Tampere University

Miika Kuusisto

RGB-D SLAM BASED RECONSTRUCTION OF 3D OBJECTS

Faculty of Information Technology
and Communication Sciences
Bachelor's Thesis
May 2019

ABSTRACT

Miika Kuusisto: RGB-D SLAM based Reconstruction of 3D objects
Bachelor's Thesis
Tampere University
Bachelor's Degree Program in Information Technology
May 2019
Examiner: Joni Kämäräinen

An important part of robot design is its vision, so it is important to study and compare the methods for vision. RGB-D Simultaneous Localization And Mapping (SLAM) based methods are one way to take the color and depth data and compute make a 3D model of a scene. Most of the RGB-D SLAM based methods are designed to model a large area such as a room, but this thesis focuses on comparing RGB-D SLAM based methods on a smaller scale. The purpose is to study and compare the methods by reconstructing single objects. The chosen objects vary in size and complexity to test the methods more broadly.

There were two methods that we were able to get working in reasonable time and thus were included in the comparison. The first was Kinect Fusion which is a part of a Point Cloud Library (PCL) and the second was Static Fusion which is its own method. RGB-D data was gathered by using an Orbbec Astra series sensor. The criteria that was used to evaluate the methods were ease of install and the quality of the 3D model.

The main result is that both methods are good but depending on what the priorities are one was superior to the other. Static Fusion was fast and easy to install and to get working, does not depend on many libraries and is simple in design. Kinect Fusion as its part of a whole brings many other neat and useful features in addition to its RGB-D Slam method, which can be a pro or a con.

Keywords: RGB-D SLAM, PCL, Kinect Fusion, Static Fusion

TIIVISTELMÄ

Miika Kuusisto: Kohteiden 3D-rekonstruktio RGB-D SLAM menetelmien avulla
Kandidaatintyö
Tampereen yliopisto
Tietotekniikan kandidaatin tutkinto-ohjelma
Toukokuu 2019
Tarkastaja: Joni Kämäräinen

Tärkeä osa robottien suunnittelua on niiden näköjärjestelmä, joten on tärkeää tutkia ja vertailla erilaisia tapoja, jotka toteuttavat näköjärjestelmän 3D-havainnointia. RGB-D SLAM menetelmät ovat tapoja ottaa väri- sekä syvyysnäkö sensorista ja yhdistää nämä 3D-malliksi. Osa RGB-D SLAM menetelmistä on selvästi suunniteltu suuren alueen kuvaukseen ja mallintamiseen, esimerkiksi kokonaisen huoneen kuvaamiseen ja luomaan siitä 3D-mallin.

Työssä tarkastellaan RGB-D SLAM menetelmien käyttämistä pienemmässä skaalassa. Erityisesti tarkoituksena on tutkia ja vertailla menetelmiä käyttäen yksittäisiä objekteja malleina ja vertailukohteina. Työssä valitut objektit ovat kooltaan ja monimutkaisuuksiltaan erilaisia, jotta saataisiin mahdollisimman laaja ote menetelmän tuottamasta tuloksesta

Työssä saatiin toimimaan kaksi menetelmää, joita verrataan toisiinsa, Static Fusion ja Kinect Fusion. Kinect Fusion on osa suurempaa kirjastoa Point Cloud Library (PCL) ja Static Fusion on oma erillinen ohjelmansa. RGB-D datan keräämistä varten työssä käytettiin Orbbec Astra sensoria. Menetelmiä arvostellaan tarkastelemalla niiden tuottamien 3D-mallien laatua, sekä menetelmien käyttämisen ja asentamisen vaikeuksien avulla.

Tutkimuksen lopputuloksena oli, että molemmat menetelmät ovat hyviä, riippuen mitä ominaisuuksia käyttäjä haluaa ja kuinka nopeasti menetelmä pitää olla toimintakunnossa. Static Fusion on huomattavasti helpompi asentaa ja käyttää, eikä ole riippuvainen kuin muutamasta ohjelmistokirjastosta. Kinect Fusion tuo mukanaan monta muuta ominaisuutta pelkän RGB-D SLAM menetelmänsä lisäksi.

Avainsanat: RGB-D SLAM, PCL, Kinect Fusion, Static Fusion

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

CPU	Central Processing Unit
GPU	Graphics Processing Unit
RGB-D	Red, Green, Blue and Depth. Color vision in addition to depth
SLAM	Simultaneous Localization And Mapping
TOF	Time of Flight.
3D	3-dimensional.

1. INTRODUCTION

Machine vision has become a core part of robotics and artificial intelligence. Usually when talking about machine vision we talk about 2-dimensional pictures or video. 2-dimensional machine vision is used for example to identify objects within a picture or on video [1]. With technological advancements 3-dimensional machine vision is becoming the new and improved form of machine vision for robots and artificial intelligence. 3-dimensional imaging provides a more accurate model of the wanted objects.

Visual simultaneous localization and mapping (SLAM) technologies work by tracking set points through successive camera frames to triangulate their 3D position, while simultaneously using this information to approximate camera pose [2]. This is possible with a single 3D vision camera as long as there are a sufficient number of points being tracked through each frame. Visual SLAM is still an emerging technology that has a lot of potential especially in robotics. SLAM is a way for a robot to localize itself in an unknown environment, while incrementally constructing a map of its surroundings [3]. Due to recent advances in CPU and GPU technologies, the real time implementation of the required algorithms is no longer an insurmountable problem. Indeed, variety of solutions using different visual sensors including monocular, stereo, omni-directional, time of flight (TOF) and combined color and depth (RGB-D) cameras have been proposed [3]. In chapter 2 we will look further into visual SLAM technologies including the methods used in this BSc thesis project.

The purpose of this thesis is checking multiple methods of visual SLAM and test how well they reconstruct 3-dimensional objects of varying size and complexity. As most of the visual SLAM methods are made for at least room scale spaces, the goal is to see how well the methods work on a smaller scale in order to use the methods in robotics e.g. in object manipulation. In chapters 3 and 4 we go more into detail about the project and its results.

As the used methods work with different components that affect the outcome, overall usability and ease of installation of the methods, it is important to note that if different

components are used then the results may vary. In the research an Orbbec Astra sensor was used to gather the combined color and depth data. The methods were installed on a Windows 10 operating system, with a NVIDIA GTX 980 Ti graphics card and an Intel I7-6700K was used as the CPU.

Chapter 2 explains briefly the relevant information on RGB-D SLAM and goes in-depth on the theory side of the methods used in this project. Chapter 3 presents the methods used in the research and the procedures used in the installation of the methods as well as relevant information on the hardware used in addition to the software. Chapter 4 presents the results of the methods that were successfully installed. Lastly, Chapter 5 ends the document with the conclusions of the research.

2. RGB-D SIMULTANEOUS LOCALIZATION AND MAPPING

This chapter briefly explains the theoretical side of machine vision and RGB-D SLAM as well as the two methods used in the research.

2.1 RGB-D data

Digital color images from digital cameras are usually described by three color values: R (red), G (green), and B (blue) [4]. It is possible to represent all colors by using these three primary colors. In order to acquire a digital image, the hardware requires a camera, lens and a lighting source. Software is then needed to display and extract information from the images.

2.2 Kinect Fusion

Kinect Fusion was originally created by Microsoft for Microsoft's Kinect sensor [5]. The version of Kinect Fusion used in this work is the open source version of Kinect Fusion [6] from Point Cloud Library's project. PCL uses the original Kinect Fusion code and calls it internally as Kinfu.

Kinect Fusion consists of four main components: Surface measurement, Surface reconstruction update, Surface prediction and Sensor pose estimation [5]. Surface measurement is a pre-processing stage, where a dense vertex map and normal map pyramid are generated from the raw depth measurements obtained from the sensor [5]. Surface reconstruction update is composed of the global scene fusion process, where the pose determined by tracking the depth data from a new sensor frame, the surface measurement is integrated into the scene model maintained with a volumetric, truncated signed distance function (TSDF) representation [5]. Surface prediction closes the loop between mapping and localisation by tracking the live depth frame against the globally fused model [5]. This is done by raycasting the signed distance function into the estimated frame providing a dense surface prediction against which the live depth map is aligned [5]. Lastly the sensor pose estimation is tracking the sensor live using a multi-scale ICP alignment between the predicted surface and current sensor measurement, which the GPU based implementation uses all the available data at frame-rate.

2.3 Static Fusion

Static Fusion [7] is a SLAM system for RGB-D cameras that focuses on background segmentation and the filtering of dynamic objects in the foreground. Static Fusion works by jointly estimating the motion of an RGB-D camera and segmenting the scene into static and dynamic parts [7]. By decoupling the static and dynamic parts, it is possible to build a background model of the environment which fuses only the static elements. Static Fusion applies a dense mapping system which fuses only the temporally consistent data and does so quite efficiently achieving around 30ms/frame runtime [7],

Static Fusion takes in an input of a stream of registered RGB-D images and takes an RGB-D pair as a color and depth image. Firstly, the incoming pair is segmented into geometric clusters and each cluster is assumed to behave as a rigid body, which allows the solving of static/dynamic segmentation cluster-wise as opposed to pixel-wise [7]. Second, an artificial image pair is rendered by placing a virtual camera at the previous camera pose estimate, within the then constructed map of the static scene [7]. Thus, given the image and the predicted image the next step is to jointly obtain the camera motion and a motion-based segmentation of the scene between the two time instances [7]. Lastly, each cluster is assigned a score based on the level of dynamism [7]. With that done the clusters and scores are used to compute a per-pixel segmentation image for each point belonging to the background, which is used in with the color and depth images for weighted 3-dimensional fusion [7].

3. INSTALLATION OF AVAILABLE METHODS

The methodology of the work for any of the RGB-D SLAM methods was as follows. First a quick background check was done to find out how active the development was for the method and what are the current version's dependencies. Next step was to check for possible compatibility issues with the dependencies and if no obvious issues were found then to download and install the dependencies. The last step was to install the RGB-D SLAM method itself and test it.

In this chapter we will first list and briefly talk about some of the methods that failed to work. After we will talk about the methods that were gotten to work. The computer components which affected the progress of the research was mainly the operating system and graphics card. Windows 10 was the primary operating system and Linux was used with a virtual machine inside the primary operating system. The graphics card was a NVIDIA GTX 980 Ti which runs on the 5.2 CUDA architecture. It is unsure if the CPU's architecture affects these methods but the one used in this research Is Intel's I7-6700K.

3.1 Failed methods

Dynamic Fusion [8] was a method that was tested on Linux's Ubuntu 15.04. Dynamic Fusion has a script which was used to download all the dependencies and install the method. There were two problems that arose while trying to build Dynamic Fusion. First, there was an issue where the script tried to build a folder and didn't have the privileges to do so, which was solved with running the script with root privileges but then the script gave an error that the folder was already made. It was found that with root privileges the script tried to build the same folder twice and errors on that. The more troubling error occured later as the script tries to download data from an internet page that no longer exists, it was quickly decided that this would be difficult to fix so Dynamic Fusion was dropped.

OpenChisel [9] was tested on Linux's Ubuntu 14.04. While the open chisel part was relatively easy to install with its dependencies, the method also depended on a slightly separate wrapper called Chisel ROS. Chisel ROS had some errors with installing its dependencies. As OpenChisel was one of the first methods to be tried in the research it was put on hold due to difficulty in the errors and was eventually dropped.

3.2 Working methods

Static Fusion was simple and easy to install as it was the lightest RGB-D SLAM method with the just a couple of dependencies. Static Fusion was installed on Windows 10, with just having to tweak the CUDA architecture to make sure the right version was being supported when building.

Point Cloud Library and its Kinect Fusion method was troublesome to install and get working, due to PCL being a much more than just an RGB-D SLAM method. The errors that occurred during the installation of Kinect Fusion and its dependencies were numerous yet always seemed simple and easy to fix. The problems included wrong or mismatched versions of the dependencies, problems detecting necessary files and some incompatibilities with hardware being used. The CUDA architecture required heavy tweaking and testing, in addition to the Orbbec sensor and its own software. In order to get these two to work parts of the Kinect Fusion's code was changed by hand to use the newer software. After multiple troubleshooting sessions PCL along with Kinect Fusion was for the most part successfully installed on Windows 10.

4. EXPERIMENTS

This chapter presents the individual results of the methods and after that they are compared. In the figure below is shown the scene that is used in the experiments.

Figure 1. *The scene and the objects used in the experiments*

4.1 Static Fusion

First with Static Fusion the results of the very first few seconds of forming can be seen in figure below, which was taken about 3 seconds after turning the method on.

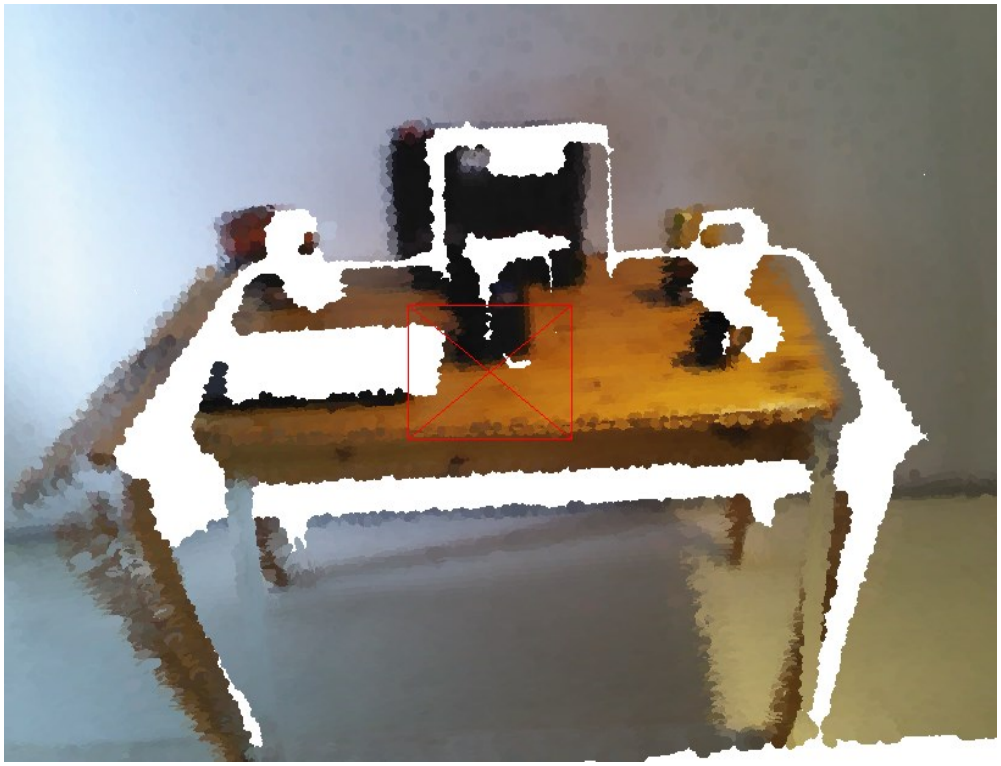

Figure 2. *Results of the Static Fusion after a few seconds.*

Figure 2 shows the raw output Static Fusion taken from a single location giving a few seconds for the results to form. Next to produce a better model the sensor was moved around the table while trying to keep the center of focus on the midmost object.

To produce a better model the sensor had to be moved around the objects, but as Static Fusion removes the dynamic elements of the scene great care had to be taken when moving the sensor. The sensor was moved around very slowly in about 180 degrees around the objects on the table, as this was done by hand it is possible there was some shaking and other small movements, which could have caused some error with the method.

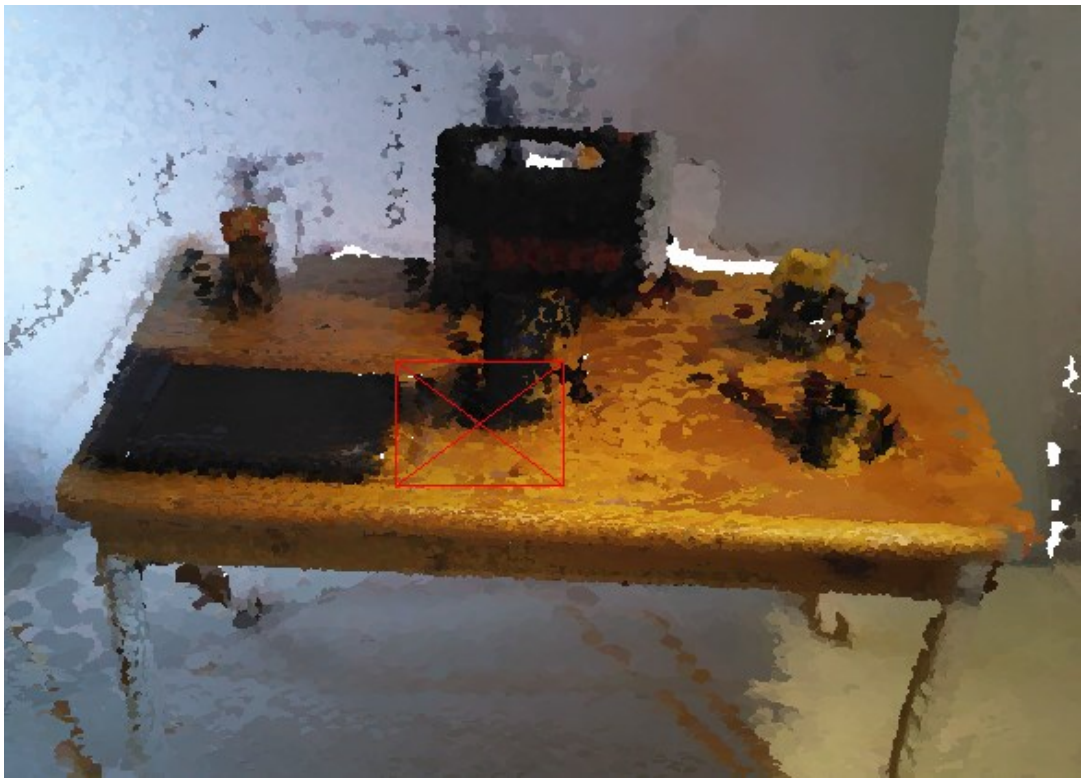

Figure 3. *Static Fusion results after moving the sensor around the objects*

Figure 3 shows a more complete model that does still have some errors around some of the objects, overall the results seem fine.

4.2 Kinect Fusion

For Kinect Fusion getting the colors along with the model proved tricky. The method creates a raw model but takes the colors as a separate point cloud. There was no clear way to merge these two results and all tries to do so ended with errors. Kinect Fusion creates the raw model near instantly and the results of that can be seen in Figure 4.

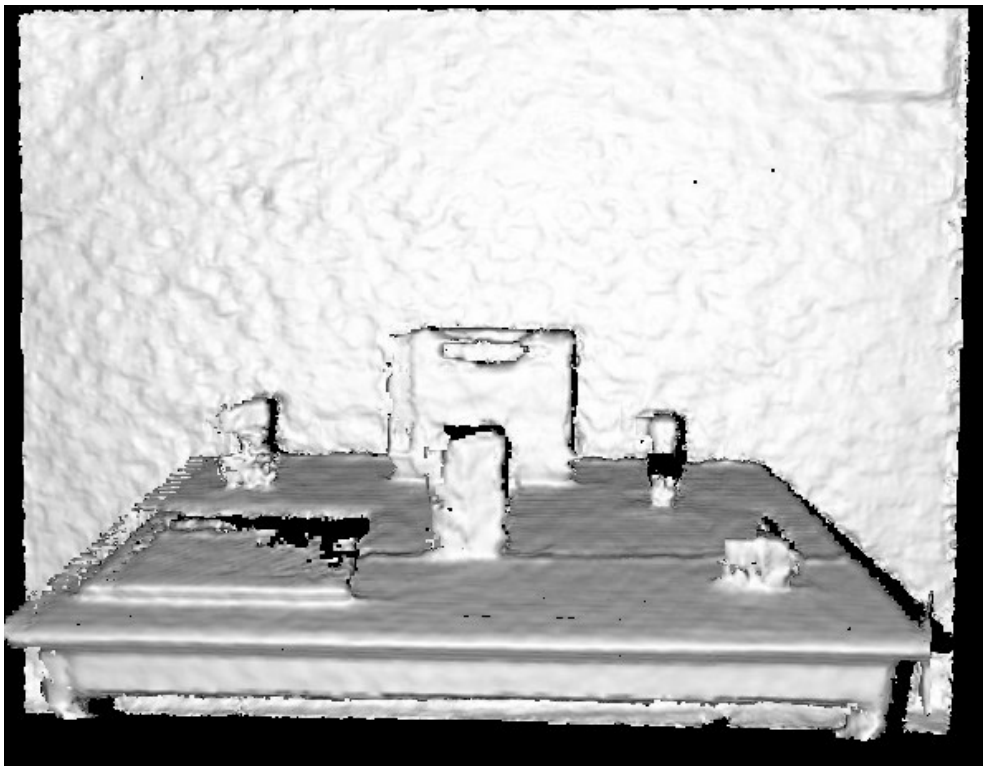

Figure 4. *Results of Kinect Fusion's colorless model taken from a single location.*

Figure 4 shows the instant results of Kinect Fusion. The model looks a bit choppy but overall captures most of the objects fine. Kinect Fusion seemed to be susceptible to some error when moving the sensor around, resulting in the objects disappearing from the model, so the sensor was moved quite little overall. While making the model there is an option to paint the model with the color image that the sensor sees, this can be seen below in Figure 5.

Figure 5. *Colored view of Kinect Fusion's model*

While Figure 5 does not represent the real results of a colored version it does give a little more context to the model itself.

4.3 Comparison

For the comparison pictures the cleanest results were used and for Kinect Fusion just the model without color was chosen. These results are shown in Figure 6.

Figure 6. *Cleanest results of Kinect Fusion (left) and Static Fusion (right)*

Figure 6 shows that the clean results at least do not differ beyond Static Fusion having color. Just by looking at the models alone they are both similar, the small mug has slight error as the handle is missing. The middle two rectangular objects are clear and easy to identify while the two smaller rectangular objects off the sides have slight error in the middle of them. The laptop computer is generally nicely shown with more error forming on the top on Kinect Fusion's side.

The Kinect Fusion has much more than just the RGB-D SLAM method and when the models themselves are compared there isn't much to differentiate between Kinect Fusion and Static Fusion. Kinect Fusion was also much faster in producing the model. But, if easily achieved colored models are needed then Static Fusion is the better option.

During work a point of failure was discovered and that was present in both methods. The failure occurred when the scene being modelled was too simple and symmetrical, in the experiment this was a single object in the middle of the table. At first the scene is modelled fine but as the sensor is being moved over 90 degrees from the starting point the produced model starts shaking and goes completely off as the view in the sensor flips sharply. The error probably occurs as the sensor has trouble finding its location from the scene it sees, as it is symmetrical and simple.

5. CONCLUSIONS

The goal of this research was to see which RGB-D SLAM method produced the best results in reconstructing 3D objects. While the goal was reached and the best method found, the amount of methods successfully tested was quite slim and thus the project gives a narrow look into the RGB-D SLAM methods. If more methods were successfully tested, the project's results would have been more significant.

Based on the results, Static Fusion gains the lead with its simple design and easy installation in comparison to Kinect Fusion. While the Kinect Fusion was good and has more features for more advanced techniques, it lacks the ease of installation and use.

REFERENCES

[1] Feature Extraction. Available from: https://deepai.org/machine-learning-glossary-and-terms/feature-extraction

[2] Vision Online Marketing Team. What is Visual SLAM Technology and What is it Used For? Available from: https://www.visiononline.org/blog-article.cfm/What-is-Visual-SLAM-Technology-and-What-is-it-Used-For/99

[3] Yousif, K., Bab-Hadiashar, A. & Hoseinnezhad, R. Intell Ind Syst (2015) 1: 289. Available from: https://doi.org/10.1007/s40903-015-0032-7

[4] Kwon KS, Ready S. Practical Guide to Machine Vision Software: An Introduction with LabVIEW. Berlin: John Wiley & Sons, Incorporated; 2014.

[5] KinectFusion: Real-Time Dense Surface Mapping and Tracking. Available from: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/ismar2011.pdf

[6] Point Cloud Library, Project. Available from: https://github.com/PointCloudLibrary/pcl

[7] StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments. Raluca Scona, Mariano Jaimez, Yvan R. Petillot, Maurice Fallon, Daniel Cremers. IEEE International Conference on Robotics and Automation (ICRA) 2018. Available from: http://www.robots.ox.ac.uk/~mobile/Papers/2018ICRA_scona.pdf

[8] Dynamic Fusion, Project. Available from: https://github.com/mihaibujanca/dynamicfusion

[9] OpenChisel, Project. Available from: https://github.com/personalrobotics/OpenChisel