RAM ROBOTICS AND MECHATRONICS

Inspection of underground water pipes with use of a laser profiling sensor and offline processing

D.V. (Dylan) Kamermans

BSc Report

Committee:
Dr.ir. E.C. Dertien
Dr.ir. L.L. olde Scholtenhuis
Dr.ir. F. van der Heijden

February 2017

001RAM2017
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE.

MIRA CTIT
BIOMEDICAL TECHNOLOGY AND TECHNICAL MEDICINE

Summary

The pirate project focusses on the in-pipe inspection of gas distribution pipes. In this project the goal is to develop an autonomous inspection robot which can inspect these pipes. In previous work a sensor was developed. This sensor is able to create a laser projected cone and capture this with a camera. This laser will show if there are differences in the internal wall of the pipe. In another previous work is this data used to create a 3d reconstruction of the pipe. The goal of this research is to implement previous systems into a working system to use for the inspection of water pipes. Therefore the system needs to small, stand-alone and waterproof. Since it is the goal to inspect water pipes of at least 600 meter, it has to be analysed if the sensor is capable of doing inspections of this length. Is this not the case then the sensor has to be improved. Or find other ways of achieving the goal.

To implement previous systems into a working system it should be first known what to implement and what is needed for a proper inspection of the internal wall of the pipe. With the use of a laser profiling sensor the pipe can be inspected. The laser profiling sensor consists of a structured laser light and IMU. The generated data is logged on-board on two micro SD cards. This data is then used for offline processing in matlab. In matlab there is a reconstruction made of the laser projection. When there are flaws in the pipe, it can be seen in the reconstruction.

Contents

1 Introduction

1.1 Context

Due to leakages in water pipes 3,3 billion liter water is wasted arround the globe every day (Andrew Johnson and James Burton, 2017). This is an enormous ammount of water. A lot of households could benefit from such ammount of clean water. Since replacing every pipe in the ground or digging them up and inspect them from the outside is too expensive, they have to be inspected from inside the pipe. This is to reduce the costs. Such that instead of replacing the whole pipe, only parts have to be replaced if there are flaws.

The PIRATE project started in 2006 at the University of Twente. The aim of the project is to develope an autonomous in-pipe inspection robot for small diameter low pressure gas pipes. For this research a device for the inspection of water pipes has to be developed.

In-pipe inspection of underground water pipes is currently done in empty and large pipes. A robot with a camera is released in the pipe and is controlled by an operator above the ground. The robot is directly connected with the operator such that the data it is generating is directly send to the operator. This is done on relatively small sections of the pipe. For the inspection the device has to be able to go through PVC and AC pipes of 200 mm diameter. To inspect these pipes without households to not have water at any time, the device has to be going inside the pipe and has to be waterproof. Another thing the device needs to have is to withstand the water pressure.

1.2 Problem statement

In this assignment it is the goal to implement the already developed subsystems into a single system for the use of in-pipe inspection of water pipes. This system should be fully waterproof and should be able to inspect the pipe using a non destructive method. Because the system has to be waterproof it is not able to attach a cable between the system and a computer to send data to, such that the system needs to be stand alone. All the necessities needed for a smooth measurement should be on board of the system. After the measurement is done the generated data has to be retrieved from the system and stored on a computer for further processing. Since the inspection takes place in pipes of arround 600 meter it will generate a lot of data. This data has to be processed which will take a lot of time.

1.3 The approach

To inspect the interior wall of the pipe a laser profiling sensor is used. This sensor consists of a structured light vision system developed by M.Reiling in 2014(Mark Reiling, 2014). The structured light vision system generates a laser projected cone on the interior wall which is filmed by a camera mounted on the front of the sensor. This video is stored on a micro sd card on the camera board. Another part of the laser profiling sensor is a internal measurement unit(IMU), the IMU measures the orientation of the sensor and this data is also logged on a micro sd card. The data from the sd cards are then used for offline processing in matlab. The laser profiling sensor is mounted in a waterproof bus which is placed in a prop which is released in the water pipe. The prop is moved through the water pipe with the help of water pressure at the back of the prop. A wheel encoder at the back of the pig measures the distance it has run.

1.4 Report outline

This report continues in chapter 2 with a state of the art analysis about laser profiling systems. After this an analysis is done about the requirements and constraints of which the design needs to handle. In chapter 3 is the design which was given at the start of the research explained and

analysed. The analysis is based on the long term runs of the device. In chapter 4 the performed experiments and their results are discussed and used for recommendations of the design. In chapter 5 a conclusion will be given and the recommendations will be summarized.

2 Analysis

In this chapter an analysis will be given of laser profiling. Furthermore the method of inspection using a structured laser light will be explained. After that the requirements will be set of which this work should be to do or have.

2.1 Pipe inspection

For pipe inspection a non destructive method is preferred to do. The costs for such a inspection method are lower than removing entire pieces of pipe and inspect them. In a paper by (Poozesh et al., 2007) non destructive ways of inspection is presented. In commercial use of in-pipe inspection of water pipe there are multiple ways used. Breivoll inspection technologies(BIT)(Breivoll inspection technologies, 2016) is a compnay which focusses on the inspection of water pipes using acoustic resonance sensor. This type of sensor measures the thickness of the pipe on different places. But this system requires the pipes to be made out of cast iron or steel. Since in the PIRATE project the use of vision is used to navigate and inspect the pipe this will be explained in here. Using vision as inspection method can be simple. Just add a camera and a light on the front of the sensor and the pipe can be inspected by the human eye. Only thing to do is to have an operator above the ground looking at the output of the sensor. This is a relatively simple but effective method. However more can be done to inspect the pipe. In a research on the university of delft the goal is to inspect sewer pipes (Walter van der Schoot, 2015). The sewer pipes are dry and the setup is static however they use laser profiling to inspect the pipe. This is a very effective way of inspecting flaws in the inside wall of the pipe. The laser is projected on the inside wall of the sewer pipe. When the laser is deviating from the 'standard' structure of the wall there is a flaw. In figure 2.1 is the principle of laser profiling shown. The cart moves through the pipe and a laser is projected alongside the internal wall of the pipe. Another research by C.W. Frey a different kind of laser profiling is done (Frey, 2008). Four straight laser modules are used and they constantly rotate during measurements. With this kind of inspection method an in-depth rotation is the result. This is done to map the inside surface of the pipe. The end result is arround the same when there is a 'circular' laser patern used. But with the straight laser modules the length of the laser is determining if there is some clay for example on the inside of the pipe. Continuing on laser profiling for water pipes with the pipe still active, such that there is a lot of water going through the pipe will the inspection is performed. This require that the sensor could withstand a lot of water pressure and that it has to be waterproof. A way of moving a robot through the pipe is PIGing. Pipe Inspection Gauges move through the water pipe using water pressure. The speed of such a PIG can be controlled by the water pressure and water flow.

2.1.1 Structured laser light

In previous researches about laser profiling the setup was large and not suitable for 200 mm pipes.

The inspection has the goal to inspect a PVC pipe of 200 mm in diameter. This will be further explained in the design chapter.

With the use of a structured laser light developed by M. Reiling (Mark Reiling, 2014) a laser projection is made to inspect the pipe. This setup is very small and able to work in 200 mm pipes. The principle of the sensor is as follows: A laser module projects in a circular shape in front of the sensor. A camera in front of the sensor captures this laser projection. If the laser projection on the wall of the pipe for example is not a circular shape then there is a flaw in the pipe. A cross section of the sensor is shown in figure 2.2. The laser module project a single

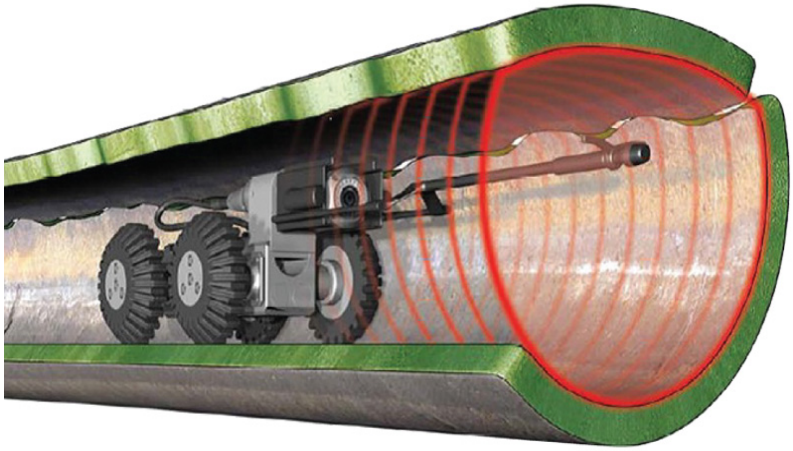

Figure 2.1: Principle of laser profiling (south east services, 2017)

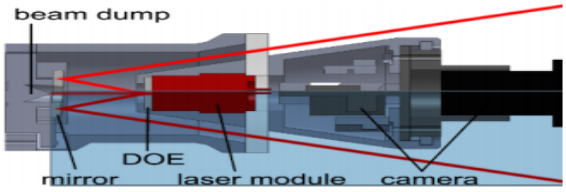

Figure 2.2: Cross section of the structured laser light sensor(Mark Reiling, 2014)

laser to the Diffractive Optical Element(DOE) which splits the laser in a circular shape. A mirror makes sure that the laser is projected to the front of the sensor.

2.2 Requirements

2.2.1 Previous work

In previous pirate work close to this work it was the goal to achieve a 3D reconstruction of a gas distribution pipe (Alex van der Meer, 2016). The reconstruction was made such that abnormalities and obstacles could be detected. This was done using a sensor vessel with a structured light sensor moving through an empty pipe. This vessel measures the odometry, orientation and the laser projection in front of the vehicle. For that research the pipe its orientation is known. When the pipe is flat on the ground and the vessel moves through the pipe it measures the internal wall using a structured laser light. When the orientation of the camera is slightly upwards it is shown in the values of the IMU and can be used to get a better view of the pipe. In this research the orientation of the pipe is not known, since the pipe is underground such that the assumption has to be made that the sensor is exactly in the middle of the pipe and has the laser projection in a nearly perfect circle. In this way the IMU should measure the pipe its orientation, since this would be the same as the sensor its orientation.

Since this research is about the inspection of water pipes a lot of requirements has to be different to the ones in previous systems. In previous work the requirements for speed, resolution, power, processing and the reconstruction can be found in the table below.

speed	4 cm/s
resolution	3 mm in depth orientation in 1 degree
power	laptop or outlet
processing	no priority on duration
3D reconstruction	emphasis on deviating situations

2.2.2 This work

The requirements for this research are explained in this section. The requirements are set such that the system could be used for long runs. The data below are based on the plan of attack for the inspection and on previous work.

Data to be inspected

The different data which want to be inspected are found below. These were found in the plan of attack of the inspection and are from Acquaint (Acquaint, 2017).

- Length of the pipe
- Coordinates of the pipe
- Angularly displacement
- Ovality
- Diameter changes
- Leakages

Setup

The inspection will take place in 200 mm PVC water pipes. The whole construction is placed in these pipes and will move with a maximum speed of 0,5 m/s.

Resolution

The resolution for this research should be arround the same as previous research where a 3D reconstruction is made. Since the system is used for inspection of the pipe and not for navigating through the pipe a in depth distance of 3 mm or less between frames there can be an accurate reconstruction made of the pipe.

Power

Because this research focusses on the long run in-pipe inspection of water pipes it should be clear that the complete system is stand alone, waterproof and requires a power source on board. This power source could be a battery pack with a fixed voltage output or a power bank where the voltage is already fixed. The inspections are in pipes of 600 meters. Therefore the power supply should deliver power until the inspection is done. With a maximum speed of 0,5 m/s, it requires the power source to run for at least $\frac{600}{0,5} = 20$ minutes.

Data storage

Since the system should be stand alone and waterproof there will be no connection between an operator and the system. Therefore the data which will be generated should be stored onboard. This could be on any kind of small storage device. For example a micro SD card.

Data processing

The system will inspect large sections of the pipe such that it will generate large ammounts of data. The ammount which will be generated is checked later in an experiment. This ammount of data needs to be processed to get a image of the pipes interior and it can be checked if the pipe has any flaws in it. The time needed for the processing will be investigated, wheter a simple laptop could be used for running or that a supercomputer is neccessary.

3 Design

In this chapter the design used for the research will be presented and analysed. The analysis will not only determine if the current device is enough to do the inspection, but also gain insight for improvements.

3.1 Design

The current design of the device for water pipe inspection consists of a structered light vision system, an internal measurement unit(IMU) and a datalogger. These three parts combined makes the laser profiling sensor. This sensor is placed in a waterproof bus which is placed in a pipe inspection gauge (PIG). This PIG is released in the water pipe and with the use of the structered light vision system it is measuring the internal wall of the pipe. An overview of this system is shown in figure 3.1. The data which is generated is stored on micro sd cards onboard. This data can be used for processing offline such that a 3D reconstruction can be made. At the back of the PIG there is a wheel encoder. This wheel encoder measures the odometry of the system and stores this data. In figure 3.2 is the data transfer shown between the different parts. Since only the laser profiling sensor is available for testing and researching, this will be used for this research.

3.1.1 Laser profiling sensor

The laser profiling sensor mentioned above consists of 3 parts. Namely a structured light vision system, an IMU and a datalogger. In figure 3.3 is the design of the sensor shown. The structured light vision system is the one described in the analysis. The laser projection is retrieved by a camera and stored on a micro SD card. The data can then be processed.

An IMU from xsens (xsens, 2016) is used to measure the orientation of the sensor. The IMU measures the orientation of the system in quaternions. These quaternions are then transformed into euler angles to make it more human readable. The orientation of the sensor can be used together with the camera and laser profiler for the inspection and reconstruction of the pipe.

3.2 Analysis on design

The above mentioned design is used as ground point for this research. Since the design is not developed by me but by Cees Trouwborst and Mark Reiling it is being analysed and improved to check whether it is able to do inspections in water pipes. Especially the long runs should be doable. Otherwise a redesign is necessary.

3.2.1 Data to be measured

In previous chapter a list is stated of which data should be measured. In the table below are the ways how to measure the data with the use of the current hardware shown.

Data to be measured	**Way to measure**
Length of the pipe	IMU and wheel encoder
Coordinates of the pipe	IMU and wheel encoder
Angularly displacement	differences in laser projection
Ovality	differences in laser projection
Diameter changes	differences in laser projection
Leakages	high quality imaging with the structured light sensor

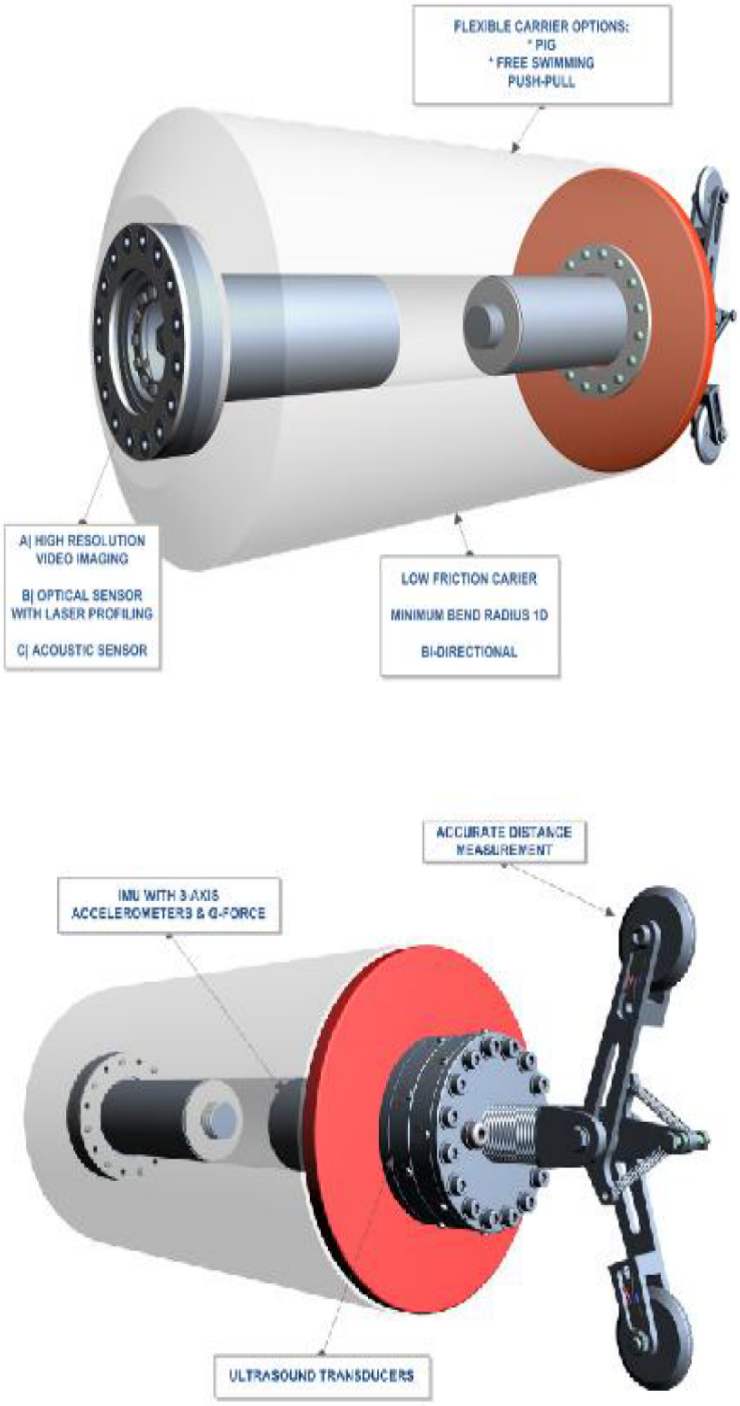

Figure 3.1: Overview of inspection setup (Acquaint, 2017)

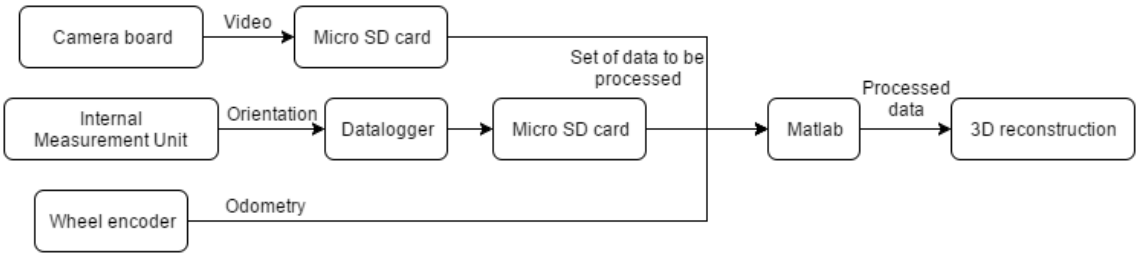

Figure 3.2: Data flow of the sensor and processing

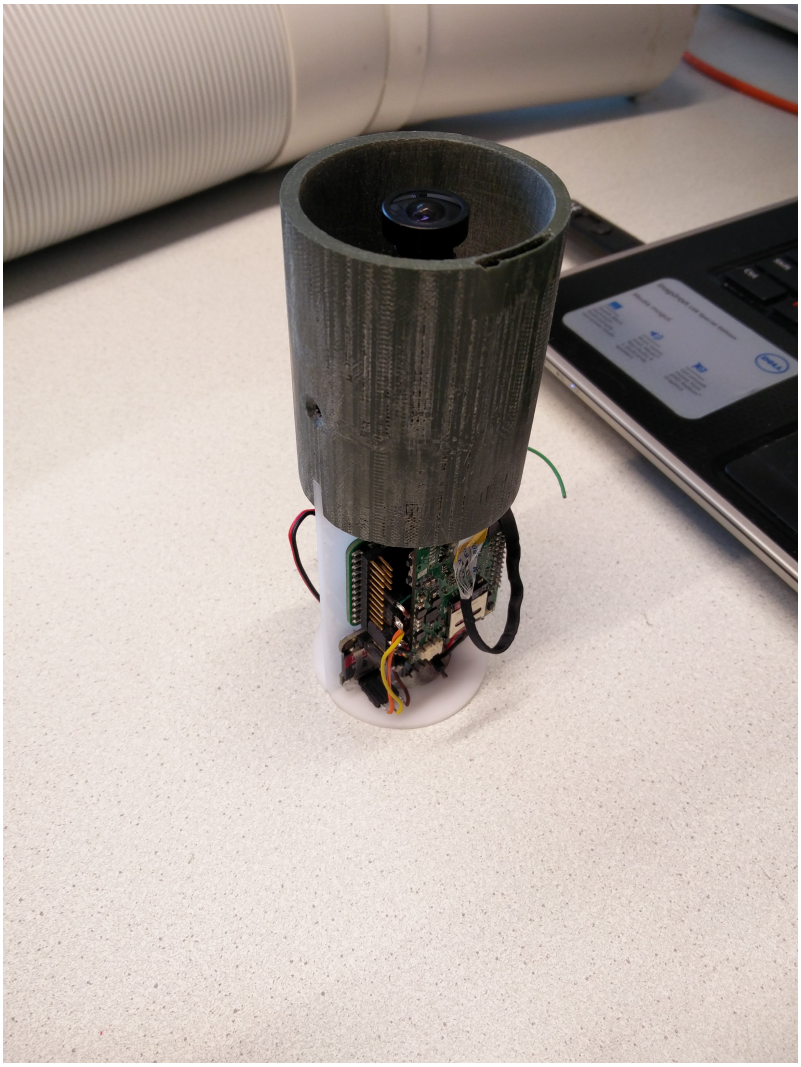

Figure 3.3: Laser profiling sensor

3.2.2 Distance between frames

Because the sensor is for inspection of the pipe and not for navigating through the pipe, it requires the distance between 2 frames to be small. With the current speed of 0,5 m/s and a framerate of 60 this becomes: $\frac{0,5}{60} = 0,0083$ meter. This distance should be lowered by a bit. In the requirements it was set to have a distance of 3 mm between 2 frames. This distance can be achieved by lowering the speed to: $0,003 * 60 = 0,18$ m/s. To measure the distance a wheel encoder is used. The wheel encoder is externally organised and is not available for this research. When this is used and synchronized with the data, the in-depth distance between the frames can be geometrically determined.

3.2.3 Dimensions

The pipe has a diameter of 200 mm such that the PIG has to be smaller than this and flexible to get past corners. The laser profiling sensor has to go into a metal bus, this bus has a inner diameter of 56,5 mm and a length of 151 mm. Such that the design should have these dimensions.

3.2.4 Power source

Since the sensor is stand alone and waterproof, the use of a powerbank is required. The current powerbank has a max capacity of 2600 mAh.

A small measurement is performed to measure the power consumption of the sensor. The powerbank is attached to the sensor with a current meter in between. This measures the current which flows through the entire sensor. This current is 600 mA. With a voltage of 5 V.

The battery will last for at least 4 hours with a consumption of 600 mA. But the current powerbank used is not tested for its protection circuits. Such that when the battery is completely drained, it is not known if the battery is able to use its full capacity again. Also tests has to be performed to know what the behaviour of the powerbank is when it is long exposed to the sensor.

3.2.5 Data storage

The generated data from the IMU and the video camera needs to be stored on the sensor. In the current design this is done for the IMU with a datalogger. The datalogger receives the data from the IMU with the use of the I2C protocol. This data is then stored on a micro sd card. The video camera (Camonetec, 2016) has a pre developed board which contains different output methods. It has a feature for live streaming with the use of a HDMI connection. The video can be stored on a seperate micro sd card, therefore there will be 2 micro sd cards that contains the data of the sensor. The card for the video board is a 32 GB card and for the IMU it is a 16 GB card.

3.2.6 Data processing

After the sensor is finished with the inspection and the data is generated on the micro sd cards they need to be processed to construct a 3D reconstruction of the pipes interior. The 3D reconstruction is done to see if there are any unwanted appearances in the pipe. The inspection length is arround 20 minutes, such that 20 minutes of data needs to be processed.

The video board safes the video in different files of a certain length. This length can be changed in the setup file of the board. When the processing takes too much time these lengths of the videos can be changed to smaller pieces. However the processing takes multiple runs.

During the processing the video file and the logging file are loaded. The video is processed using an algorithm developed by E. Drost (Eric Drost, 2009) and slighty adapted by me for this research. What the algorithm does is retrieve the red laser patern from the image and attach

cartesian coordinates to every retrieved point. These points are then plotted behind each other with a distance of 1 to show the created 3D image. When the odometry is synchronized to the system, it could be used for the distance between frames. The IMU file is connected to the video data by synchronization steps, in the section synchronization below will be explained how this is done. After the synchronization the data is plotted using an algorithm by A. Meer (Alex van der Meer, 2016), which consists of the rotation and translation of the frames.

3.2.7 PIGing order for inspection

The current method of doing an inspection is to put a dummy PIG in the pipe first, followed by the PIG with sensor. The result of this inspection was very muddy water such that the camera could not penetrate the water to see the laser projection. The muddy water was the result of the dummy PIG which scrapes the biofilm from inside of the pipe.

Since there is no real inspection performed during the research all of the below mentioned inspection orders are theoretical and concepts, since it can't be tested if it will work or not.

A method of doing the inspection could be to first doing a clean run with one dummy. When this dummy is through the pipe, the biofilm is already scraped and probably drained. Then do the original setup again to perform the inspection.

A second method could be to do a cleaning run first, this makes sure that the internal wall of the pipe will be clean before the sensor goes through. Another advantage of thiis is that the laser measures the internal wall and not the wall with biofilm and bacteria.

3.2.8 Synchronization

Before the data can be used together they must be synchronized to eachother. The video board captures the video with 60 frames per second. The IMU does this close to 73.

From tests performed with the matlab script and both the data streams, it seems that the video captures the video in 59.9401 frames per second. The framerate of the IMU seems a bit different and not consistent arround 73.

Also the video board start later with capturing a video and the video board needs to be manually stopped otherwise the video will be corrupt. While the IMU stops when the power stops. This has the result that the IMU is always a few seconds ahead of the video and this gets bigger when the time proceeds.

The camera captures the video in time, while the IMU captures the frames with a timestamp. This timestamp could be used to synchronize the data to the video. Different ways can be done to do this. One is to remove the data from the IMU. This can be to check when the video start the data and remove all the IMU frames in front of that time. After that the data at the end of the IMU should be removed. Since the video stops earlier with capturing and the IMU stops with capturing when the power stops there are some frames left at the end of the IMU datastream. These can be removed by estimating the time left and thus frames left. This means that when the above is done properly the beginning and the end are synchronized, but there are still excessive points in between the IMU data. When these points are removed the video and IMU should be synchronized to eachother. Due to time constraints this could not be tested extensively.

4 Experiments

4.1 Experiment; Long run

As a first experiment a long run with the sensor is performed. The sensor is turned on and put on a table. After a few minutes the device is shaked and then again put on a table. The total experiment is about 30 minutes. That is a little bit longer than a real life experiment wich takes about 20 minutes. The experiment is done to see if the power source is capable of keeping the sensor on for a long time. Also with a power meter in between the powerbank and the sensor the ammount of power is used is measured. It can be seen how much data will be generated during that time. Also it can be checked how the IMU and the video are connected and synchronized to eachother, since the two are stored on 2 different micro SD cards. When the device is turned on and put on the table, shaked and then again on the table it can be seen whether the video and imu is synchronized or that one is ahead of the other.

4.1.1 Result

When the long run of 27 minutes is performed there are a bit of surprises. The laser module loses intensity after +- 20 minutes. This is an odd phenomenon but it drops down to a few problems and has to be tested. The laser module gets overheated, which could result in a lesser intensity. Another problem could be that the video board, or the combination of the IMU and datalogger uses too much power over time. A last problem could be that the powerbank could not deliver the ammount of power needed after +-20 minutes. When this is checked and tested it turns out that the powerbank can not deliver the ammount of power needed.

The ammount of data generated for the video in this period of time is 2,5 GB. For the IMU this is 12.000 KB. With the current cards of 32 GB and 16 GB it is enough to have an inspection running of 12 hours.

The power consumption of the sensor is 600 mA with 5 V. The powerbank of 2600 mAh should be able to reach this by ease, however as mentioned above the laser shuts down and does not get overheated. The powerbank delivers at the beginning 4,98 V, but at the end when the lasers intesity is already much lower than at the beginning the powerbank delivers only 4,88 V.

The two micro sd cards are not synchronized with eachother, the video board starts later with capturing while the IMU starts when the power is turned on. To cope with this, it is partly synchronized using the matlab program. As explained in the previous chapter the data should be synchronized to eachother. However there are still some flaws and differences between frames. This strongly depends on where to look for in the datastream. But the most offset is 20 frames off.

In figure 4.1 a datastream from the IMU is shown. The roll, pitch and yaw are converted from quaternions. It is clearly seen that the sensor turns/shakes at certain times. What is odd is the yaw. At the beginning of the data it is seen that the sensor should between timestamp $0.1 \cdot 10^6$ and $0.8 \cdot 10^6$ be turned around the z axis, however at that time the sensor is not moved at all. Due to time constraints it could not be researched why the yaw is continuously moving.

4.2 Experiment; Processing

This experiment is performed to see how much time is needed for processing a certain ammount of data and of the program is working properly. Since the measurement is arround 20 minutes, it is checked if the ammount of processing time is within an achievable time. This experiment only require the matlab program to run with some data. The data used is from a measurement where the sensor is put on the ground and point upwards, a PVC pipe is placed on top of the sensor. The sensor could then perform a measurement in a semi real life envi-

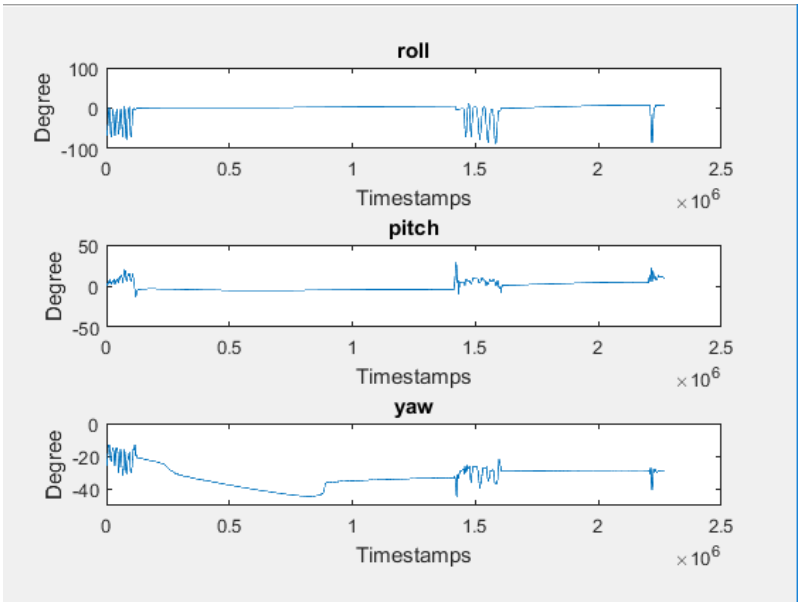

Figure 4.1: Datastream from the IMU

ronment. Since with a bit of preknowledge of the program it take some time to run. Therefore 5 minutes of data will be extracted from the video and processed. The processing will be done on a computer with 16 GB DDR4 RAM and a I5-6600 CPU.

4.2.1 Result

Since the program is completely in matlab the reconstruction is only the retrieved laser projection and not a complete and full reconstruction of the pipe with geometric values. In the work of Alex (Alex van der Meer, 2016) the main focus was on building the 3D reconstruction of the pipe, but in that research the red laser is already retracted from the screen and processed to polarmaps. In this research there is an algorithm used from E. Drost(Eric Drost, 2009) with slight changes by me. This algorithm retrieves the laser from the frames and attach cartesian coordinates to it. These coordinates can be plotted and used to show a reconstruction. There are no geometric values attached to the projection. In figure 4.2 a frame of the laser is shown. This laser is extracted from the screen and further processed until the result in figure 4.3.

The time needed to process is 1:20:20. The max memory usage is when the plots are made, this is 2728 MB for 5 minutes of data. The rest of the processing part it remains the same arround 670 MB. Also the cpu uses only 40% of its capacity to run the program. What can be derived from the matlab data is that for 5 minutes of data there are 17983 frames processed and that the frame rate of the camera is slightly below 60 fps.

4.3 Experiment; Video storage capacity

To check how much storage capacity the micro SD cards should have an experiment is performed. In a previous experiment the ammount of data for a long run was measured. This holds for the IMU since that are only lines and more lines means more data. But the video files can be different. When a video with only a black screen is shown there is much less data required than when the video contains a lot of movement. To perform the experiment there will be different video files generated. One with a lot of movement and one with zero movement and a black screen. Also a video from a previous test where the sensor is in the pipe and only the red laser is shown is taken into account.

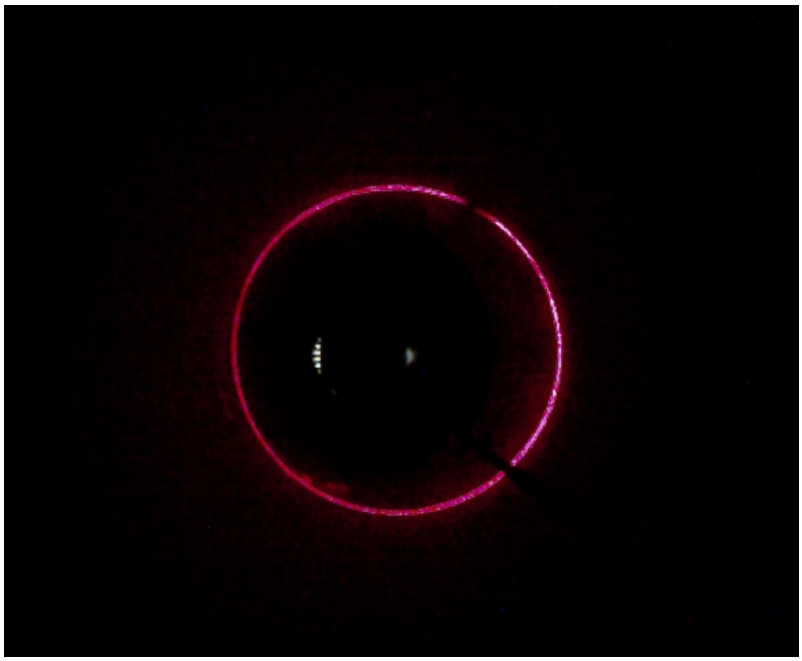

Figure 4.2: Laser projection in the pipe

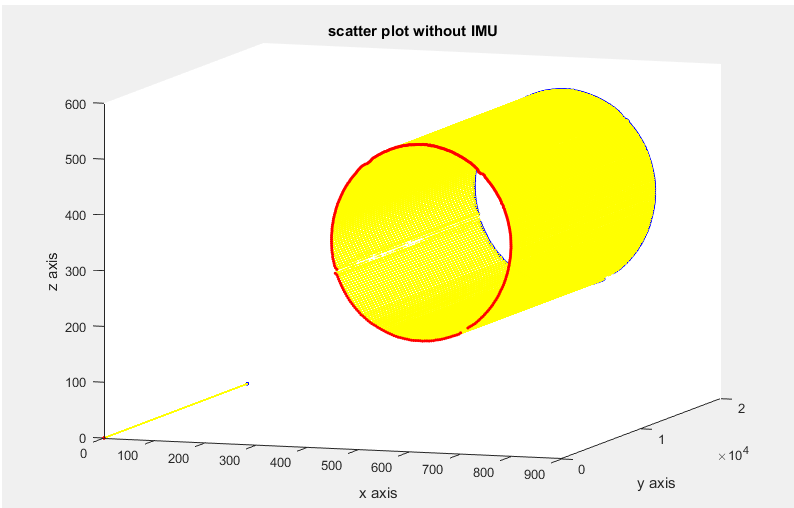

Figure 4.3: Laser projection in the pipe without the use of an IMU

4.3.1 Result

The ammount of data which is generated is different with some videos. For a video where there is a lot of light and movement in front of the camera generates more data than for a movie where there is only darkness and the red laser circle. From different files with a length of 10 minutes the file size various between 200 MB and 900 MB. The video with a size of 200 MB is mostly a black screen. With 900 MB the video consist of a lot of movement. A file where the sensor is in the pipe and only the red laser is visible the ammount of data for a 10 minute video is 360 MB. Since the inspection consists of a setup time, measurement time and time for demounting the PIG, it uses different ammounts of data during these times. The setup and demounting will consist of a bright movie with a lot of movement in front of it. The measurement time on the other hand will be in pitch dark and only a laser projection in front of it. This has to be taken into account when choosing the right storage capacity of the video board.

A weighted average can be performed to guess an estimation of the total data needed for the inspections. In the weighted average below there is the assumption made that the setup and the demounting time are both 5 minutes and that the in pipe time is 20 minutes. Such that 34 % is bright and lots of movement and 66 % is pitch dark and only the laser.

When a weighted average of the data is performed the ammount of data which is generated per 10 minutes is: $\frac{900 \cdot 17 + 360 \cdot 66 + 900 \cdot 17}{100} = 543,6$. Per second this is $\frac{543.6 \cdot 60}{600} = 54,36$ MB/s.

4.4 Experiment; 200 mm PVC pipe

This experiment is performed to simulate the behaviour of the sensor in a 200 mm PVC pipe. In a real inspection the pipe has be able to inspect drinking pipes with a diameter of 200 mm. The laser profiling sensor will go into a PIG such that the sensor is centered in the middle of the pipe.

That the sensor is in the middle of the pipe is an assumption. The imu measures the orientation of the sensor in the pipe during the inspection, but since the orientation of the pipe is not known it is not sure whether the pipe is excatly alligned with the sensor.

In the processing experiment it is already a bit explained how to do it. But with this experiment the ammount of data which is generated by the sensor is checked. Since the video data differs with how much light and movement is in the videos, it will be simulated in this experiment with a setup, measurement and demounting part. The complete measurement will be arround 20 minutes. The setup of the experiment is shown in the picture below. The sensor is put on the ground with a 200 mm diameter PVC pipe on top of it. A piece of paper is placed on top of the PVC pipe to make it dark inside. After a certain time the sensor and pipe are tilt to see the behaviour of the sensor during changes in orientation.

In figures 4.4 and 4.5 is the measurement setup shown. The laser is put on the ground with the lasers direction upwards and a 200 mm diameter PVC pipe is placed on top of the laser.

4.4.1 result

A frame captured by the camera within the pipe is shown in figure 4.2. The time it takes for 5 minutes of data to process is 1:20:20. For a 20 minutes inspection this will be 4 times as long so: 5:21:20. This is quite some time to process. The result of the processing is shown in figure 4.3. This is the result without the use of the IMU, the space between frames along the y axis is 1. In figure 4.6 the result is shown with IMU alongside with the IMU values. What can be noticed in the reconstruction figure with IMU values is that is a very unclear image and less detail is taken out of the image. Also the yaw is constantly changing while the sensor is placed on the ground. It is unclear why it does this and has to be further researched.

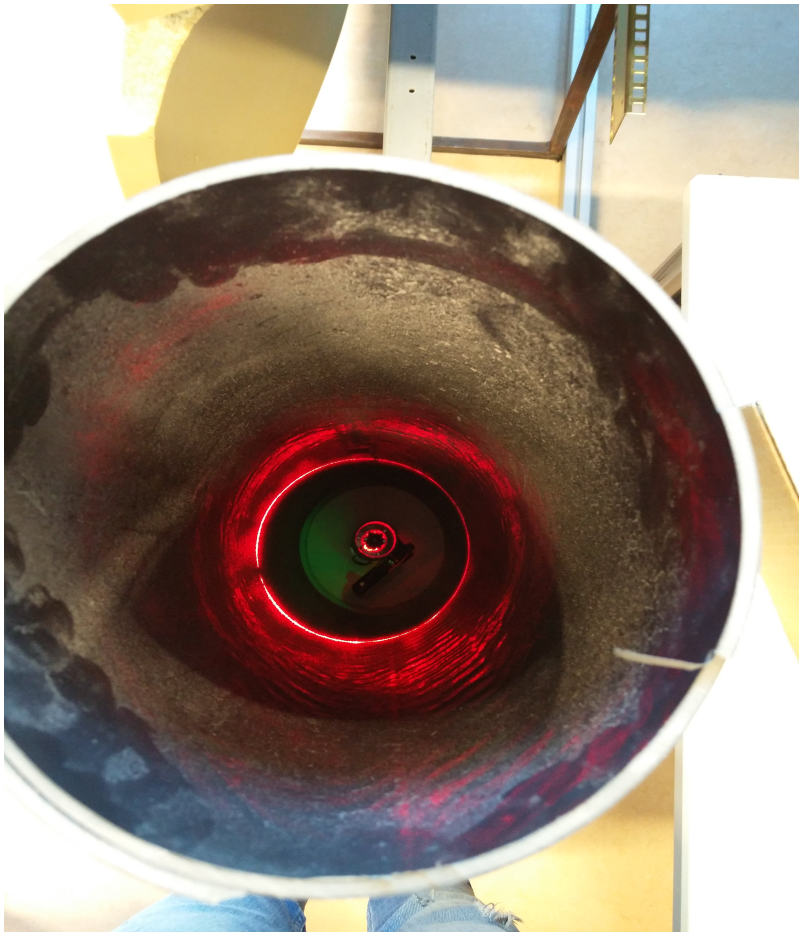

Figure 4.4: Measurement setup

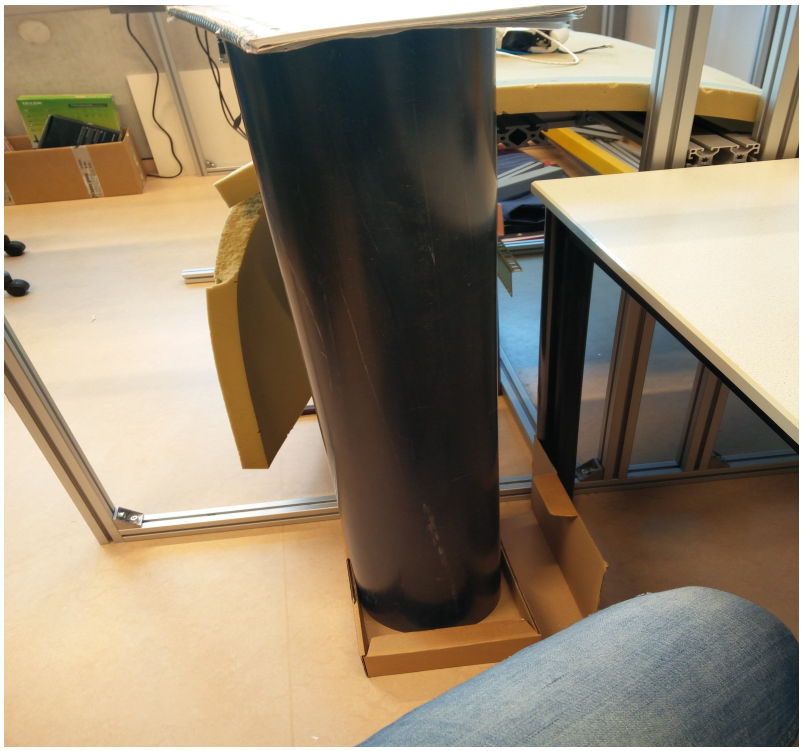

Figure 4.5: Measurement setup

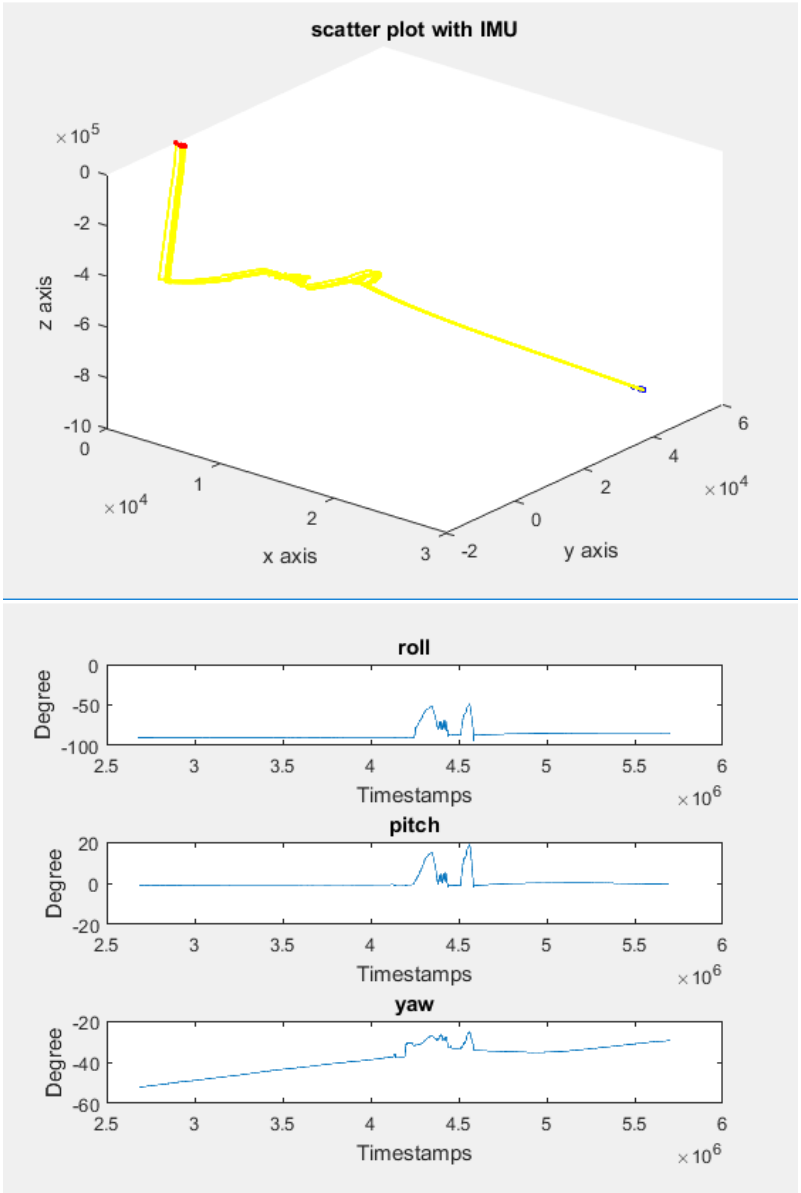

Figure 4.6: Above is the 3d image and below are the corresponding IMU values

5 Conclusion and Recommendations

5.1 Conclusion

As a whole the laser profiling sensor is a good start for the in-pipe inspection of water pipes. The realized design is capable of doing inspections, saving the generated data onboard and reconstruct the pipe offline. The sensor measures the inside wall of the pipe with the laser and have the orientation connect with that. The performed experiments shows that it is capable of performing an inspection for a long time. However on the long run it is seen that the laser loses intensity, this is due to the power source which loses power over time. Also the data is not properly synchronized. The biggest error was 20 frames difference between the video and IMU. It is important to have this fully synchronized for a proper reconstruction. Another flaw in the IMU data is that it is not measuring the orientation properly since the yaw has a constant change when the sensor is completely still. The cause of this can be due to the setup of the IMU. It could be that the IMU needs some time and movement to orient itself. But this has to be tested which could not be done due to time constraints. The reconstruction of the pipe gives a nice result, where the inner wall of the pipe is shown. However there are no geometric values attached to the image. This is also due to time constraints. Since the odometry is externally organised it was not researched. But when the odometry can be synchronized with the video and IMU it can be used for in-depth geometry of the pipe. When there is a flaw detected, it can be found by the distance measured by the odometry.

5.2 Recommendations

As mentioned in the conclusion there are a few problems which are not yet fixed and what can be improved in future research. The first is the power source, the power source can not deliver enough power to have the full sensor working during the long run. When it is fully loaded it delivers enough power, but loses its intensity after a certain time. What can be done is to design a more constant power source or have a decent powerbank which can deliver a constant power. Another problem is the orientation of the sensor vs. the pipes orientation. The IMU measures the orientation of the sensor, but the pipes orientation is not measured. It is assumed that the sensor is fully alligned with the pipes, but this is not always the case during the inspection. One side of the PIG could lag and will cause that the sensor is still in the middle, but not alligned in the same orientation as the pipe and will give false information about the pipe. A third problem is synchronization. The data is not properly synchronized. The current way of doing the synchronization is by recognizing movements in the video and IMU data stream. This way involves a lot of guessing and is not robust for different files. For fully synchronized data there should be a way of starting and stopping both the video and IMU at the same time. When the power kicks in the video needs to load for a while before it captures the video. If this is always the same ammount of time and the IMU uses the same ammount of time before the logging begins, it can be synchronized to each other. The stop time is hard time to find for both because the video board stops with a stop butten and the IMU when the power shuts down. When the stop button and the datalogger can be connected with eachother, it could be a way of stopping at the same time. Another way of synchronizing the data is to make one device the master of them all. It can be to attach a microcontroller which is the master of the other devices. A fourth problem is that the reconstruction gives no geometric values to the images. The odometry is not used in this research, since it was not available. But when the odometry is connected, calibrated and synchronized with the sensor it could be used to determine the distance between the different frames and attach geometric values to the in-depth distance. What also needs to be done is to assign geometric values to the diameter from the retrieved images. This can be done by calibrating the sensor.

A The used Matlab code

In this appendix is the matlab code used for processing the data explained. Not all the files used are stated here, only the important ones. The main script is the most important of them all and is explained thoroughly. The other scripts are help scripts for the processing, they are just stated in here since the algorithm is from E. Drost (Eric Drost, 2009).

A.1 Main script

For the processing multiple scripts are used. The first is the main script with al the main processing and plotting. The scripts begin with a setup where the video and log are set, also the start- and stop time are set. The main loop start at the while loop. In that loop the frames are loaded and processed using the algorithm of E. Drost frame by frame, until the stop time is reached. The processing involves prefiltering, segmentation of the image and extraction of the found objects. The objects are then averaged and cartesian coordinates are attached to these points. After the main processing the IMU logger file is loaded and excessive data is removed until it is just as big as the video data. After this the Roll, Pitch and Yaw is calculated for human readable orientation changes and plotted. Then the 3d reconstruction is plotted without and with IMU values. The for loop with the IMU contains the rotation and translation. After that is done the frames are plotted.

```
%% main script to run
% this is a single script with all the functions in one loop, since this
% reduces the memory usage.

%% setup
% In this part the memory is cleared and the changes are made. Different
% video files, start and stop times etc..
clc
clear all
close all

colormap(gray);
flag = 1;
filename = 'FILE0004.MP4';
logname = 'LOG0004.TXT';
start_time = 0*60 + 30; % minutes*60 + seconds
stop_time = 0*60+ 30.1;  % minutes*60 + seconds

%% file loading and processing processing
% Load the video and assign the total number of frames, start- stopframe
% and the number of frames needed for the processing
v = VideoReader(filename);
TotalNumOfFrames = v.NumberOfFrames;
FrameRate = v.FrameRate;
StartFrame = ceil(start_time * FrameRate);
StopFrame = ceil(stop_time * FrameRate);
NumFrames = StopFrame - StartFrame;
clear v

% load video from start time
v = VideoReader(filename);
v.CurrentTime = start_time;

mov = struct('cdata',zeros(720, 1280,3, 'uint8'));

x_pos = zeros(1, 360);
```

```
z_pos = zeros(1, 360);
k = 1;
% The main processing happens in this while loop
while v.CurrentTime <= stop_time
% load video in gray
mov(k).cdata = readFrame(v);
im = rgb2gray(mov(k).cdata);

% process video, processing is done by Eric Drost (2009) with slight
% changes by Dylan Kamermans (2017). The files: preprocess_image,
% segment_image, extract_objects and find_curve_mod are made by Eric.
% In the preprocess_image the
% filters are applied and is the frame being readied for the
% segmentation where all the colors in the image are found. This needs
% the frame to be dark with a red laser. In extract_objects are the
% objects which are segmented retrieved. After that the retrieved
% objects are put in an empty frame where there is now only the laser.
impp = preprocess_image(im);
[objects, n_objects] = segment_image(impp);
[O_laser, O_pipe, area] = extract_objects(im, objects, n_objects);
laser = im(:,:) .* uint8(O_laser>0);
[r c] = size(laser);
%     laser = im(:,:) .* uint8(O_laser>0);
[e f] = find(laser);
cy(1) = min(e);
cy(2) = max(e);
cx(1) = min(f);
cx(2) = max(f);
ccx = round((cx(1)+cx(2))/2);
ccy = round((cy(1)+cy(2))/2);
rc = sqrt((ccx-ccx)^2+(cy(1)-ccy)^2);
rc = rc-20;

% since the extract_objects retrieve large objects anywhere on the
% frame, it is removed below. Every large object smaller than the
% radius is removed
for i = 1:r
for j =1:c
if rc > sqrt((j-ccx)^2+(i-ccy)^2)
laser(i,j) = 0;
end
end
end
% attach coordinates to the points in the frames and clear all the
% unnecessary data
[pcd theta] = find_curve_mod(laser,ccy,ccx);
clear impp
clear objects
clear n_objects
clear im
clear O_laser
clear O_pipe
clear laser
clear theta

% add padding to have all pcd consists of 360 points
x = pcd(1,:);
z = pcd(2,:);
if length(x) < 360
x = x';
x = padarray(x, 360-length(x), 'post');
x = x';
z = z';
```

```
z = padarray(z, 360-length(z), 'post');
z = z';
end
x_pos(k,:) = x;
z_pos(k,:) = z;

k = k + 1;
clear mov
end

% below is the odometry, but during this research (2017) there is no data
% available thus there is 1 between every frame.
y_pos = zeros(k-1,360);
for i = 2:k-1
y_pos(i,:) = y_pos(i-1,:) + 1;
end

%% load logger file
% in here the logging file is loaded and adjusted such that it matches the
% video data.
formatSpec = '%f%f%f%f%f%f%f%f%f%f%f%f%f';
T = readtable(logname,'Delimiter',',','Format',formatSpec);
m = table2array(T);
M = (m(3:end,1:end));
clear m
clear T

% delete unneaded frames
x = 398;
set = M(x,3);
M(:,3) = M(:,3)-set;
j = 1;
for i = 1:length(M(:,3))
if M(i,3) >= 0
N(j,:) = M(i,:);
j = j+1;
end
end
N = N(1:end-100,:);
lengte = length(N);

number_of_removals = lengte-TotalNumOfFrames;
intermediate_states = lengte/number_of_removals;

removalList = round(1:intermediate_states:lengte);
N(removalList,:) = [];

% reduce IMU to start time and stop time
N = N(StartFrame:StopFrame,:);
time = N(:,3);
quat = N(:,4:7);

% calculate the roll pitch and yaw
[roll pitch yaw] = calculateRollPitchYaw(time,quat);

%% plot roll pitch and yaw
% plot the roll pitch and yaw for human readable orientation changes
figure(1)
subplot(3,1,1)
plot(time, roll);
title('roll');
xlabel('Timestamps');
ylabel('Degree');
```

```
subplot(3,1,2)
plot(time, pitch);
title('pitch');
xlabel('Timestamps');
ylabel('Degree');
subplot(3,1,3)
plot(time, yaw);
title('yaw');
xlabel('Timestamps');
ylabel('Degree');

%% plot 3d image
figure(2)
hold on
for i = 1:k-1
if i == 1
color = 'red';
clear flag
elseif i == k-1
color = 'blue';
clear flag
else
color = 'yellow';
flag = 1;
end
if flag == 1
scatter3(x_pos(i,:),y_pos(i,:),z_pos(i,:),flag,'.',color);
else
scatter3(x_pos(i,:),y_pos(i,:),z_pos(i,:),'.',color);
end
cxmin(i)= min(nonzeros(x_pos(i,:)));
cxmax(i) = max(nonzeros(x_pos(i,:)));
czmin(i)= min(nonzeros(z_pos(i,:)));
czmax(i) = max(nonzeros(z_pos(i,:)));
centrumx(i) = (cxmax(i)+cxmin(i))/2;
centrumy(i) = y_pos(i);
centrumz(i) = (czmax(i)+czmin(i))/2;
scatter3(centrumx(i),centrumy(i),centrumz(i),'.','black');
end
title('scatter plot without IMU');
xlabel('x axis');
ylabel('y axis');
zlabel('z axis');
hold off

Pt = [[]; []; [];];
positioncurrent = [0, 0, 0];
posHist = [positioncurrent(1); positioncurrent(2); positioncurrent(3);];
%% plot 3d image with imu values
% rotation and translation is developed by Alex van der Meer (2016) and
% used in here
figure(3)
hold on
for i = 1:k-1
R = quat2rotm(quat(i,:));
Pframe = [x_pos(i,:); y_pos(i,:); z_pos(i,:);];
Pframe = R*Pframe;

movementvector = [0; 50; 0;];
Rotate = quat2rotm(quat(i,:));
movementvector = Rotate*movementvector;

positioncurrent = [positioncurrent(1) + movementvector(1,1),
```

```
positioncurrent(2) + movementvector(2,1),
positioncurrent(3) + movementvector(3,1)];

PframeHomo = [Pframe(1,:);Pframe(2,:);Pframe(3,:);Pframe(1,:)*0 + 1;];
Tr = [1 0 0 positioncurrent(1);
0 1 0 positioncurrent(2);
0 0 1 positioncurrent(3);
0 0 0 1 ];
PframeHomo = Tr*PframeHomo;
Pt = PframeHomo;

if i == 1
color = 'red';
clear flag
elseif i == k-1
color = 'blue';
clear flag
else
color = 'yellow';
flag = 1;
end
if flag == 1
scatter3(Pt(1,:),Pt(2,:),Pt(3,:),flag,'.',color);

else
scatter3(Pt(1,:),Pt(2,:),Pt(3,:),'.',color);

end
end
title('scatter plot with IMU');
xlabel('x axis');
ylabel('y axis');
zlabel('z axis');
hold off
```

A.2 Processing scripts

Below are the 4 processing scripts shown. The first script is the filtering, the second the segmentation, third is the extraction and the last is assigning cartesian coordinates to the found points. They are just stated here for reading, the details are found in the work of E. Drost (Eric Drost, 2009).

```
function impp = preprocess_image(im)
% apply filters on the frame
img = im;
imf = medfilt2(img,[4,4]);
imth = imf > 50;
SE = strel('square',8);
impp = imdilate(imth,SE);
```

```
function [objects n_objects] = segment_image(im)

% Connected Component Labeling:
%   1. scan image to find 4-connected pixels
%   2. Sort equivalent label pairs into equivalence classes
%   3. Re-label image


% SCAN IMAGE FOR CONNECTED pixels and label them
```

```
%
% if both neighbors are 0 assign new label
% if left neighbor is 1 assign left label
% if upper neighbor is 1 assign upper label
% if both neighbors are 1 assign upper label and make a note of equiv

[h b] = size(im);        % pad image with zeros
im_pad = zeros(h+1, b+1);
im_pad(2:h+1, 2:b+1) = im;

L = zeros(size(im));     % Labels
current_label = 0;     % current label
eq_cnt = 0;      % equivalence counter

for r = 2:h+1
for c = 2:b+1
if im_pad(r,c) == 1
if im_pad(r-1,c) == 0 && im_pad(r,c-1) == 0
current_label = current_label+1;
L(r-1,c-1) = current_label;
elseif im_pad(r-1,c) == 0 && im_pad(r,c-1) == 1
L(r-1,c-1) = L(r-1,c-2);
elseif im_pad(r-1,c) == 1 && im_pad(r,c-1) == 0
L(r-1,c-1) = L(r-2,c-1);
elseif im_pad(r,c-1) == 1 && im_pad(r,c-1) == 1
L(r-1,c-1) = L(r-2,c-1);
if L(r-2,c-1) ~= L(r-1,c-2)   % make note of equivalence
eq_cnt = eq_cnt+1;
E(eq_cnt,:) = [r c L(r-1,c-2) L(r-2,c-1)];  % Equivalences
end
end
end
end
end

% GROUP EQUIVALENT LABELS
%
% for each entry in E
% if [0] entry in _class_ -> create new class
% if [1] entry in _class_ -> update classes
% if [2] {different} entries in _class_ -> regroup classes
if eq_cnt ~= 0
next_class = 2;
class(1,:) = E(1,3:4);
for e = 2:eq_cnt
[c1 tmp] = find(E(e,3) == class);
[c2 tmp] = find(E(e,4) == class);
if isempty(c1) && isempty(c2)
class(next_class,1:2) = E(e,3:4);
next_class = next_class + 1;
elseif ~isempty(c1) && isempty(c2)
lc1 = length(find(class(c1,:)));
class(c1, 1:lc1+1) = [class(c1,1:lc1) E(e,4)];
elseif isempty(c1) && ~isempty(c2)
lc2 = length(find(class(c2,:)));
class(c2, 1:lc2+1) = [class(c2,1:lc2) E(e,3)];
elseif ~isempty(c1) && ~isempty(c2) && c1~=c2
[cmin imin] = min([c1,c2]);
[cmax imax] = max([c1,c2]);
lc = [length(find(class(c1,:))) length(find(class(c2,:)))];
class(min(c1,c2), 1:lc(1)+lc(2)) = [class(cmin, 1:lc(imin))
                                    class(cmax, 1:lc(imax))];
class(cmax:size(class,1)-1,:) = class(cmax+1:size(class,1),:);
```

```
class = class(1:size(class,1)-1,:);
next_class = next_class - 1;
end
end
end

% REASSIGN LABELS
%
% if L(r,c) not in LU_Table and not in class
%   -> update LU_table (using L) and L
% if L(r,c) not in LU_Table but in class
%   -> update LU_table (using class) and L

if eq_cnt ~= 0
label = 1;
n_objects = current_label - length(find(unique(class))) + size(class,1);
LU_table = zeros(n_objects, size(class,2));
for r=1:size(L,1)
for c=1:size(L,2)
if L(r,c) ~= 0
[lb tmp] = find(L(r,c) == LU_table);    % if not in LU_table
if isempty(lb)
[lb tmp] = find(L(r,c) == class);   % if not in equiv classes
if isempty(lb)
LU_table(label,1) = L(r,c);     % create new entry in LU_table
L(r,c) = label;                 % update labels
label = label + 1;
else LU_table(label,:) = class(lb,:); % if not in LU_table but label IN class
L(r,c) = label;             % update LU_table and labels
label = label + 1;
end
else L(r,c) = lb;           % if in LU-table assign label
end
end
end
end
end

objects = L;
```

```
function [O_laser, O_pipe, area] = extract_objects(imrgb, objects,
                                                    n_objects);

[h b] = size(objects);
center = [(h+1)/2 (b+1)/2];

% determine area of object
area = 0;
for i=1:n_objects
area(i) = sum(sum((objects == i)));
end

% determine location of object
location = 0;
for i = 1:n_objects
[r c] = find(objects == i);
location(i,1) = sum(r)/area(i);
location(i,2) = sum(c)/area(i);
end

% The loops below are made by Dylan Kamermans (2017), These determine the
```

```
% objects of a certain area size and pass them through.

[maxarea label_max] = max(area);
O_laser = zeros(size(objects));
if maxarea > 13000
O_laser = objects .* double(objects == label_max);
else
for i = 1:n_objects
if area(i) > 740
O_laser = O_laser + objects .* double(objects == i);
end
end
end


O_pipe = zeros(size(objects));
% Below is unused
% Dmax = 50;
% define pipe end as objects around center other than laser and white color
% for label = 1:n_objects
%     if label ~= label_max
%         DTC = ((location(label,1)-center(1))^2 +
%               (location(label,2)-center(2))^2)^(1/2);
%         if DTC < 200
%             R = imrgb(round(location(label,1)), round(location(label,2)));
%             G = imrgb(round(location(label,1)), round(location(label,2)),2);
%             B = imrgb(round(location(label,1)), round(location(label,2)),3);
%             if (R > 100) % && G > 100 && B > 100)
%                 O_pipe = O_pipe + objects .* double(objects == label);
%             end
%         elseif DTC <500
%             O_laser = O_laser + objects .* double(objects == label);
%         end
%     end
% end




% O_laser = zeros(size(objects));
% [h b] = size(objects);
% center = [(h+1)/2 (b+1)/2];
% Ath = 50; Dmin = 30; Dmax = 400;
% for label = 1:n_objects
%    DTC = ((location(label,1)-center(1))^2 + (location(label,2)-
%          center(2))^2)^(1/2);
%    if area(label) > Ath && DTC > Dmin && DTC < Dmax
%         O_laser = O_laser + objects .* double(objects == label);
%     end
% end
```

```
function [points theta] = find_curve_mod(laser, h, w)

[r,c] = find(laser);
np = length(r);
center = [h w]; %[491 661]; % Epipole in image plane
polar_coord = zeros(np,3); %[theta radius intensity]

p = 1;
for i=1:np
```

```
ri = r(i); ci = c(i);

if ri <= center(1) && ci > center(2)
polar_coord(p,1) = atan((center(1)-ri)/(ci-center(2)));
elseif ri < center(1) && ci <= center(2)
polar_coord(p,1) = 1/2*pi + atan((center(2)-ci)/(center(1)-ri));
elseif ri >= center(1) && ci < center(2)
polar_coord(p,1) = pi + atan((ri-center(1))/(center(2)-ci));
elseif ri > center(1) && ci >= center(2)
polar_coord(p,1) = 3/2*pi + atan((ci-center(2))/(ri-center(1)));
end

polar_coord(p,2) = ((ri-center(1))^2 + (ci-center(2))^2)^(1/2);
polar_coord(p,3) = laser(ri,ci);
p=p+1;

end

% calculate radius at sub-pixel location
n = 1; nn = 1;
noangle = 0;
curve = [0 0 0];
for t = 0 : 2*pi/360 : 2*pi - 2*pi/360;
i = find(polar_coord(:,1) >= t & polar_coord(:,1) < t+2*pi/360);
if length(i) > 1 %~isempty(i)
u = unique(polar_coord(i,2));
ud = diff(u);
if max(ud) < 10
radius = (polar_coord(i,2)' * polar_coord(i,3)) / ...
sum(sum(polar_coord(i,3)));
curve(n,1:2) = [t radius];
n=n+1;
else
[junk index] = max(ud);
TH = (u(index)+u(index+1))/2;
i1 = find(polar_coord(i,2) < TH);
i2 = find(polar_coord(i,2) > TH);
radius1 = (polar_coord(i(i1),2)' * polar_coord(i1,3)) / ...
sum(sum(polar_coord(i1,3)));
radius2 = (polar_coord(i(i2),2)' * polar_coord(i2,3)) / ...
sum(sum(polar_coord(i2,3)));
curve(n,1:3) = [t radius1 radius2];
n=n+1;
end

else noangle(nn) = t; % Find gap
nn=nn+1;
end
end

u = center(2) + curve(:,2) .* cos(curve(:,1));
v = center(1) - curve(:,2) .* sin(curve(:,1));
points = [u'; v'];
theta = curve(:,1)';


% % sort the curve
% amin = min(noangle); amax = max(noangle);
% index1 = find(curve(:,1) > amax);
% index2 = find(curve(:,1) < amin);
% curve1 = curve(index1,1:3);
% curve2 = curve(index2,1:3);
% curve = [curve1; curve2];
```

```
%
% % Points
% u = center(2) + curve(:,2) .* cos(curve(:,1));
% v = center(1) - curve(:,2) .* sin(curve(:,1));
% points_r = [u'; v'];
% [index junk] = find(curve(:,3) > 0);
% n_dpoints = length(index);
% u = center(2) + curve(index,3) .* cos(curve(index,1));
% v = center(1) - curve(index,3) .* sin(curve(index,1));
% points = [points_r [u';v'] ];
```

B Experimental procedure

For the experimental procedure not much has to be performed. The current setup has already the files needed for experiments uploaded. The only thing to do is to clear the sd card from excessive log files. Be carefull what is deleted on the video sd card since this contains the setup file. When this is deleted there is no harm done because the video starts in its default mode when there is no file present. There are not much things what have to be changed in the file. What could be changed is the length in which the camera captures videos, but that is up to what the next person fancies to have. The datalogger has an arduino compatible code to fetch the data from the IMU. Furthermore when the sd cards are back into place the powerbank can be attached. After the experiment is done, the data is found on the sd cards. A .txt on the imu and a .mp4 on the video sd cards. The matlab script main_script is used to process the data. In this script the setup has to be changed to the current filenames and the start and stop times has to be filled in.

```
filename = 'FILE0004.MP4';
logname = 'LOG0004.TXT';
start_time = 0*60 + 30; % minutes*60 + seconds
stop_time = 0*60+ 30.1;  % minutes*60 + seconds
```

Bibliography

Acquaint (2017).
https://www.acquaint.eu/

Alex van der Meer (2016), *Mapping the underground*, Bachelor's thesis, University of Twente.

Andrew Johnson and James Burton (2017).
http://www.independent.co.uk/news/uk/home-news/water-torture-3300000000-litres-are-lost-every-single-day-through-leakage-2034 html

Breivoll inspection technologies (2016).
http://www.breivoll.no/

Camonetec (2016).
http://www.camonetec.com/de/produkte/kamerasysteme/flycamone-eco-1080p.html

Eric Drost (2009), *Measurment system for pipe profiling*, Master's thesis, University of Twente.

Frey, C. W. (2008), Rotating Optical Geometry Sensor for fresh water pipe inspection, in *2008 IEEE Sensors*, pp. 337–340, ISSN 1930-0395, doi:10.1109/ICSENS.2008.4716450.

Mark Reiling (2014), *Implementation of a Monocular Structured Light Vision System for Pipe Inspection Robot PIRATE*, Master's thesis, University of Twente.

Poozesh, S., M. Mehrandezh, H. Najjaran and R. Paranjape (2007), Design and Development of a Smart Vehicle for Inspection of In-Service Water Mains, in *2007 Canadian Conference on Electrical and Computer Engineering*, pp. 1203–1206, ISSN 0840-7789, doi:10.1109/CCECE.2007.307.

south east services (2017).
http://www.southeastsvc.com/our-services/laser-profiling/

Walter van der Schoot (2015), *Laser profiling of sewer pipes*, Master's thesis, Delft University of Technology.

xsens (2016).
https://www.xsens.com/products/mti-1-series/