Miyagawa, M. Kimura and Y. Hasegawa (1999), Micro inspection robot for 1-in pipes, **vol. 4**, no.3, pp. 286–292, ISSN 1083-4435, doi:10.1109/3516.789686.

[16] Takayama, T., H. Takeshima, T. Hori and T. Omata (2015), A Twisted Bundled Tube Locomotive Device Proposed for In-Pipe Mobile Robot, **vol. 20**, no.6, pp. 2915–2923, ISSN 1083-4435, doi:10.1109/TMECH.2015.2411752.

[17] Wu, T., S. Lu and Y. Tang (2015), An in-pipe internal defects inspection system based on the active stereo omnidirectional vision sensor, in *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pp. 2637–2641, doi:10.1109/FSKD.2015.7382373.

[18] Yamashita, A., K. Matsui, R. Kawanishi, T. Kaneko, T. Murakami, H. Omori, T. Nakamura and H. Asama (2011), Self-localization and 3-D model construction of pipe by earthworm robot equipped with omni-directional rangefinder, in *2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1017–1023, doi:10.1109/ROBIO.2011.6181421.

[19] Zhu, Y., Y. Gu, Y. Jin and C. Zhai (2016), Flexible calibration method for an inner surface detector based on circle structured light, **vol. 55**, no.5, p. 1034, ISSN 0003-6935, 1539-4522, doi:10.1364/AO.55.001034.
https://www.osapublishing.org/abstract.cfm?URI=ao-55-5-1034