Rob 195 – Automated Object Detection in a Collaborative Robot Workspace

Preliminary Studies of Bachelor Thesis Rob195

Degree course: Micro and Medical Technologies

Author: Aeschlimann Dario

Tutors: Rochat Sarah, Gruener Gabriel

Constituent: AHB Biel

Experts: Aigner Nikita

Date: 06.05.2019

Management Summary

The present document is the preliminary study to the Bachelor Thesis "Rob 195 - Automated Object Detection in a Collaborative Robot Workspace" in the degree course Micro and Medical Technologies at the Bern University of Applied Sciences. The purpose of the Bachelor Thesis is to improve the safety in collaborative robotic systems by avoiding collisions between the robot and objects in its workspace. This should be realized by integrating a vision system to detect objects in the planned trajectories of the robot. The input on this vision system is delivered by a commercially available camera which delivers its gathered data via point cloud data to the system. The system then compares the workspace and its occupied parts with the planned motion of the robot. The robot, used for this project is the Fanuc CR35iA Cobot which is located at the the department AHB of the Bern University of Applied Sciences in Biel. Using the data which the system shall provide, the Robot should calculate an alternative route to its initial end coordinates without colliding with the object which was detected in the workspace. In addition, a safety stop, based on a minimal distance to an object shall be integrated in the system.

During the execution of the preliminary study, the following tasks have been performed:

- Time planning of the preliminary study.
- Literature review.
- Familiarize and setup of programming environment.
- Selection and commission of parts for test-setup.
- Familiarize with the robot and its interface.
- Project Plan based on milestones for the Bachelor Thesis.

As the initial planned GPU-Voxels algorithm did not work, the author therefore decided to proceed with a point cloud based system. For the communication between Robot and System the TCP/IP Protocol has been chosen.

To realize the goals which are set for the Bachelor Thesis, the following milestones have been defined:

- M1: Running programming environment

- M2: Robot communication

- M3: Scanning of workspace

- M4: Simple 2D Motion Planning. End-effector only

- M5: Simple 3D Motion Planning. End-effector only

- M6: Collision detection for whole robot model

- M7: Collision detection in a dynamic workspace

Milestone 1 to 4 define the bare minimum of requirements which the Bachelor Thesis has to fulfill. The fulfillment of all 7 Milestones would indicate the best possible solution within the context of this Thesis.

Contents

1 Introduction

Collaborative robots are meant to be flexible and easy to be reprogrammed so that they can be used efficiently in the modern industrial environment. One of the most important aspects for collaborative robots is safety. The robot shall not cause any harm to people or material. Currently, most Systems rely on force or torque sensors to stop any motion if a collision is taking place. This means, that the robot motion only stops when a collision already happened. This approach works reasonably well for human-machine collisions, but in a collaborative situation, the robot acts within a dynamic workspace. Objects like containers with liquids or heavy and unstable objects may obstruct the workspace. These objects may be tipped over by the robot without triggering a halt based on the force sensors, which could pose a health risk. In addition, halts caused ba the force sensors cause downtime in the production process which leads to higher production costs and longer production times, which companies like to avoid. The goal of this thesis is to develop a vision system, which detects in real-time any occupied area in the workspace, that means people or objects which are within reach of the robot, and adapts the robots trajectories to avoid any collisions, thus providing an extra layer of safety. This would allow the robot to work in a modifiable workspace together with a human and to adapt his movements according to the human ones [9].

2 Specifications

2.1 Purpose

The purpose of this project, consisting of preliminary studies and bachelor thesis, is to add an additional layer of safety to collaborative robotic systems, which should detect and avoid collisions. Currently most systems use force and torque sensors which are telling the robot to stop its motion, as soon as a collision occurs. This presumes a given amount of force or torque to lead the robot to an action. In term of the collaboration between human and machine, this principle fits the given needs. But when it comes to lighter objects like containers with (sometimes toxic) liquids or other unstable objects, the robot will not stop its motion because the sensor did not get a measurement within the specified force range. Systems with force sensors, only react when a certain force threshold is reached. Collisions with light objects may not reach the force threshold and therefor may not stop the robot motion. To avoid such cases, a vision system shall monitor the workspace and detect objects, then adapt the robots trajectories according to the position of obstacles to avoid a collision.

2.2 Description and goals

The main goal is to integrate a vision system to detect any unexpected temporary objects in real-time in a robot workspace to add an extra layer of safety for the employee and to reduce downtime within the production process. The work shall be performed in a collaboration with the local department of Bern University of Applied Science for architecture, wood and civil engineering, AHB in Biel, using their Fanuc CR35iA Cobot. The CR35iA is currently the biggest collaborative robot available on the market. It has a maximum payload of 35kg and a reach of 1813mm. The

robot is placed in a factory like building, surrounded with other robots and machines, therefore representing an industrial environment, similar to the foreseen use in the industry. The detection of obstacles inside the robots workspace shall be realized by using 3D sensors. The data provided by these sensors will be processed by the GPU-Voxels Algorithm into voxelized maps. Two maps shall be implemented, one for the workspace and one for the robot model and its trajectories. Using the robots joint speeds, trajectories may be predicted to a certain point and can be integrated into the robot model map. By comparing the maps with each other, upcoming collisions can be predicted. Upon detecting collisions, the robot trajectories shall be adapted to avoid the previously predicted collision, without stopping the robots motion to calculate an alternative route for the robots motion. For extra credit, the system shall be demonstrated in a dynamic workspace, with human detection integrated. The Bern University of Applied Science executes multiple Bachelor Theses in robotics, and specifically for collaborative tasks, a group of students working in this field has been build. Teamwork is highly required and recommended to avoid making the same mistakes and to be sure to share knowledge and progress. Tasks which can be shared or even adapted offer a great advantage to all students in this group. In addition, it helps the experts to coordinate the work which is done.

2.3 System Specifications

Based on the content in section 2.2, Description and goals, a system specification has been elaborated by the author.

2.3.1 System Functions

Object detection

The System shall detect Objects inside the robot workspace using one or more stereo cameras. Possible cameras are the Microsoft Kinect and the Asus Xtion ProLite, which are available at BFH TI for this Thesis. A third possible camera would be the e-con Systems Tara or TaraXL, which would have to be ordered. These cameras would also be available for users in the industry as they are available on the market. The cameras shall provide point cloud data which will be processed

using the GPU-Voxels algorithm. The algorithm provides functionality regarding voxelisation, data compression and map generation.

Motion Planning

When an obstacle is identified within the workspace of the robot, the algorithm should compare its position with the planned trajectories of the robot. If an upcoming collision is identified according to the obstacles position, a new trajectory shall be calculated, so that the robot can move to its goal without colliding with the obstacle.

The Open Motion Planner Library (OMPL) should provide different planning algorithms, which could be used to solve the motion planning. New ways, avoiding the obstacle, shall be calculated by using the voxel map which the system provides. The system shall calculate the shortest way form the robots current position to its goal by using the free spaces which the map provides.

Robot Communication

The used Fanuc CR35iA Cobot only runs Karel code. Karel is a programming language based on Pascal. To use any C++ library for the motion planning, a communication between the robot using Karel and the C++ program needs to be implemented. For this case, Fanuc enables socket communication over TCP/IP, which can be used to transfer robot pose and joint speeds to the C++ code. The C++ Code processes the data from the robot and the object detection and delivers desired go-to positions which will be read by the Karel code and used to move the robot to the goal.

2.3.2 Safety and perfomance requirements

Jam avoidance

Between the two joints of Fanuc CR35iA Cobot, marked in Figure 2.1 no force measurement is available, which means objects or human limbs can be jammed between the joints. This is a massive safety risk for the user of the robot. To increase the safety of the user, any detected object in between the joints should lead to an immediate stop of the robot movement which would

lead to a jam. If possible, the robot shall turn and move away from the object to prohibit a jam and again calculate a different path to its initial goal without damaging anyone or anything and without pausing the production.

Figure 2.1: Fanuc CR35iA Cobot at BFH AHB

Safety stop

When a certain minimum distance to the obstacle is reached, the robot should perform an immediate stop for safety reasons. This minimal distance can be defined by the user of the system, considering the speed and given circumstances in the workspace.

If the goal is not reachable, due to any object preventing a collision free movement (e.g. a bottle laying on the goal position), the robot shall stop until the obstacle is removed and the goal is free.

Environment

Regarding the current environment of the robot and future use, the system should work in a normal industrial environment, without being too sensitive to changes of the light conditions depending on the time of day. Therefor additional light sources may be included while testing the system.
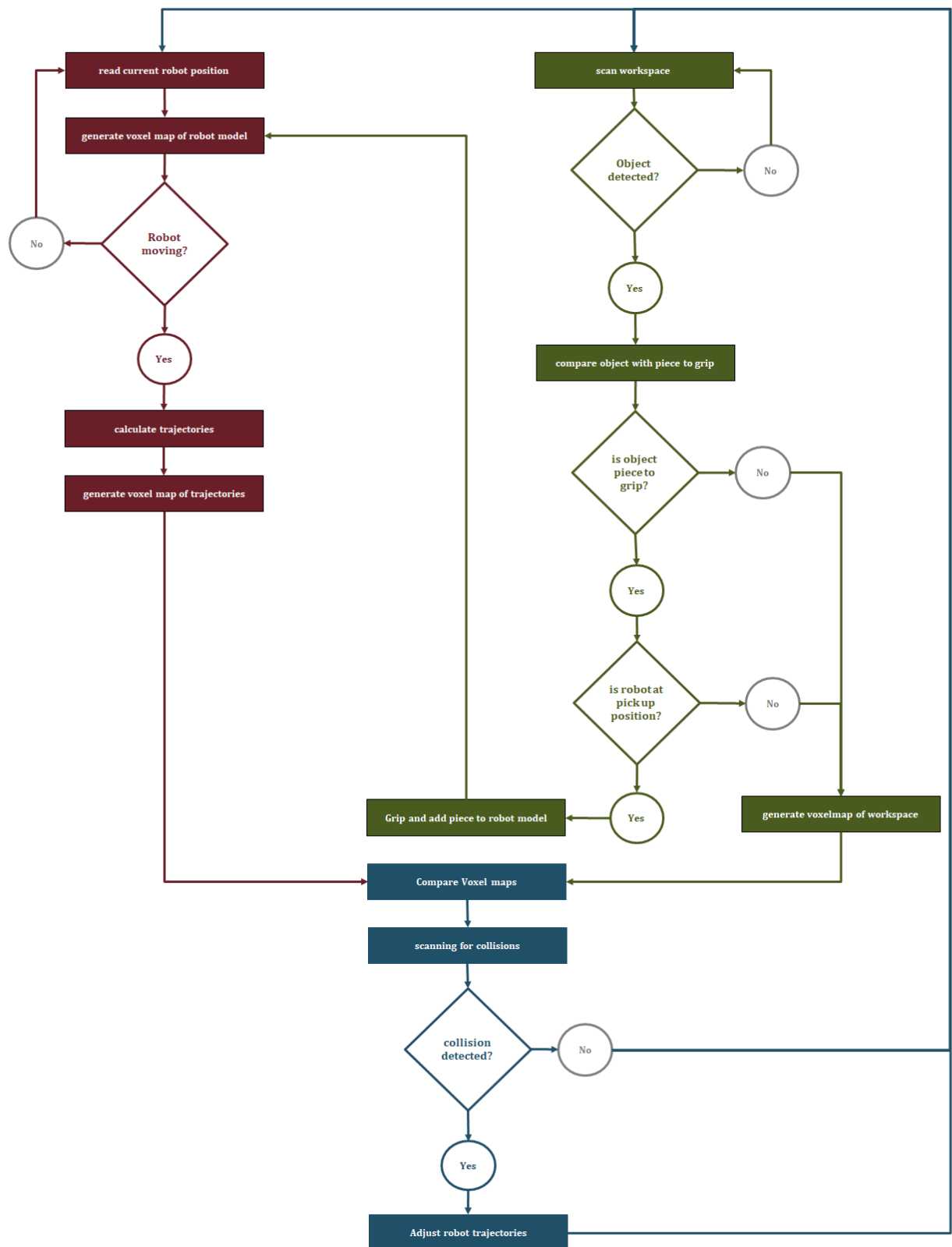
2.4 System Flow Chart

Figure 2.2: Block diagram of the System

3 Planning of the preliminary study

The first step of this preliminary study was to plan, how to attempt this project. A plan and a timetable had to be created by taking in account the given work specification [9]. These work specifications defined the following goals for the preliminary study.

- Write specification, test and validation plan.
- Perform a literature review.
- Select, order and commission parts for test setup.
- get familiarized with robot system and Interface.
- Integrate a model of the robot in the collision-avoidance software.
- Generate a project plan for the thesis.

The procedure was planned according these goals by giving them a timely order and a duration per task. This plan was implemented in a Gantt chart which is shown in figure 3.1. During the execution of this preliminary study, the plan has been adapted to the current state of the work.

As planned, the literature review started right away but takes place over the whole preliminary study, as there will always be some topics that need further research. The research has been done online by using forums and library and program documentations. Next to the research, the system specification has been one of the first tasks to approach, due to the reason that the specifications build the fundament for the implementation. When system specifications are defined, an according test to validate the system shall be defined and documented. These tests allow to define measurable goals for each implemented specification as the test can be either successful or not. With this information the validation can be carried out. While finishing the test and validation plan, the author must get acquainted with the robot system and its interface.

While getting familiar with the robot system, a decision about the needed hardware components need to be taken. It's important to choose the parts in an early stage of time as they may need

some time to be delivered. All components need to be available in week 12 of the project, so that they can be used when the Bachelor Thesis starts.

A big amount of time is planned for the implementation of the robot model. The goal of this implementation is, that the planning of the thesis can be done basing on the progress of this implementation. So the further this implementation is along, the more detailed and realistic the planning of the Thesis can be foreseen.

The Last Week of the preliminary study acts as buffer zone, with reserved time for proofreading and finishing the documentation.
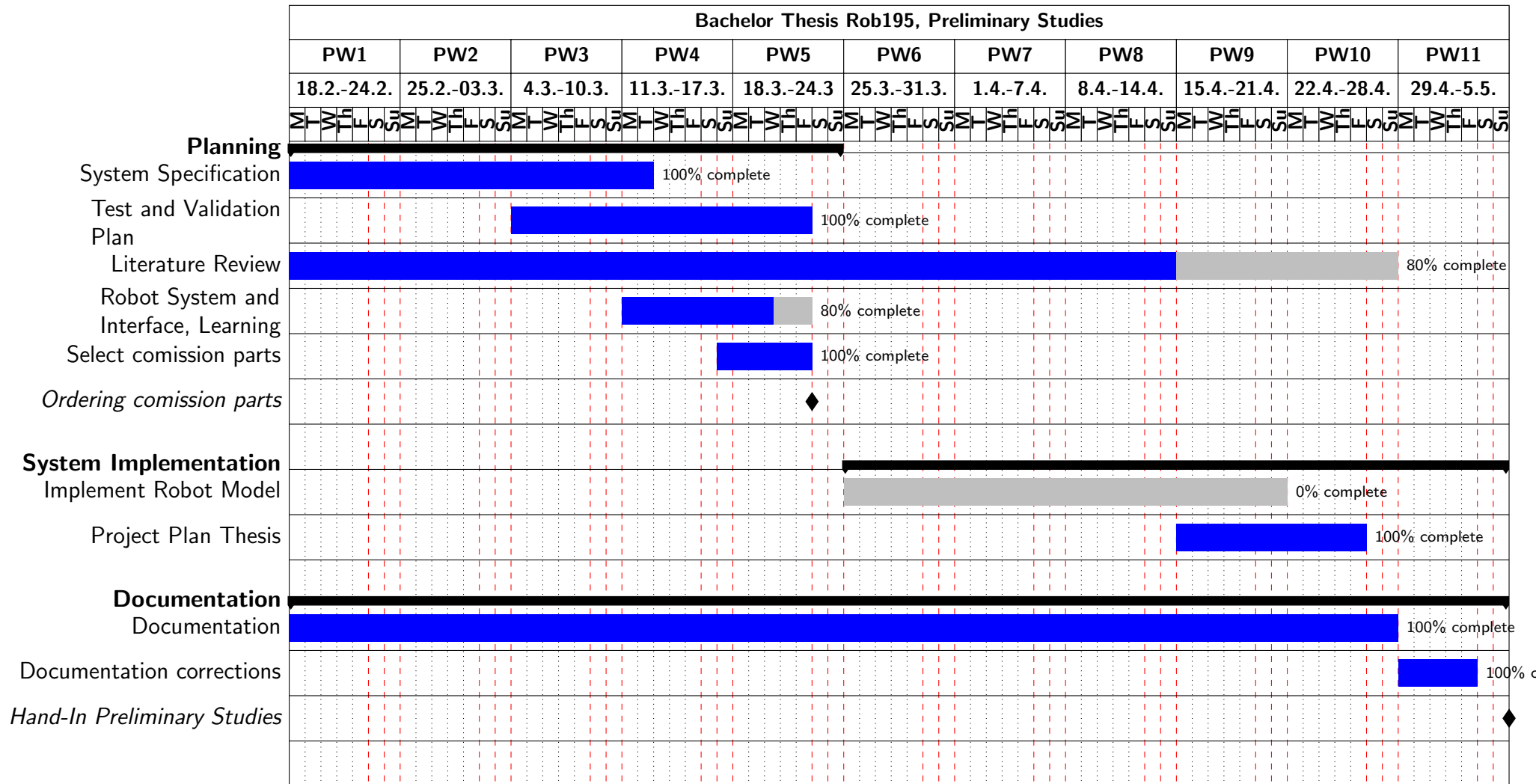
Figure 3.1: Gantt Chart, Preliminary Studies Rob195

4 Execution and Research

As a first step in the execution phase of the preliminary study, the programming environment needs to be set up. As the needed GPU-Voxels algorithm is only tested by the provider FZI Forschungszentrum Informatik on 64-bit Ubuntu 14.04 Linux Trusty and 16.04 Xenial systems, it was decided by the author to favor Ubuntu over Windows, but to use a newer Ubuntu distribution to work on a current system. Therefore, the newest version was chosen, which was at this point of time the version 18.04 Bionic Beaver LTS. As programming IDE, Qt Creator was chosen, since this IDE was already used by the author during previous courses in robotics and informatics and is thus well known. As a prerequisite for the GPU-Voxels algorithm, a CUDA compute capability greater than 2.0 is required. The compute capability of a GPU determines its general specifications and available features. This leads to the decision to use the authors private tower computer instead of the laptop used for the studies. The laptop couldn't be used because its onboard graphics card isn't supporting nVidia CUDA at all. Before week 12, start of the Bachelor Thesis, a Laptop with according performance shall be provided by the AHB to be used for this project.

4.1 GPU-Voxels Algorithm[6]

The GPU-Voxels algorithm was developed at the FZI Forschungszentrum Informatik in Karlsruhe, Germany. GPU-Voxels is a CUDA based library which allows a high resolution volumetric collision detection between animated 3D models and live point clouds from 3D sensors of all kind. Mainly developed for the use in robotics planning and monitoring tasks. The algorithm is able to voxelize 3D models and point clouds. By comparing voxel positions of the robot model and the point clouds, collisions can be accurately detected and visualized using third party visualizers [4].

Voxel

A voxel is the three dimensional counterpart of the two dimensional pixel. As the pixel represents a certain area, depending on the resolution, in an XY-plane, a voxel represents a certain volume in an XYZ-space. Alongside its position a certain value can be assigned to a Voxel. This can be either binary, representing a simple free or occupied state, or multivalued, representing different data (e.g. color, density, heat or pressure).

Point cloud

A point cloud is a set of data points in a three-dimensional space, mostly representing the surface of an object or space. Based on how many points are used to create a point cloud, surfaces can be reconstructed very accurately. Typically, LIDARs or Stereo Cameras are used to gather Point Clouds.
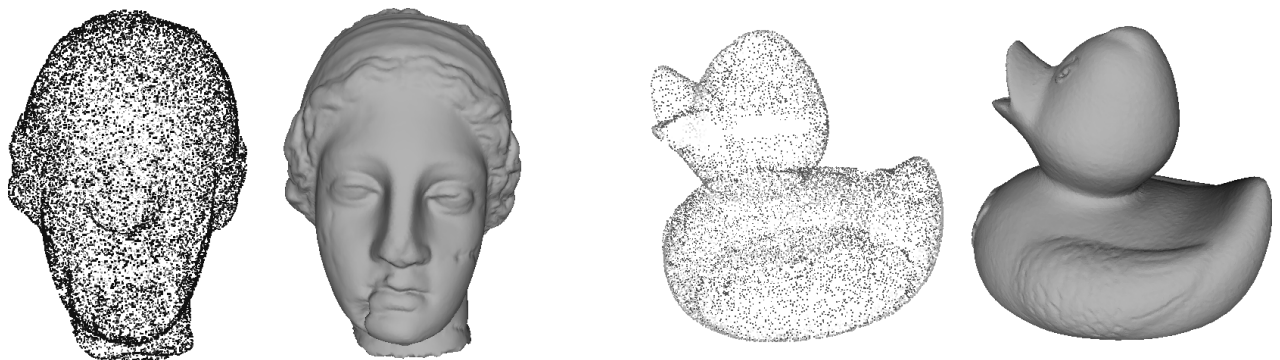
Figure 4.1: Point Clouds and their reconstructed surfaces. [2]

4.1.1 GPU-Voxels Prerequisites

The GPU-Voxel Algorithm depends on several other Libraries which need to be installed in order to be able to compile GPU-Voxels. The following packages are necessary and must be available:

- **nVidia CUDA version 5.5 or higher**
  CUDA enables many computing cores in a graphics processor to perform general-purpose mathematical calculations, which leads to dramatic speedups in computing performance.

- **Point Cloud Library (PCL)**

  The Point Cloud Library is an open source library for 2D and 3D image and point cloud processing.

- **OpenNI**

  OpenNI is an open source software development kit (SDK) used for applications for depth sensors such as Kinect and Asus Xtion.

- **Boost**

  Boost is a set of libraries for C++ that provides support for tasks and structures in linear algebra, pseudorandom number generation, multithreading and image processing. It contains over eighty individual libraries which are aimed at a wide range of C++ users.

- **TinyXML**

  TinyXML is an XML parser which converts data from an XML format in an in-memory form and creates a way for programs to use that data.

In order to use the Visualizer further packages are necessary, but in a first installation they are not required and therefore not described in detail. Installing the necessary packages, has proven difficult because some of them weren't compatible with each other. Sadly, no screenshots of the error message have been taken when occurred and are therefore now not available to be shown in this documentation. This is a finding, which the author now keeps in mind for the documentation of the Bachelor Thesis. While installing the nVidia CUDA Toolkit 10.0 the first time. Somehow the graphics driver was damaged, and a simple re-installation of the graphics driver didn't fix it. So, this led to a complete system re-install, with wiping the current Ubuntu partition and starting from Scratch. In a second attempt all packages could be installed, and the only task left was compiling the GPU-Voxels algorithm. While compiling, cmake had problems finding packages, which was a time consuming but not unsolvable problem. During the step of fixing the package location in the cmake files of the GPU-Voxel Algorithm, several missing packages have been found. Most of them related to Robot operating System (ROS). ROS is an open-source operating System for robotics, which provides common functionality and services used in robotics. While GPU-Voxels state that their algorithm is independent from ROS, but it is supporting it, it was never possible to compile the code without ROS. Some of these packages were found as standalone packages, but couldn't be installed because of unmet dependencies, which lists several other packages and states that

they are not going to be installed. This error could be avoided using the packages included in ROS. After installing the ROS Package, the graphics driver was damaged again. This lead again, to a clean start with a fresh Ubuntu installation, because a reinstallation of the graphics driver wouldn't fix the Problem. In the third attempt the installation order of the packages was changed, and ROS was installed right after CUDA. Afterwards the other necessary packages were installed and compiling the GPU- Voxels Algorithm could start again. As before, some packages were still missing, but could not be installed because of unmet dependencies. This whole process of trying to run the GPU-Voxels algorithm, was very time consuming and frustrating. A major mistake was, to lose focus on the whole project and how much time had already passed. In this process a lot of time was wasted without getting any results or achieving something. This phase made clear that the planning phase had failed, and the approach to the project was too straight forward directed to the main goal. Together with the missing documentation of the detailed errors and solutions, this Problem had a major influence on not being able to reach the set goals of this preliminary study.

4.2 Looking for alternatives

After the struggles with the installation of the packages reached a critical point in terms of time that passed, a decision about the next steps had to be taken. Possible alternatives which allow a progress in this project needed to be found.

As a first step, the task which should be completed by the GPU-Voxel algorithm was divided in different functionalities. Functionality one is, that the workspace needs to be scanned and processed. Functionality two is, that the data from the workspace needs to be compared with the robot model and its trajectories, to predict collisions. Using the data from the collision prediction, a motion planning has to take place in order avoid collisions. To achieve these functionalities the search for packages began by looking into the packages the GPU-Voxel algorithm uses to solve these tasks. A few of them sounded promising, but the search was widened to other standalone libraries for each task to ensure a profound research with the best solution possible. The most promising ones are shortly explained in the following sections.

4.2.1 Point Cloud Library [7]

The Point Cloud Library (PCL) supports the Microsoft Kinect cameras as well as the Asus Xtion Pro, which are available for this project. With this library it should be possible to create the necessary point clouds and process them into an octree data structure. The octree data structure was already mentioned in the documentation of the GPU-Voxels algorithm. PCL also provides efficient search routines.

Octree data structure

The octree is a tree data structure in which each internal node has exactly eight children. The root node describes a cubic bounding box which encapsulates all points from the point cloud. By dividing this cube into eight smaller cubes each representing a certain part of the root cube. The node is always the center point of the cube it is representing. Processing a point cloud into an octree data structure already provides a form of voxelizing, the resolution can be chosen by defining how many layers the octree data structure should have.
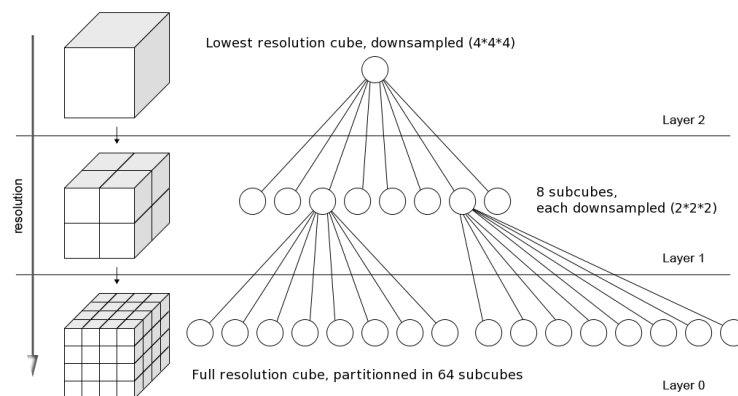

Figure 4.2: Graphical Illustration of an octree data structure. [1]

4.2.2 OctoMap [5]

The Octomap library implements a 3D occupancy grid mapping approach, providing data structures and mapping algorithms in C++ suited for robotics. The implementation of the map is based on the previously explained octree data structure, which further allows map-based motion planning to take place.

4.2.3 Open Motion Planning Library [8]

The Open Motion Planning Library (OMPL) consists of many state-of-the-art sampling based motion planning algorithms, but does not contain anything related to collision checking or visualization. So, to use these motion planning algorithms, the collision detection needs to happen before the motion planning and the data needs to be provided accordingly.

4.3 Fanuc Roboguide and TCP/IP Socket messaging

4.3.1 Roboguide

Fanuc has its own simulation software for Fanuc robots called Roboguide. Roboguide provides the user with a complete and realistic simulation of the robot including teach pendant controls of robots. It can also be used to write the Karel programs, which will be later executed from the robot. With Roboguide, every aspect of the robot handling should be able to be simulated, including TCP/IP connections. Simulations are done in workcells, where a robot model can be included and an environment representing the actual robots environment can be created. So as a first step to learn how to use Roboguide, a new workcell was created, which was held very simple only including the robot model. Since it was said that the current physical environment of the robot was going to change in the near future, the work to model the current state was postponed till after the changes has been realized. After including the robot model into the workcell, a simple Karel program was written to move the Robot in a rectangular shape. This code has just been written, to learn some basic Karel functionality and syntax.

4.3.2 TCP/IP

As a first step, the existing work of a former student, Nino Sutter, has been analyzed. The work was provided via a gitlab repository, where access was granted as his work has showed similarities. The code provided should act as an example on how to enable a TCP/IP connection with the robot. The example has a Karel code which connects the robot with a system running a python program. Using the python program, tasks for the robot could be loaded remotely from the memory of the robot. A similar procedure is planned for this thesis. In order to provide the robot with coordinates for his destination point, a C++ program should calculate these and send them to the robot for movement commands. Technically it should be possible to adapt the Karel code from Nino Sutter to achieve the TCP/IP connection for this project on the Fanuc robot. A first attempt using the example code to simulate a TCP/IP connection within Roboguide has not been

successful. The reason for that may be found in missing settings inside the project work cell. But this was not further investigated due to timely limits of the preliminary study and the problems which occurred during GPU-Voxel algorithm setup. Alternatively, the AHB has currently socket messaging, which is working on the robot for another project. This method of communication has not yet been reviewed, since the information came up in week 11 which is too close to the end of the preliminary studies due date. During the Bachelor Thesis this will be further analyzed to make sure to use the communication method that fits best [10].

5 Planning of the Bachelor Thesis

To have small and achievable goals to work on during the Bachelor Thesis, the project is divided in several milestones. In this chapter these milestones are listed and explained in detail on what is the goal, what are the methods used to achieve it, how is it tested and how to proof that they worked. The milestones will start from a few that are necessary to have the bare minimum done and will go up until the optimal solution. After the milestones were described, predictions are made on how far the project will be solved under different circumstances (e.g. worst-case scenario, no problems etc.).

5.1 Milestone 1: Running programming environment

Description

Until this point of time, the work for this project was completely done on private systems of the student. As mentioned in chapter 4 a laptop was provided by the AHB. Since it is a school laptop, a Windows system is installed. In order to not uninstall the Windows off the Laptop, an external SSD was provided which should be configured to act as a bootable Ubuntu 18.04 Bionic Beaver System. Therefore, the system will start again from a clean installation and all needed libraries will be installed on to this system.

Goal

The Goal is straightforward to have a running programming environment which has all needed libraries and programs installed.

Methods used to achieve the goal

Following installation instructions of different libraries which are offered by the providers of the needed libraries.

Testing

Ubuntu is bootable from multiple Devices. The boot will be tested on at least two different devices.

Validation and proof

The device allows a flawless boot of the Ubuntu into the desired home screen without changing the predefined setups. Proof is a written statement of the user / author.

5.2 Milestone 2: Robot communication

Description

Without the communication with the robot, the task of the project can't be achieved. Therefore it is very important to set up a stable connection and communication from the system to the robot in order to provide each side of the system with the necessary data.

Goal

Flawless communication between collision avoidance system and robot.

Methods used to achieve the goal

Review the example provided over git and the socket messaging method already in use at AHB. Work with the Fanuc manual in order to set up the communication.

Testing

In order to test the communication, a simple C++ program should be able to feed the robot Cartesian or joint coordinates to move to. The robot shall move accordingly.

Validation and proof

The C++ as well as the Karel code shall be pushed to git in a separate folder, to provide a simple example for future communication tasks. A video shall be provided as proof for the functioning communication. The video shall show the execution of the C++ code and the robot's movements. The video and the source code will be committed in a single commit in order to have a clear identification.

5.3 Milestone 3: Scanning of the workspace

Description

The cameras shall be implemented into the system and a point cloud of the Workspace shall be created. The data shall be processed into a two-dimensional occupancy grid for a first simple motion planner.

Goal

Visualizing the workspace in form of a point cloud and an occupancy grid.

Methods used to achieve the goal

Using the Point Cloud Library, the cameras shall be implemented. Since the PCL supports either Microsoft Kinect or Asus Xtion Pro, both cameras may be used, the one with the better resulsts shall be used for the rest of the project.

Testing

In order to test the task, the workspace shall be changed by placing different objects on various points inside the workspace. Point clouds and occupancy grids will be reviewed manually and checked for their accuracy.

Validation and proof

The successful test will be proceeded and documented with both kinds of cameras. The resulting grids are analyzed manually and a ranking for completeness and accuracy will be done. Proof shall be provided over screenshots and written comments of the observation and usability of the author.

5.4 Milestone 4: Simple 2D Motion Planning, End-Effector only

Description

In a first collision avoidance step, the robot shall move around objects in a static two dimensional environment. This means during robot movements the workspace shall not change. A simple algorithm for path planning is the PRM algorithm, which should be available over the Open Motion Planning Library. PRM is already known since it was used during the robotics 2 course. In this course an implementation was realized over Matlab.

Goal

Move from start position to given goal position without any collision.

Methods used to achieve the goal

The point clouds gathered from the cameras using PCL, shall be processed into an 2D occupancy grid using OctoMap. The probabilistic road map planner (PRM) uses occupancy grids and places nodes inside the map on randomly chosen free cells. The nearest nodes to the starting and goal location are searched. Then a path along the nodes is calculated and thus provides a path from start to goal position. The PRM planner can be used from the OMPL library. This step plans the motions only for the end-effector, which means the robot arm needs to remain elevated over the obstacles in order to remain collision free. Therefor movements can only be made in the XY-plane in Cartesian space.

Testing

By changing the workspace with different objects on various places, the map generation and the path planning shall be tested. The starting position shall be read from the Karel code as the current robot position, the goal position shall be provided over the C++ Code as an user input.

Validation and proof

Three successful test runs with different objects in different positions and variable start and end points shall be proceeded. A video of the robot's movement around the object shall be made for every run to proof that the path planning works in different test set-ups.

5.5 Milestone 5: Simple 3D Motion Planning, End-Effector only

Description

As a second collision avoidance step, the robot shall move around in a static three dimensional environment. The objects inside the workspace shall remain static during robot movements. For the motion planning, again the PRM algorithm shall be used, but this time in a 3D space.

Goal

Move from start position to given goal position without any collision in the three dimensional space.

Methods used to achieve the goal

The point clouds gathered from the cameras using PCL, shall be processed into an 3D occupancy grid using OctoMap. Using the PRM planner from OMPL a path from start to goal shall be calculated and afterwards moved along with the robot. This step plans the motions only for the end-effector, which means the robot arm needs to remain elevated over the obstacles in order to remain collision free. Therefor movements can only be made in the Cartesian space.

Testing

By changing the workspace with different objects on various places, the map generation and the path planning shall be tested. The starting position shall be read from the Karel code as the current robot position, the goal position shall be provided over the C++ Code as an user input.

Validation and proof

Three successful test runs with different objects in different positions and variable start and end points shall be proceeded. A video of the robot's movement around the object shall be made for every run to proof that the path planning works in different test set-ups.

5.6 Milestone 6: Collision detection for whole robot model

Description

The next step to improve the collision avoidance system shall provide a collision detection not only for the end-effector but for the whole robot model. This shall be achieved by creating a occupancy grid for the robot model and its trajectories. By comparing the two different grids (workspace and robot) collisions shall be predicted and avoided.

Goal

Find or create a motion planner who is able to calculate a path based on swept volumes. Move from start position to given goal position without any collision, movements can be done in joint space or Cartesian space.

Methods used to achieve the goal

A 3D model of the robot provides the base for the robot occupancy grid. Using the joint speeds, a prediction of the joint positions can be calculated and swept volumes can be created inside the occupancy grid. Both occupancy grids shall be compared by simply checking if a cell is occupied in both grids. The cells which are occupied in both grids show a collision that needs to be avoided. To have a proper motion planning algorithm based on these data, further literature review is necessary.

Testing

The workspace shall be changed for different motions by placing various objects in different positions. In order to test the collision detection for the robot arm, objects shall be placed in front of the arm to force a movement to avoid the collision not only of the end-effector.

Validation and proof

Three successful test runs with different objects in different positions and variable start and end points shall be proceeded. A video of the robot's movement around the object shall be made for every run to proof that the path planning works in different test set-ups.

5.7 Milestone 7: Collision detection in a dynamic workspace

Description

The previously used motion planner shall be adapted to work in a dynamic workspace (e.g. moving objects, humans). The movements of the robot shall be adapted in real time to avoid collisions. The results of the collision avoidance system shall be visualized in a GUI.

Goal

Robot moves from start to goal location in a dynamic workspace, without any collisions. Implement a GUI to show the results of the collision avoidance system.

Methods used to achieve the goal

The workspace shall be monitored in real-time, updating occupancy grids on the fly. The motion planner should work with the permanently updating occupancy grids in order to react to dynamic obstacles. Further literature review is necessary to achieve this goal.

Testing

In a first phase, object shall be moved inside the workspace, using objects which can be moved from outside of the workspace to reduce risks for humans. In a second phase, when the system has a high level of success an values can be considered as trustworthy, humans shall enter and leave the workspace in order to simulate a walk by.

Validation and proof

Six successful test runs with different objects in various positions and variable start and end points shall be proceeded by using objects which are moved from outside of the workspace. A video of the robot's movement around the object shall be made for every run to proof that the path planning works in different test set-ups. After the successful first testing phase another three test runs with humans shall be proceeded and recorded on video.

5.8 Prediction

Milestones one to four are defined as bare minimum, as these are the necessary milestones to have a first path planning based on the sensor data, which implements a first step of the collision avoidance system. Based on the actual knowledge it should be possible to reach Milestone six within the given eight weeks of the Bachelor Thesis. If progress during the work is exceptionally good, even milestone seven can be reached.

To prevent the lost of track, an escalation to the experts is needed, whenever the forseen timeline of the worst case is not fulfilled. This would mean, that the minimal goal is threatened and a solution to this upcoming problem need to be found. In addition it helps the author to prevent losing focus as it happened in the preliminary study.
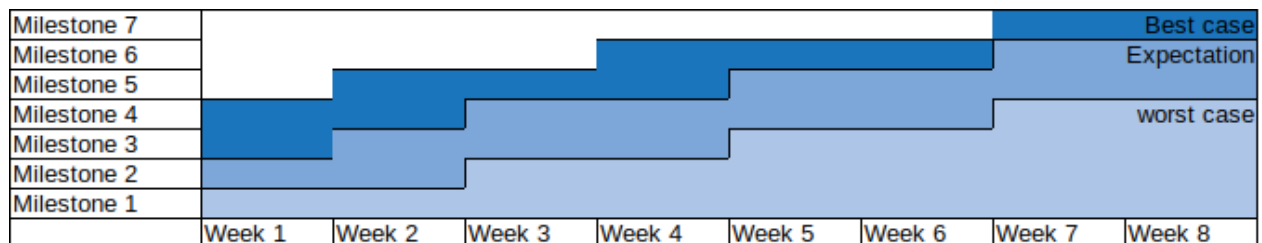

Figure 5.1: Milestones in Timeline

6 Conclusion / Results

Overall the process of the preliminary study did not work out as expected. On one hand the technical requirements for the programming environment caused way more problems than expected and on the other hand, the time planning and focus of the author did get out of hand. The combination of these two issues caused major time and motivation problems. Therefore the result of the preliminary study is not satisfying for the author in accordance to the initial expectation. Nevertheless, working on this project is very interesting and challenging, which means that the will to continue this project is unbroken.

Content wise there should have been a bigger accomplishment according the time given and the massive amount of time invested. After many hours of literature review, a sobering low amount of portable possibilities has been found. This leads to the conclusion that the way of research has to be adapted in order to find suitable solution for the existing issues. Due to the adaption of the initial plan and the exchange with the experts, a realistic approach has been defined. Under consideration of all these factors, the best possible result is now provided.

The most important learning for the work on the milestones is to keep an strong focus on the actual tasks combined with a good and mindful time management. Also the exchange with other students and the experts shall be kept alive to benefit of the given knowledge. Help shall be searched at a earlier point of time, which does not mean, not to try to fix problems by the authors own possibilities and serious tries.

The definition of the milestones provides a promising planning of the Bachelor Thesis and seems to allow a good result for the main goal of the project. Needed Hardware and Software is mostly identified and methods to fulfill the requirements are described.

An intense research of the topic and possible alternatives for occurred problems has been executed and leads to a knowledge which can be used and extended during the Bachelor Thesis.

For the start of the Bachelor Thesis, only the learning will be taken into account and the disappointment is left behind. With great motivation and a strong will to perform well, the author is now looking forward to the realization of the milestones of the Bachelor Thesis.

Declaration of primary authorship

I hereby confirm that I have written this thesis independently and without using other sources and resources than those specified in the bibliography. All text passages which were not written by me are marked as quotations and provided with the exact indication of its origin.

Place, Date: Biel, 06.05.2019

Last Name, First Name: Aeschlimann, Dario

Signature: ....

Bibliography

[1] "Octree." [Online]. Available: https://cvlab.epfl.ch/research/page-90589-en-html/research-completed-medical-8tree-index-php/

[2] "Pdes and variational method on 3-d surfaces and point clouds." [Online]. Available: https://elmoatazbill.users.greyc.fr/point_cloud/index.html

[3] "Process engineering." [Online]. Available: http://processengineering.co.uk/article/2020966/fanuc-extends-cobot

[4] J. B. S. K. A. R. A. Hermann, F. Drews and R. Dillmann, "Unified gpu voxel collision detection for mobile manipulation planning," 2014. [Online]. Available: http://www.gpu-voxels.org

[5] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[6] A. E. Kaufmann, *Voxels as a Computational Representation of Geometry*. State University of New York at Stony Brook, 1994. [Online]. Available: https://pdfs.semanticscholar.org/84d4/0dbadd6a0b25ffe7f46844e9a2636285b949.pdf

[7] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[8] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[9] B. TI and G. Gruener, "Rob195 work description."

[10] B. TI and N. Sutter, "Motionserver.kl."

List of Figures