BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF CONTROL AND INSTRUMENTATION
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

METHODS FOR SIMULTANEOUS SELF-LOCALIZATION AND MAPPING FOR DEPHT CAMERAS

METODY SOUČASNÉ SEBELOKALIZACE A MAPOVÁNÍ PRO HLOUBKOVÉ KAMERY

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR AUTOR PRÁCE	Bc. Adam Ligocki
SUPERVISOR VEDOUCÍ PRÁCE	prof. Ing. Luděk Žalud, Ph.D.

BRNO 2017

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Master's Thesis

Master's study field **Cybernetics, Control and Measurements**

Department of Control and Instrumentation

Student: Bc. Adam Ligocki ***ID:*** 154791

Year of study: 2 ***Academic year:*** 2016/17

TITLE OF THESIS:

Methods for Simultaneous Self-localization and Mapping for Depht Cameras

INSTRUCTION:

1. Get familiar with state-of-art methods of simultaneous self localization and mapping (SLAM) using RGBD cameras.
2. Choose one of previously described open-source projects from semestral thesis and modify it for using with available RGBD camera.
3. Extend the chosen project with the possibility of adding the external data with the position and the rotation of the RGDB camera, all in run-time.
4. Evaluate the program in pre-defined conditions with and withouth added position and rotation information.

REFERENCE:

R. A. Newcombe et al., "KinectFusion: Real-time dense surface mapping and tracking," 2011 10th IEEE International Symposium on Mixed and Augmented Reality, Basel, 2011, pp. 127-136.
doi: 10.1109/ISMAR.2011.6092378

Assigment deadline: 6. 2. 2017 ***Submission deadline:*** 15.5.2017

Head of thesis: prof. Ing. Luděk Žalud, Ph.D.
Consultant:

doc. Ing. Václav Jirsík, CSc.
Subject Council chairman

ABSTRACT

This Master's thesis deals with existing visual SLAM and wheel odometry data fusion. The result of this data connection is the possibility of suppressing measurement error of each position estimation method and creating more accurate 3D model of examined environment. At the beginning this thesis is aiming on theoretical principles that are necessary to deal with 3D SLAM.Next, the features of used open source SLAM project and its modifications are described. Then the principles of visual and wheel odometry data fusion are explained, followed by specification of differential chassis used for odometry. In conclusion, the thesis summarises the results obtained by data fusion and compares them with the original accuracy of visual SLAM.

KEYWORDS

RGBD camera, SLAM, Kinect, Odometry, Data fusion

ABSTRAKT

Tato diplomová práce se zabývá tvorbou fúze pozičních dat z existující realtimové implementace vizuálního SLAMu a kolové odometrie. Výsledkem spojení dat je potlačení nežádoucích chyb u každé ze zmíněných metod měření, díky čemuž je možné vytvořit přesnější 3D model zkoumaného prostředí. Práce nejprve uvádí teorií potřebnou pro zvládnutí problematiky 3D SLAMu. Dále popisuje vlastnosti použitého open source SLAM projektu a jeho jednotlivé softwarové úpravy. Následně popisuje principy spojení pozičních informací získaných vizuálními a odometrickými snímači, dále uvádí popis diferenciálního podvozku, který byl použit pro tvorbu kolové odometrie. Na závěr práce shrnuje výsledky dosažené datovou fúzí a srovnává je s původní přesností vizuálního SLAMu.

KLÍČOVÁ SLOVA

RGBD kamera, SLAM, Kinect, Odometrie, Fůze dat

LIGOCKI, Adam. *Methods for Simultaneous Self-localization and Mapping for Depht Cameras*. Brno, 2017, 81 p. Master's Thesis. Brno University of Technology, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Advised by prof. Ing. Luděk Žalud, Ph.D.

Vysázeno pomocí balíčku `thesis` verze 2.63; `http://latex.feec.vutbr.cz`

DECLARATION

I declare that I have written the Master's Thesis titled "Methods for Simultaneous Self-localization and Mapping for Depht Cameras" independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the thesis and listed in the comprehensive bibliography at the end of the thesis.

As the author I furthermore declare that, with respect to the creation of this Master's Thesis, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno ....

author's signature

ACKNOWLEDGEMENT

I would also like to acknowledge prof. Žalud of the Faculty of Electrical Engineering and Communication at Brno University of Technology as the second reader of this thesis, and I am gratefully indebted to him for his very valuable comments on this thesis.
Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.
Thank you.

Brno . . . .

author's signature

CONTENTS

LIST OF FIGURES

1 INTRODUCION

In these days, there are many ways how to navigate robots in outdoor environment, for example satellite systems such as GPS, Galileo, Glonas or their assisted versions with utilisation of cellular phones network or Wi-Fi based localization. All these systems can determinate robot's position with accuracy up to few meters. With access to commercial version of GPS with higher accuracy and special functions like application of differential methods, we are able to estimate global position with centimetre precision.

For indoor navigation, with no access to satellite's signal, there have been many Visual SLAM (Simultaneous Localization and Mapping) techniques were developed already. These implementations aim at creation of the type of artificial intelligence, which would be able to see and understand its surroundings with usage of inexpensive cameras, for example RGBD ones. The output of this algorithm is real-time created 3D model of robot's environment and its estimated position inside this model.

This technology has a huge potential for future usage in automation and robotics. We can already see first practical implementations in real live. As example let's mention Google Self Driving Car Project [1] and Tesla Autopilot[2]. Those technologies are strongly based on techniques which are continuously scanning its nearby surroundings, trying to understand it.

Current state-of-art technology still have strongly limited range of application. Today's implementations are able to map areas on scale up to few hundred square meters. The most dramatic problem is that with increasing size of mapped area there is directly proportional amount of data that computer must manage.

The aim of this thesis is the extension of existing solution for pure visual odometry by supplying data from more accurate sensors mounted on the mobile robot and trying to create mobile unit, which would be able to explore unknown environment with higher precision than then original solution.

2 PREVIOUS WORK

My term paper, which preceded this Master's thesis, researched available SLAM technologies by examining its selected, existing, open source implementations, with focus on potential future experimentation and improvements to rescue robotics system Cassandra [3] developed by robotics team on Faculty of Electrical Engineering in Brno.

The beginning described the basics of self-localisation and mapping techniques, which are used in current state-of-art solutions. The reader was introduced to graph-based techniques of modelling robot's environment, 3D model storage methods and iterative algorithms for resolving current position. All those topics were later used for understanding, how do complex SLAM solutions work.

Large part of term paper focused on research of currently available RGB-D cameras, their types, its working principles, with special attention to accuracy, which is crucial for creating reliable models. As example lets mention Kinect v1, Swiss Ranger SR4000 and ZED Stereo Camera. All of them work by different principles. Every device has list of build-in technologies and all those technologies were analysed.

Previous research than explored existing RGB-D SLAM implementations, and their underlying concepts. At the beginning five open source projects (Kinect Fusion [4], Kintinuous [5], Elastic Fusion [6], Dense Visual SLAM [7] and RGBDSLAM [8]) and their papers were studied for deeper understanding how they work and how accurate they can be. Based on this research two of these algorithms (Kintinuous and Elastic Fusion) have been chosen for next stage where their source codes have been reviewed and Elastic Fusion was selected as target for the future research and extension by this thesis.

At the end of previous work, I proposed few ways of how I would be able to improve Elastic Fusion algorithm while running it on PC, which would be mounted on wheeled robot Orpheus [9] with available odometry and inertial unit. Basic idea was to fuse Elastic Fusion's visual odometry, Orpheus's wheel odometry and inertial unit. All those methods have different source of noise and uncertainty. This fact could be used to crate filters which would be able to refine visual odometry position estimation and this way improve accuracy of the whole 3D environment model creation process.

3 SLAM METHODS

In this chapter, there is described basic theoretical background of these days the most frequently used algorithms and mathematical tools which are applied for solving SLAM problems.

3.1 Simultanious Localization and Mapping (SLAM)

When we are talking about SLAM problem, there are two parallel problems we have to solve. First of them is to create 3D model of surrounding environment around our agent (robot) and the second is to localize our agent in this model. There we can notice contradictory of these two requests. We are not able to create oriented graph model of environment without knowledge about current position and localization can't be done without previously mentions map.

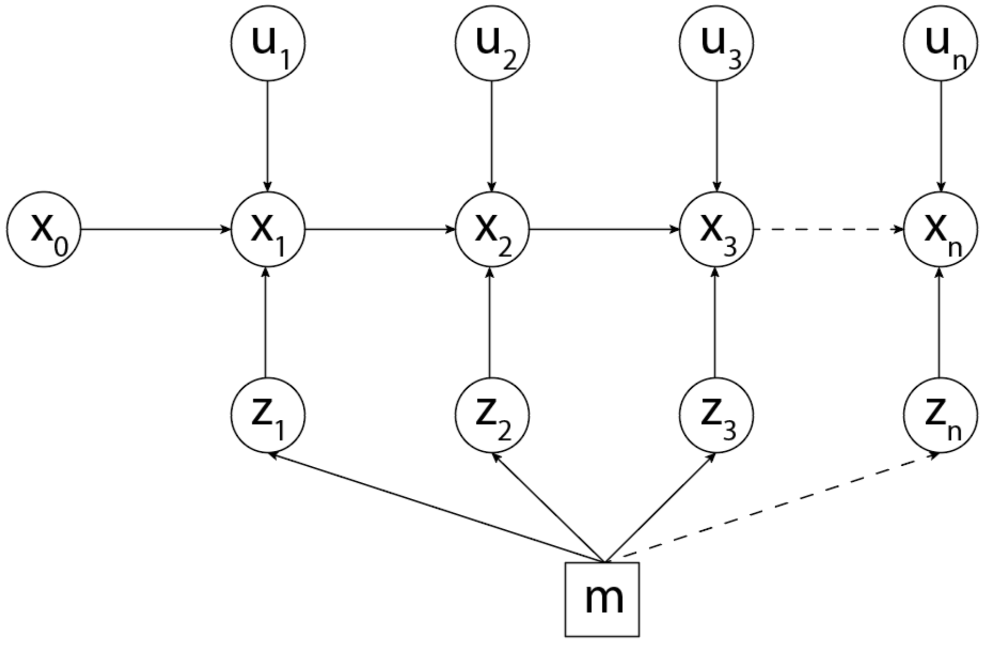

Fig. 3.1: Schematic representation of SLAM task

Let's assume that at the beginning robot is placed to unknown environment at position x_0. After that robot will act with control input $u0$ on itself and consequently its position will change to new state $x1$. Then robot perform measurements $z1$ of its surrounding which gives its new advanced information about current position state. The entire solution of SLAM problem is to iteratively search for the most probabilistic solution, which would report the least differential error between measured and expected surrounding.

$$p(x_{1:N}, m|z_{1:N}, u_{1:N}, x_0) \tag{3.1}$$

Today we can find two most frequently used approaches to map model construction. First of them is so-called "Landmark mapping". We can imagine it as a map defined with few very high contrast marks, which are placed in mapped environment, for example QR codes sticked on walls and corners. When robot notices one or more of these landmarks, it can calculate distance to each one and determinate its current position inside current model.

The second method is so-called "Dense SLAM". This kind of map is formed by large number of points, when each one of them has its own 3D coordinates position. This way of space representation is called Point Cloud. Typically, this data format come out for example from LIDAR sensors. In this model representation technique robot performs its localization by correlating model m and current measurement z_n. The position with the least disparity is most probably current in-map position.

3.1.1 Graph Based SLAM (Pose Graph)

Idea of Graph Based SLAM is based on graph construction, in which there are custom robot positions x_n during the time t represented as graph nodes, just as landmarks do. The connections (graph edges) between these nodes are representing their mutual relations [10].

Let's place robot at the starting position and let's call this position the zero coordinates. At the same time this position will become our first graph node. Then the robot will perform movement to a new position and if this new state meets strictly defined conditions like minimal distance from nearest node or large uncertainty of current position, then this state is added to graph as a new node.

This way constructed structure is very inaccurate. It is caused by control input uncertainty and cumulative error of position estimation based on numerical models which robot has implemented in his algorithms. But this error can be suppressed or at least minimized via tracking robot's environment and continuously correcting its position. This is done by introducing a new information to previously mentioned graph by adding nodes which represent high important marks from surrounding. As mentioned before, these marks could be for example high contrast elements. When robot explores some new environment, it can combine information from known control input and surrounding observations and is able to create solid network based on

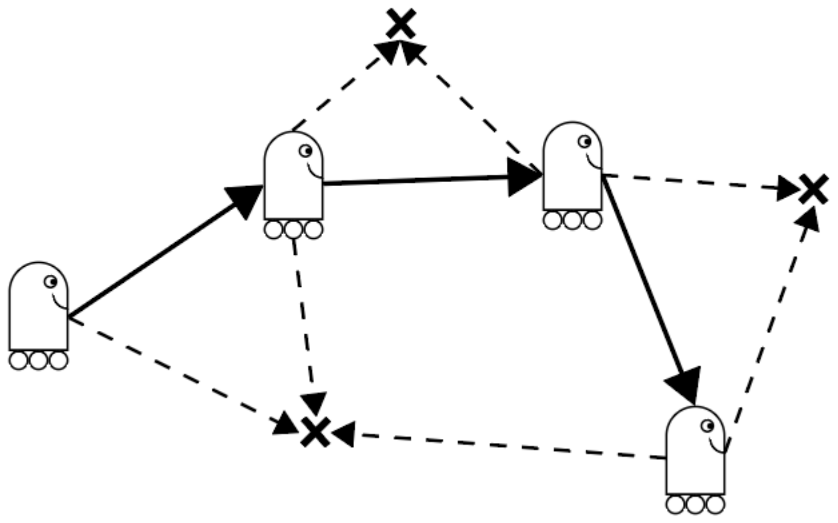

Fig. 3.2: Graph SLAM schematic. Robot's position, landmarks and relations between them creates solid environment representation graph

nodes and their mutual relations. Then, when robot takes some new measurements, it can quite easily determinate most probabilistic current position state.

$$p(z_t|x_t, m_t) \tag{3.2}$$

Equation expresses, that measurement output z_t is the function of current position x_t and current model m_t.

$$p(x_{1:T}|z_{1:T}, u_{1:T}) \tag{3.3}$$

Next equation expresses, that agent's position x is given by measurement z and control input u

So let's define vector $x = [x_0, x_1, ..., n_n]$, which represents set of all reference robot's positions (nodes) in estimated graph. Next let's have z_{ij} and $Omega_{ij}$, which are mean value and information matrix of measurements between two positions i and j. Further $\hat{z}_{ij}(x_i, x_j)$ is a prediction of relations between two nodes. Then likelihood of measurement z_{ij} can be estimated as

$$l_{ij} = [z_{ij} - \hat{z}_{ij}(x_i, x_j)]^T \Omega_{ij} [z_{ij} - \hat{z}_{ij}(x_i, x_j)] \tag{3.4}$$

In this equation we bring following substitution

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j) \tag{3.5}$$

and reduce equation into

$$l_{ij} = e_{ij}^T \Omega_{ij} e_{ij} \tag{3.6}$$

Now we can estimate most likelihood robot's position in graph as configuration of graph nodes and edges with minimised sum of all errors between individual nodes.

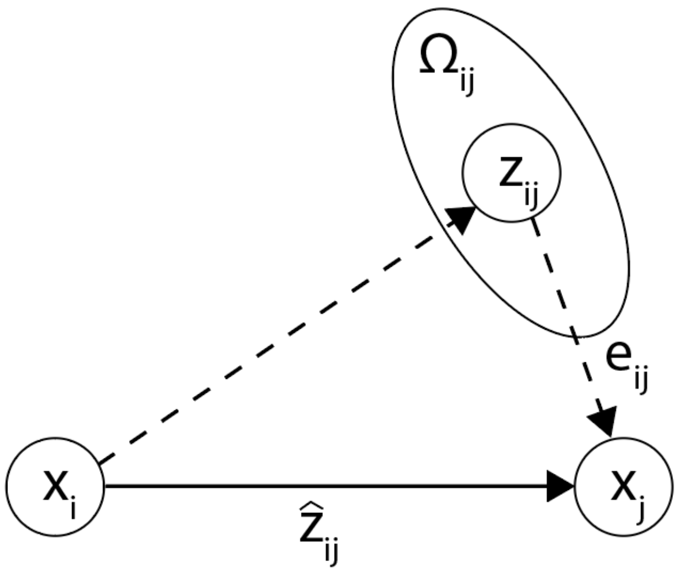

Fig. 3.3: Error of measured and estimated position

3.1.2 Dense SLAM

It is the most frequently used SLAM technique in these days, when the environment model is represented as Point Cloud, therefore set of space placed points which approximates explored space. Commonly agent localized itself in this map by correlating its current observation and map, both in point cloud, and iteratively searching for most probabilistic translation and rotation containing solution. If position is estimated, current map can be expanded with new information about previously unseen places.

3.1.3 Bundle Adjustment

Bundle adjustment is method for purposes of 3D objects reconstruction from multiple images, which was taken from different places. The most frequently it is used for static object 3D model creation.

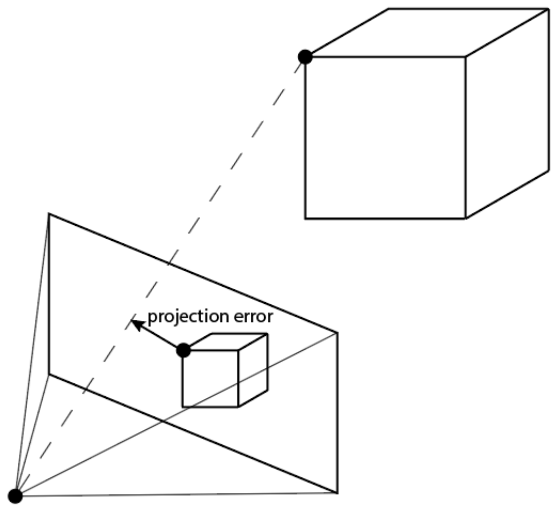

Fig. 3.4: Graphic representation of real 3D point reflection to image plane error

Let's imagine situation, when we have multiple images of some object, but the camera calibration parameters (intrinsic matrix) and its position and orientation (extrinsic matrix) are both unknown. This method is estimating this previously mentioned internal and external image capture parameters and in the next step attempts to perform 3D position calculation of all possible captured positions with minimisation of declared loss function.

$$Err(\mathbf{v}, \mathbf{a}, \mathbf{b}, \mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij} d(\mathbf{Q}(a_j, b_i), x_{ij})^2 \tag{3.7}$$

$$\min_{a,b}(Err(\mathbf{v}, \mathbf{a}, \mathbf{b}, \mathbf{x})) \tag{3.8}$$

where $\mathbf{Q}(a_j, b_i)$ expresses position of i-th point reflection to j-th plane therefore expected i-th pixel position on j-th image. a_j represents camera calibration matrix and b_i is point spatial position in reconstructed model. Function $d(\mathbf{Q}, x)$ calculates Euclidean distance between estimated $\mathbf{Q}$ and ground true position x_{ij}. v_{ij} express Boolean value, whether i-th point is present on j-th image.

Instead of real-time SLAM algorithms implementations, bundle adjustment is post-processing method. This means that it is performed after all input data has been collected. Nevertheless, some SLAM implementations use bundle adjustment for in runtime accuracy improvement.

Generally, bundle adjustment has higher computational cost, but gives better results.

Example of bundle adjustment on RGB-D data is described in [11]

3.2 Space Representation

This chapter describes few ways of how to represent and keep saved collected data about environment, that robot has just explored.

3.2.1 Point Cloud

Generally, Point Cloud (PC) is a term that in field of IT and computer graphics expresses a very large set of points which are placed to some coordinate system. When talking about SLAM, Point Cloud is used to represent created 3D (X, Y and Z axis) approximation based on non-volume points, that represents environment that agent just discovered.

This way of space representation is very typical for example for LIDAR output. It performs its measurement in way that distance of nearest obstacle is measured in predefined grid. The output is the set of N spatial placed points that approximate surrounding.

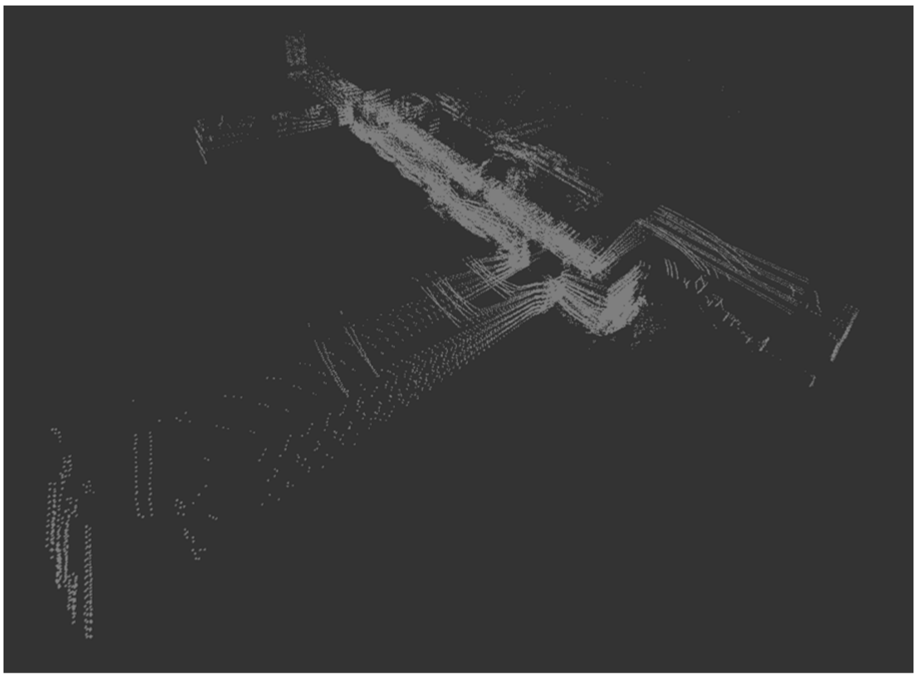
Fig. 3.5: Example of Point Cloud corridor representation

In our case Point Cloud is created by reflecting depth pixels from RGBD camera. The set of these points represents currently present obstacles in robot's proximity.

To deal with this data format there is available open source library called Point Cloud Library [12], which provides large scale of available operations over this data format.

3.2.2 Octree

It is hierarchical representation of space occupancy. The space is at the beginning represented as three-dimensional cube, which says about itself if it is empty, partly or fully occupied. If it is occupied only partly we can divide it into eight sub-cubes which are described as the parent one does. This algorithm can recurrently go on to very small structures and in this way, create approximation of 3D space occupancy.

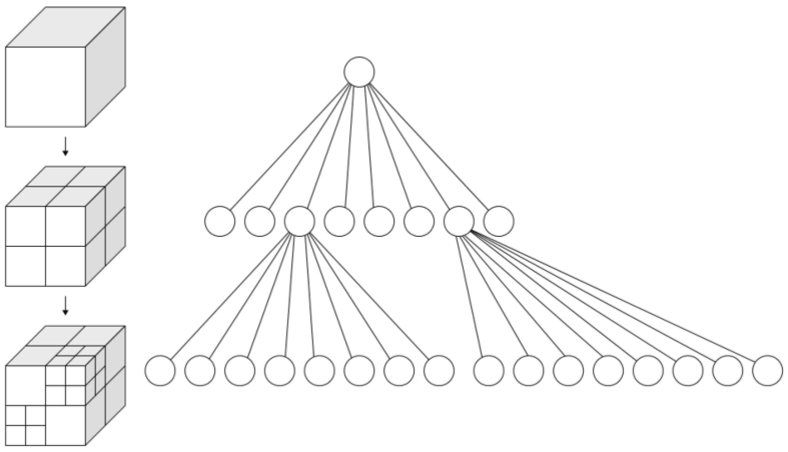

Fig. 3.6: Hierarchical representation of Octree structure [13]

In practice [14] this data format can be represented as linear array which is divided into two bit strips. Every two bits say about occupancy of appropriate cube. If these two bits keep the information, that cube is empty, or if information says that cube if totally occupied algorithm can declare that this part of space as fully defined and can continue to the next cube. But if cube is only partly occupied algorithm will divide it to eight sub-cubes and will resolve its states recursively.

3.3 Other SLAM Terms and Techniques

3.3.1 Keyframe

In SLAM terminology keyframe is called the camera video frame which fulfilled the criterial conditions and become new node in constructed oriented graph. In this way. this frame and its positions are going to be the reference for future position estimation. The mentioned criterial condition to call frame a "keyframe" could be for example defined distance from nearest other keyframe, or for example each N-th frame in scan series can be proclaimed as keyframe or for example when algorithm does not have enough referential points in current surrounding, the new one reference in form of current frame can be introduced to graph structure.

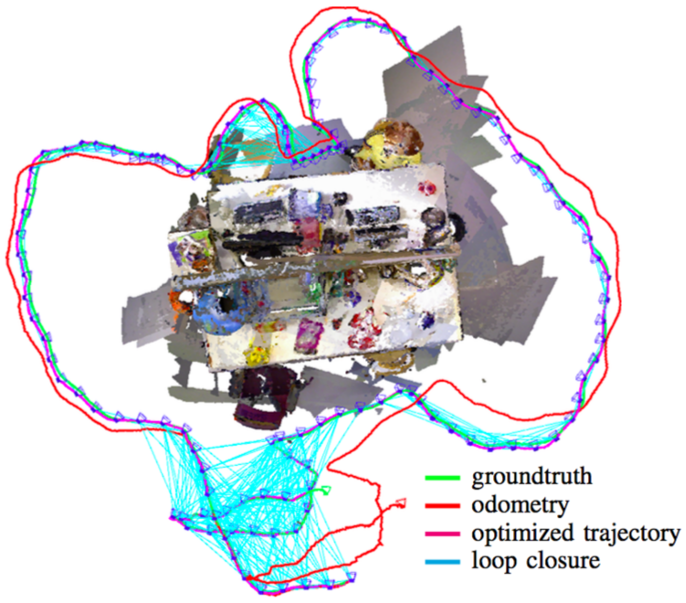

Fig. 3.7: Keyframe is commonly represented as camera flustrum (blue) on its path through unknown space during time t [7]

3.3.2 Iterative Closest Point (ICP)

This method [15] is used to find the best fit transformation matrix with six degrees of freedom (three for rotation and three for translation) between two Point Clouds. First point cloud set (map) is static and the second one (current scan output) is iteratively transformed in way to get the lowest possible disparity between this two sets.

In each iteration the algorithm performs searching for the nearest neighbour in set B for each point from set A. Then the total cumulative distance between this two sets is estimated and based on derivation of this cumulative distance sum, transformation for the next iterations is predicated. This cycle is performed until final condition is met or number of iterations reach defined value.

The main disadvantage of this algorithm is, that it can get stuck in local minimum of loss function. Because of this, it is not suitable for global model positioning estimation, rather it is used for local position estimation correction.

3.3.3 Loop Close

During SLA graph construction, there is continuous accumulation of position estimation error caused by inaccuracy visual odometry techniques. This results into effect that when robot crosses position which was already visited by it in the past, these two identical positions in real space are in oriented graph represented as two different nodes. Then position estimation according to this inconsistent graph is not precise because one real space position is represented in graph twice.

But if we define criterion that two graph nodes are identical and can be merged if some similarity condition is met, position graph can be deformed and previously mentioned disparity can be removed This way graph consistency could be improved, cumulative position estimation error can be removed and robot can explore unknown area on larger range.

Of course, during graph deformation, we merge two graph nodes into one (the newer one to older one, because we can expect that older one has smaller cumulative position estimation error) but also relations between all other connected nodes must be adjusted. This process is called "Graph Relaxation" or "Graph Optimization"

There are many approaches to graph relaxation. Let's mention for example methods [16], [17] where nodes are connected between each other and their current positions are represented by Gaussian distribution probability. If one node position is changed, its shift is distributed via entire graph by modifying neighbor's position probability function.

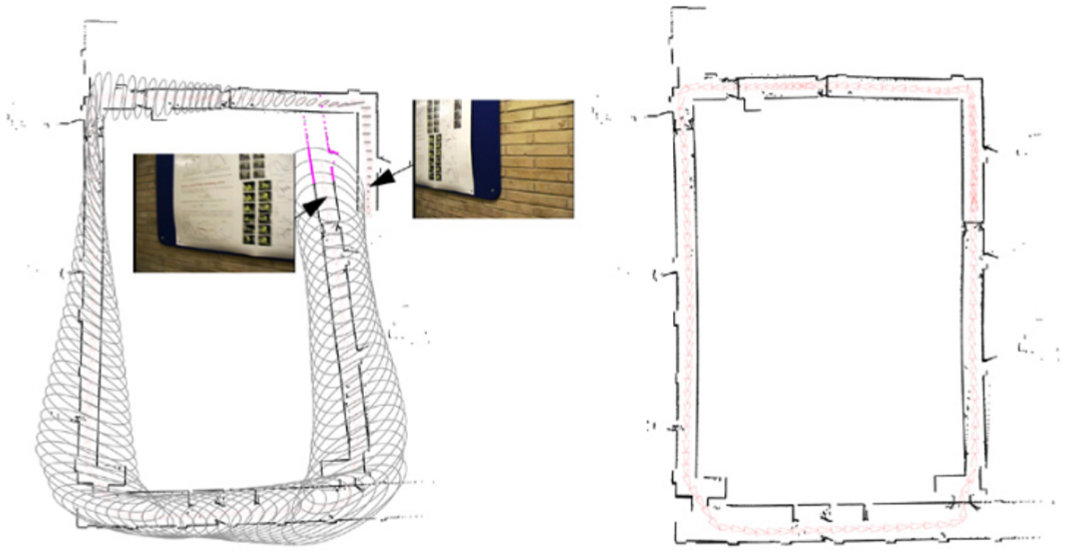
Fig. 3.8: Example of Loop Close and Graph Relaxation [17]

3.3.4 RANSAC

RANSAC [18] is iterative method for unknown model estimation over some data set, which contains large number of so-called "outliers". These are elements which do not fit to expected model. On the other site elements which suite well are called "inlayers".

Input into this method is set of M points, which had some mathematically defined relation.

At the beginning algorithm choose N elements, where N is minimal number of input data to resolve all model's degrees of freedom. The model is estimated.

In next step model is tested on entire set on input data. Each input data element is examinate if it fits model or not (is inlayer or outliers) with threshold μ. If percentage of inlayers is larger, then K where $0 < K \leq 1$, model can be called credible and can be accepted. If model do not fit for enough number of elements, new iteration will be performed.

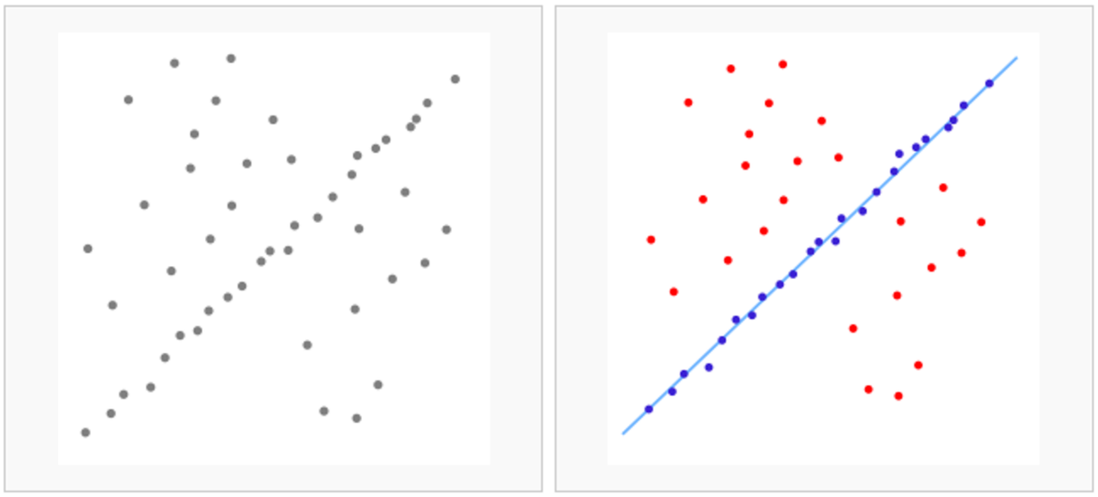

Fig. 3.9: This image shows model finding between large outlier noise in input data. The output model (blue) is inert for outliers (red) [19]

Disadvantage of this method is that it does not guarantee model estimation with one hundred percentage certainty. There is always probability that model has been estimated for at least one outliers and so it is not trustworthy.

During algorithm initialization, we can setup value L, which says how many times RANSAC will iterate previously described process so the probability of right model finding can be at least 0.99.

$$1 - p = (1 - u^m)^l \tag{3.9}$$

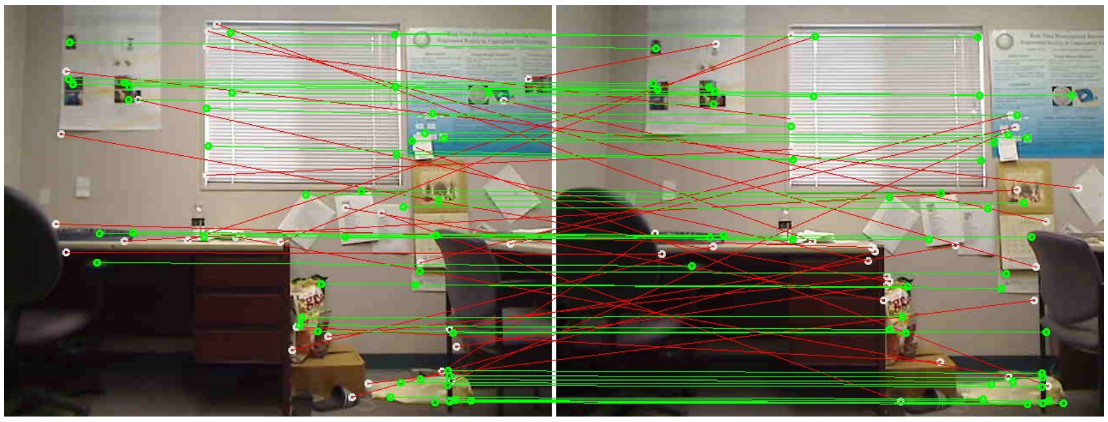

Fig. 3.10: Example of inliers (green) and outliers (red) during parity finding in between two images of same scene by RUNSUC method [20]

after formula modification

$$L = \frac{log(1-p)}{log(1-(1-v)^m)} \tag{3.10}$$

where L is number of iterations, p is probability of right model finding, u is probability that randomly picked element from dataset is inlayer, v is probability that randomly selected element is outlier.

$$u = 1 - v \tag{3.11}$$

3.3.5 Signed Distance Function (SDF)

In computer graphics field, it is common that more complex objects are represented as a set of many space oriented triangles. But this technique does not fit very well for example for spherical objects approximation. In first case, we need to create large number of polygons, which increate rendering cost and in second case even this large number of triangles do not fit true shape very well.

In this case, we can represent these objects for example with three-dimensional function, which will return negative values for point inside it, zero values for points on the surface of object and positive values for entire space beyond object surface.

Let's imagine function for sphere

$$A = f_{ball}(\mathbf{p}, \mathbf{c}, r) = \begin{cases} A = -1, & prod(\mathbf{p}, \mathbf{c}) < r \\ A = 0, & prod(\mathbf{p}, \mathbf{c}) = r \\ A = 1, & prod(\mathbf{p}, \mathbf{c}) > r \end{cases} \tag{3.12}$$

where **p** is examined spatial point, **c** is sphere center, r is sphere radius and d is Euclidian distance between points **p** and **c**. Next

$$d = \sqrt[s]{\sum_{i=1}^{N}(p_i - c_i)^s} \tag{3.13}$$

where s is number of dimensions for which function is defined.

More complex shapes can be represented by merging multiple simple shapes together. For example intersection of two shapes can be expressed by *max* function

$$max(f_{ball1}(\mathbf{p}, \mathbf{c_1}, r_1), f_{ball2}(\mathbf{p}, \mathbf{c}_2, r_2)) \tag{3.14}$$

and union of two shapes can be expressed by *min* function.

$$min(f_{ball1}(\mathbf{p}, \mathbf{c_1}, r_1), f_{ball2}(\mathbf{p}, \mathbf{c}_2, r_2)) \tag{3.15}$$

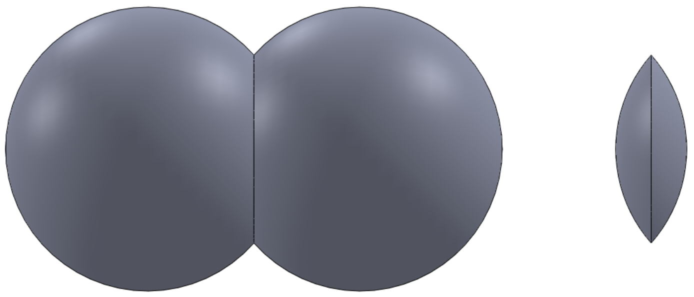

Fig. 3.11: Result of two spheres merging. Left-side minimum (union), right-side maximum (intersection)

If we apply this technique for in SLAM problem solution, we can achieve lower memory requirements for model storage, because thousands of points in point cloud can be expressed by few equations, but most interesting is noise reduction, because we can generalise rugged areas via approximating it by some soft shape.

4 RGBD CAMERAS

This thesis is very strongly focusing on one of the most frequently used solution for visual SLAMs which is RGBD camera usage. These devices are scanning their surrounding in two domains. First one sensor is common CCD/CMOS which scanning surrounding environment in color space (typically RGB). The second one is measuring distance from device to near obstacles. The output signal is 2D matrix with values of distances in defined field of view.

The best-known representative in this field is well known Kinect device developed by Microsoft and PrimeSense.

This device has been primarily focusing on game industry, but because of its low price and relatively good quality of data output has been quickly adapted in other lines like research and industry. In these days, Kinect with its skeleton detection algorithms is often used for medical purposes during patient's therapies.

Microsoft corporation parallel with Kinect v1 has also released software development toolbox (SDK) with many build in functions which set foundations for future very fast expansion of many other interesting implementations and problem solutions with this new device. Let's mention for example the Kinect Fusion, which was the first implementation of SLAM algorithm on this platform. Furthermore, we can mention PrimeSense's OpenNI project, which was open source toolbox for RGBD devices data processing. Next one for example NiTE project which was focusing on hand gestures and human skeleton movement processing.

Further the basic principles of distance measurements are described later in this chapter. Then reader is introduced to brief summaries of these days available RGBD devices and at the end of the chapter there is described the mathematical theory about camera physics and algorithms user for image correction purposes.

4.1 Active Triangulation

This is the simplest way of visual distance measurement. The principle is based on active illumination of measured object at a given angle by for example point laser.

This resulting reflection is then captured by the CCD chip. Distance of measurement object is then expressed by function of the position of reflected point on the sensor chip.

For thus formed structure, if we know the parameters describing the position of the laser source and receiver sensor, the triangulation triangle is established, then within the basic knowledge of the geometry laws, we are able to calculate the

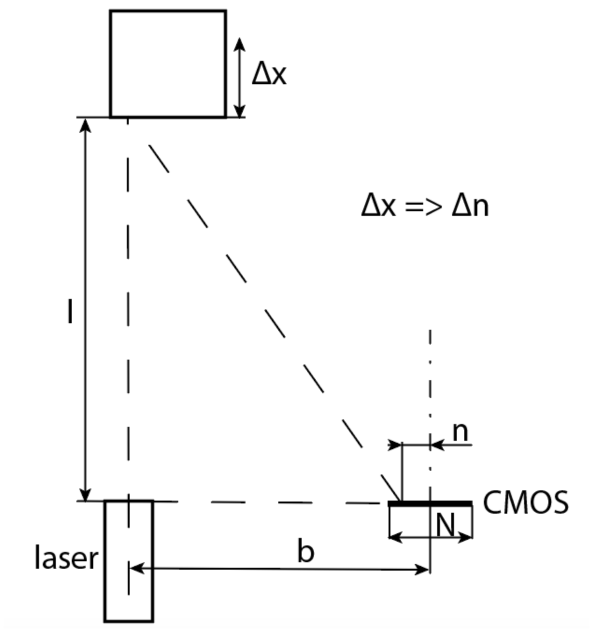

Fig. 4.1: Geometry of active triangulation

distance between the gauge's plane and measured object.

$$l = \frac{size_{CCD} * n}{N} \tag{4.1}$$

where l is measured object distance from gauges plane, $size_{CCD}$ is size of CCD sensor row, n is row index of illuminated pixel and N is total number of pixels in one sensor's row.

4.2 Passive Triangulation

During three-dimensional scene to CCD chip plane projection the depth information is lost. But this information can be restored by scanning scene multiple times from several different positions. If we know the relationships between these positions in which the pictures had been taken, for each point which was captured multiple times we can use epipolar geometry to calculate the spatial coordinates.

We can distinguish three scenarios:

- More static cameras with known position and orientation
- More static cameras with auto calibration
- One dynamic camera with auto calibration

In case of SLAM problem solutions, the most frequently used solution in the one with two cameras mounted on solid bar in way, that both camera's optical axis are parallel.

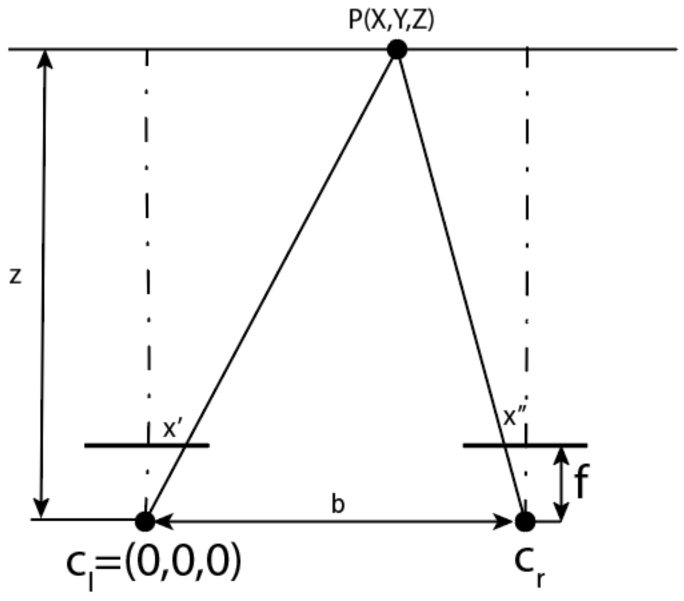

Fig. 4.2: Geometry of passive triangulation

In this configuration, each camera is watching the object from different angle. The angle between two beams of the point projection in the plane of the chip is called the parallax angle. To maintain the accuracy of measurement of the angular parallax should not drop below a certain limit. In practice, it is stated that stereovision can be used within thirty times the distance between the optical centers of both cameras.

The calculation of spatial coordinates of the point projected on both sensors is expressed as

$$p = x' - x'' \tag{4.2}$$

where p is parallax angle and x' and x'' are indexes of pixels which captured measured point.

The calculation of x, y and z coordinates are performed according to following equations

$$X = x'\frac{b}{p} \tag{4.3}$$

$$Y = y'\frac{b}{p} \tag{4.4}$$

$$Z = f\frac{b}{p} \tag{4.5}$$

where b is the triangulation triangle base, which mean distance between cameras optical centers, f if focal length of the optical systems and p is previously calculated parallax

These simple relations are valid only under the condition that optical axes of both sensors are parallel, while the cameras are spatially oriented identically. If this condition is not meet, it is necessary to perform correction calculations and perform image rectification. This problem is further discussed in chapter focused on camera calibration.

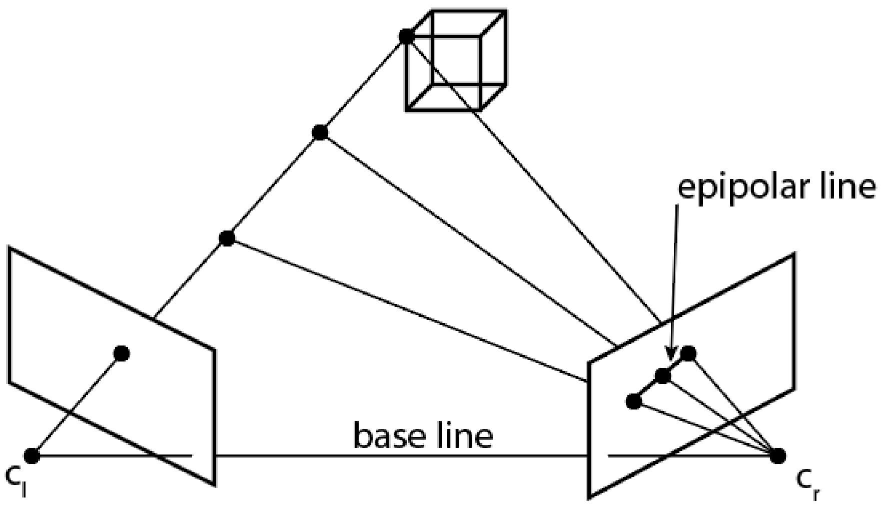

Fig. 4.3: Epipolar geometry between two planes

The searching for corresponding points can be simplified by epipolar line estimation. It says, that point projected on one sensor may be reflected on the other sensor anywhere within the defined line.

4.3 Time of Flight Principle

Time of Flight method (ToF) is generally based on the principle of measuring time during which particle or acoustic or electromagnetic wave travels an unknown distance between the transmitter and receiver. Based on knowledge of particle or wave velocity we are able to calculate unknown distance.

In practice, we meet with several types of ToF instruments, working on different physical principles. Let's mention for example ultrasonic sensors, which generates sequence of pulses by piezoelectric crystal, with frequency approximately 40 kHz. And it measures time it takes the wave to travel from ransmitter, reflect on the obstacle and return to the receiver module, which works on the same principle as transmitter do but in reverse direction. With knowledge of measured time, we can estimate distance with equation.

$$2s = v_{mech} * \Delta t \tag{4.6}$$

where s is distance between module and obstacle, v_{mech} is the propagation speed of mechanical waves in the environment and Δ is the time between signal transmission and return.

Another variation of ToF are the optical sensors. They are divided into two fundamental groups.

The first one is a simple pulse transmitter. As ultrasound transmits signal against obstacle, short laser pulses with wavelength of 850nm, that are invisible for human eye, are sent against environment and time to echo is measured. In fact, to simplify measurement there is measured phase shifting between transmitted and received pulses. Considering that in this case we work with the electromagnetic waves, which propagates through atmosphere with speed close to speed of light in vacuum, this technique is extremely demanding on very precise time measurements and very fast semiconductor devices that generate and receive the laser beams.

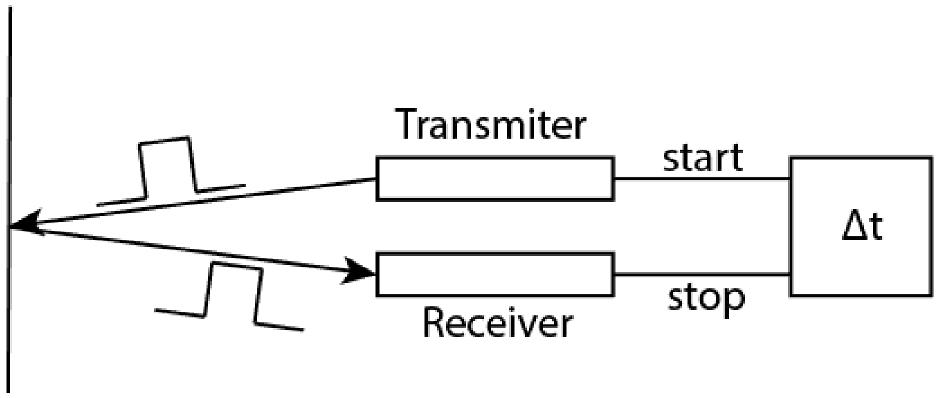

Fig. 4.4: Distance measurement with pulse

Each pixel of depth camera based on this principle contains two timers. The scene is at the beginning of measurement illuminated by short pulse and when the reflection returns, the first timer is activated (S1). When the transmitted pulse is terminated, the first timer is also stopped and the second one is activated until the

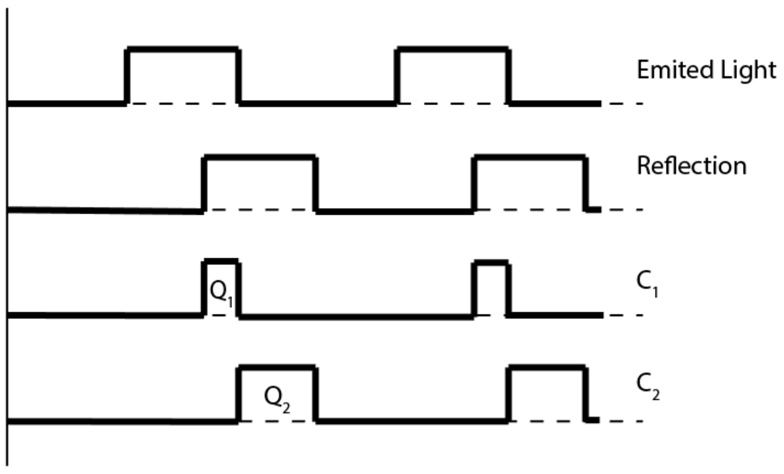

Fig. 4.5: Distance calculation from pulse phase shift

end of reflection. With usage of these two measured time intervals we can estimate distance as follows

$$s = \frac{1}{2}c * \Delta t \left(\frac{S2}{S1 + S2}\right) \tag{4.7}$$

where d is measured distance, c is speed of light in environment, Δt is light emission interval and $S1$ and $S2$ are two time intervals measured with pixel's timers.

The second possible method is to perform measurements with usage of modulated continuous infrared signal.

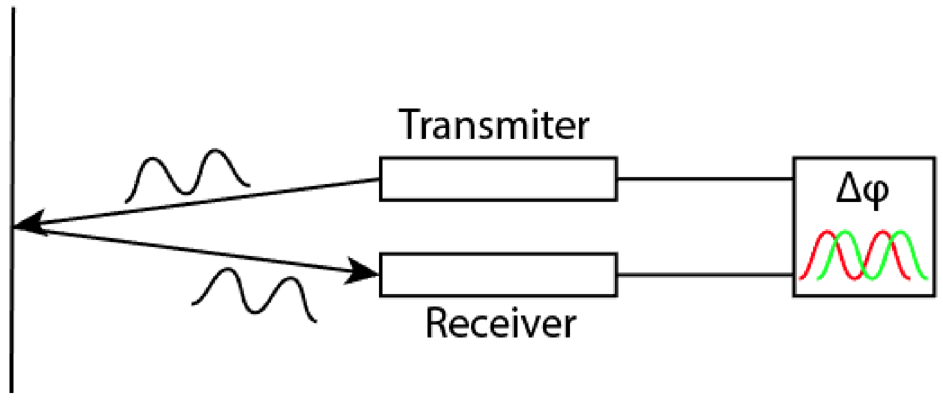

Fig. 4.6: Continuous signal distance measurement

The sensor continuously illuminates scene with sinusoid intensity signal with period about 10MHz. Each pixel in the sensor performs periodical sampling of reflected signal and using cross-correlation, it is able to estimate time delay by using information about phase shift and speed of signal in environment.

$$d = \frac{c}{4\pi\omega}\phi \tag{4.8}$$

where d id measured distance, c is speed of light, ω is modulation frequency and ϕ is phase shift.

Specifically, the calculation process of phase shift with mathematical expression takes place as follows.

The sensor generates a signal

$$g(t) = cos(\omega t) \tag{4.9}$$

after the reflection signal returns

$$s(t) = b + a * cos(\omega t + \phi) \tag{4.10}$$

where b is background illumination, a is reduced amplitude of returned signal and ϕ is phase shift between transmitted and received signal.

Following the cross correlation

$$c(\tau) = s * g = \int_{-\infty}^{\infty} s(t) * g(t + \tau)dt \tag{4.11}$$

where τ is correlation offset

We are able to reduce this correlation expression to four sampled elements $i = 4$.

$$c(\tau) = fraca2cos(\omega\tau + \phi) + b \tag{4.12}$$

$$A_i = c(i * \frac{\pi}{2}), \quad i = 0, \ldots, 3 \tag{4.13}$$

where final phase shift could be estimated as

$$\phi = arctan2(A_3 - A_1, A_0 - A_2) \tag{4.14}$$

and amplitude of incoming signal is

$$a = \frac{1}{2}\sqrt{(A_3 - A_1)^2 + (A_0 - A_2)^2} \tag{4.15}$$

Whether the second method seems to be more complex, but it has a significant advantage, as we can assume from equations above, the modulated signal can be separated form background offset.

The attentive reader would notice that this method is limited by modulated signal wavelength. If Δt is longer than one wave period, the measured distance will overflow back to zero. This effect can be removed by using second modulation frequency parallel with first one.

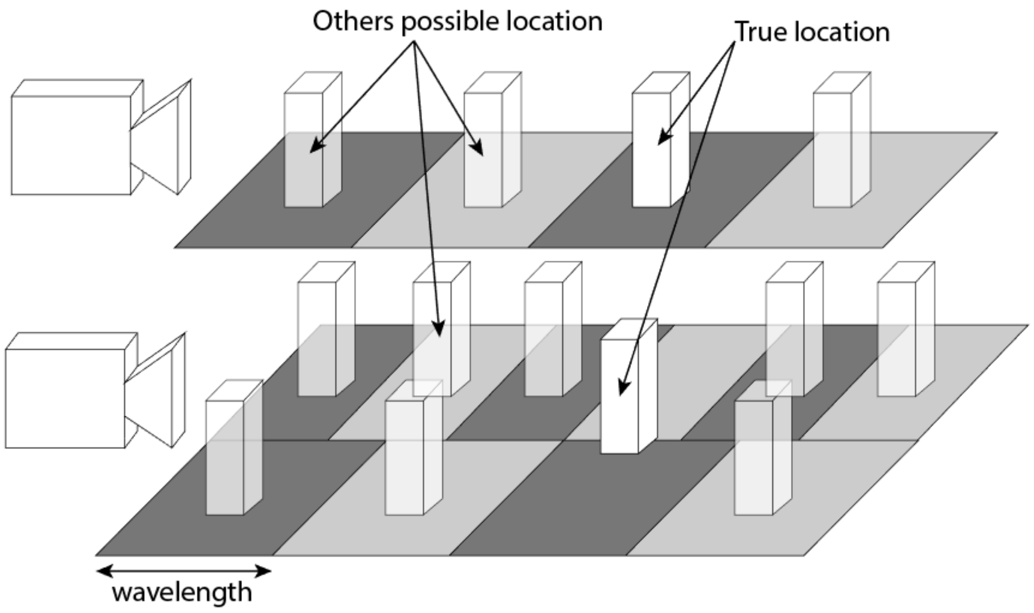

Fig. 4.7: Principle of measurement distance with two modulated frequences

4.4 Distance-Varying Illumination and Imaging Techniques for Depth Mapping

This method is described in PrimeSense's patent [21] and it is one of the possible solutions for Kinect depth cameras. The principle of method is in illuminating scene with grid of points, whereby each beam of light pass through two perpendiculars to each other cylindrical lenses with different optical power. This way manipulated pattern is changing its shape as function of distance.

4.5 Structured Light

Structured light is most often referring to a grid with periodically repeating lines or grids.

If we can accurately define projected pattern, then its deformation is directly proportioned to its to distance of projection plane. This can be simplified to simple point grid. Then we can notice the similarity to active triangulation method.

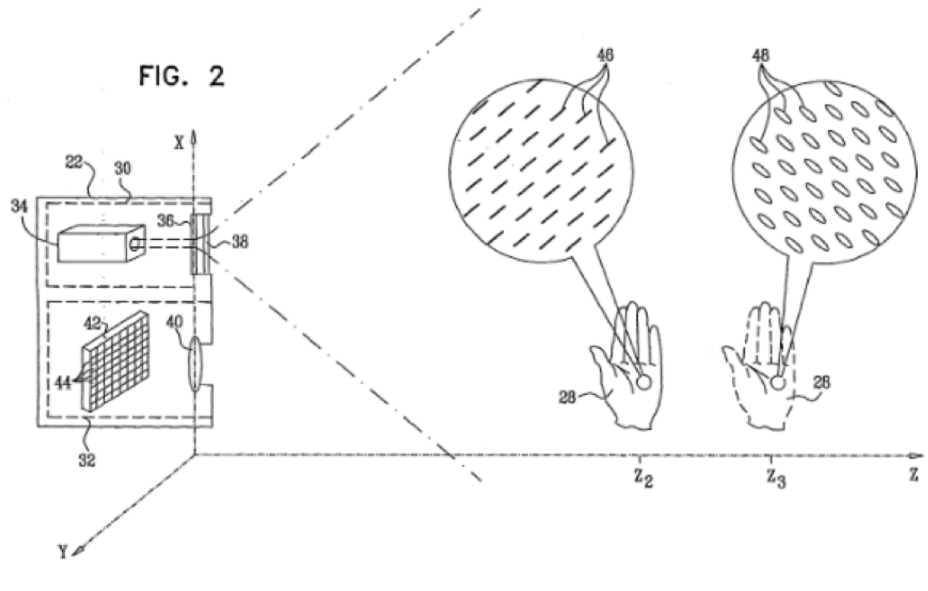

Fig. 4.8: Double focused light method. The shape of element is function of distance

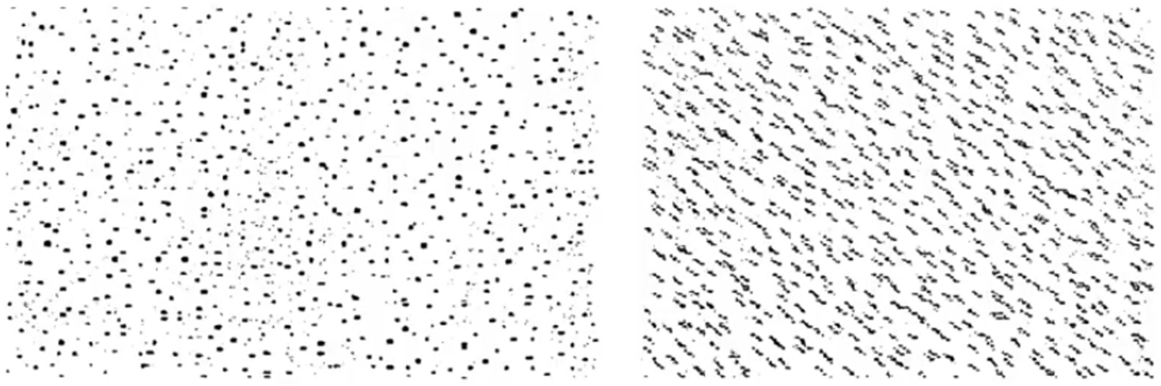

Fig. 4.9: Double focused light projected on two planes at different distance

With known triangulation base length, the distance between projector and camera and, the distance of measured point is function of projected point shift.

4.6 RGBD Cameras Overview

This chapter provides a brief overview over currently available depth cameras and summaries their operating principles and parameters. At the end of the chapter one device is chosen for future SLAM extension.

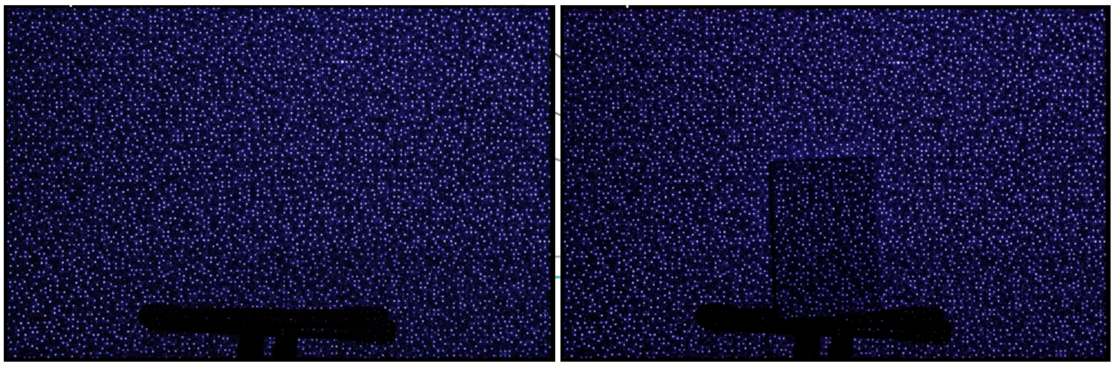
Fig. 4.10: Scene illuminated by Kinect in IR spectrum

4.6.1 Kinect v1 (Xbox 360)

The first version of Kinect device is a pioneer in the field of RGBD cameras. Microsoft and PrimeSense had developed it in 2010 as an accessory to the Xbox 360 game console, with price 150 USD, which was never seen before for this kind of device.

Principle of Kinect v1 is not officially known. But from the observations we are able to deduce that it is based on the principle of light focused through two cylindrical lenses as was described in PrimeSense's patent or on principle of active triangulation with grid of points projection.

Because of complexity of the first solution, the professional public rather incline to the second solution. The disadvantage of the first version of Kinect are shadows that are due to the large base of triangulation between the source of the pattern projection and infrared camera.

Price: 150 USD
Color resolution: 640x480px 30fps
Depth resolution: 320x240px 30fps
FoV: 43 x 57 deg
Principle: IR
Range: 0.4 – 4.5m

4.6.2 Kinecv v2 (Xbox One)

The second version of Kinect device changes measurement technique. According to observations, its measurement method uses modulated ToF. Accuracy of depth measurement is very similar to first version. However, the new Kinect device brings better depth image resolution and better geometrical precision of scanned scene (better contrast and less called flying pixels between two different distanced planes.

Price: 100 USD
Color resolution: 1920x1080px 30fps
Depth resolution: 512x424px 30fps
FoV: 70 x 60 deg
Principle: ToF
Range: 0.5 – 4.5m

4.6.3 Swiss Ranger SR4000

It is an industrial ToF camera created by Heptagon company. The device operates on an 850nm wavelength. However, it has relatively small resolution, only QCIF (176x144px) and works only in depth domain. Color output is not available. At the same time, according to [22] sensor has much less accurate than Kinect v1 output.

Price: 10 000 USD
Color resolution: N/A
Depth resolution: 176x144px 30fps
FoV: 43.6x 34.6 deg
Principle: ToF
Range: 0.5 – 5/10m

4.6.4 Asus Xtion Pro Live

It is a derivation of previous PrimeSense's Carmine 1.09, which was designed practically in the same way as Kinect v1 did. Also, the parameters of this device are very close to Kinect v1. It works on the principle of active triangulation and as previously mentioned. It projects grid pattern of the obstacles and measure distance as function of grid deformation. Color resolution is available in 1280x1024 resolution and 640x480px of depth images.

Price: 200 USD
Color resolution: 1280x1024px 30fps
Depth resolution: 640x480px 30fps
FoV: 58x 36 deg
Principle: IR
Range: 0.8 – 3.5m

4.6.5 SoftKinetic DS525

Device operating on a similar principle as Kinect v2 does. It is designed for shorter distances. The manufacturer declares that the optimum use case for default settings is from 0.15 up to 1m.

Price: 130 USD
Color resolution: 1280x720px 30fps
Depth resolution: 320x240 30fps
FoV: 63x43 deg
Principle: IR
Range: 0.15 – 1m (short range)
0.7 - 4m (long range)

4.6.6 Orbbec Astra Pro

Model designed for sensing the depth map over longer distances (up to 8 meters). The operation principle is based on the same technology as the Kinect v1. Active triangulation over projected infrared point grid is used.

Price: 150 USD
Color resolution: 1280x720px 30fps
Depth resolution: 640x480 30fps
FoV: 60x49.5 deg
Principle: IR
Range: 0.4 – 8m

4.6.7 Orbbec Persee

This device has the same color and depth cameras configuration as Orbbec ASTRA does. But very important technological shift is hidden in embedded computer inside of this device. It is equipped with processor, graphic chip, 2GB of RAM memory and Ubuntu OS support. Furthermore, the device also provides peripherals as Ethernet, WiFi, USB 2.0, HDMI and SD card slot. It is perfect option for embedded system projects with computer vision. However, this device still was not available during this thesis creation.

Price: 240 USD
Color resolution: 1280x720px 30fps
Depth resolution: 640x480px 30fps
FoV: 60x49.5 deg
Principle: IR
Range: 0.4 – 8m

4.6.8 ZED Stereo Camera

This device is based on principle of passive triangulation. Using two cameras, each with 2k2 resolution, it locates key points in the images and based on positional differences of these points it estimates the distance, as passive triangulation does.

Price: 450 USD	
Color resolution:	2208x1242px 15fps
	1920x1080px 30fps
	1280x720px 60fps
	672x376px 100fps
Depth resolution:	640x480px 30fps
FoV: 60x49.5 deg	
Principle: IR	
Range: 0.4 – 8m	

4.7 RGBD Camera Choice

Based on the mentioned facts above, I have chosen as the best suitable device for future work the Kinect v2. In these days, it has very wide support on the Linux OS platform and drivers [23] compilation and installation is quite fast and easy. Also, precision and depth range of device is good compared to the other devices and the shadow areas between color and depth cameras are much smaller, than for Kinect v1 device.

4.8 Kinect v2 Geometrical Calibration

Geometrical camera calibration, generally is a process of establishing the relationship how does the three-dimensional point is projected onto the sensor plane, in other words how the sensing point can be displayed on the CCD chip. Expression of these parameters priors to any visual measurement, because it allows to determinate and correct the errors, which would otherwise have been entered the measurement.

To determinate this relation between scanned space and sensor plane we can divide problem into three separate domains. The first problem concerns the measurement bias of optical system of the sensors, we are talking about intrinsic parameters. The second problem is to determine the translation and rotation of the sensor in global coordinates. These parameters are called extrinsic. The third problem is determining the distortion of sensor plane.

4.8.1 Pinhole Camera Model

Pinhole Camera (lat. Camera Obscura) is the simplest model of an optical camera. The device is designed as a closed chamber which one hole on site with negligible diameter. Incoming rays pass through the hole and projected to the wall on the other side of the chamber. In this case the length of the chamber equal to the focal distance f of the whole system.

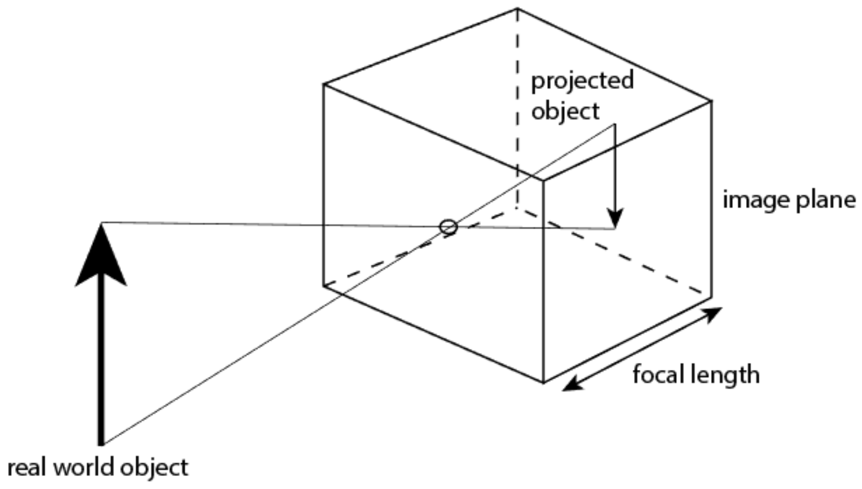

Fig. 4.11: Schematic of Camera Obscura

4.8.2 Simple Geometrical Projection $R^3 \rightarrow R^2$

Each point projected on the image can be connected with the optical center C of sensor. The optical center is an imaginary point in space at which all the scanned beams that have passed through the plane of the scanned image converge. At the point where the previously mentioned beam intersects the plane of the sensor the originally three-dimensional point is projected and creates two dimensional image.

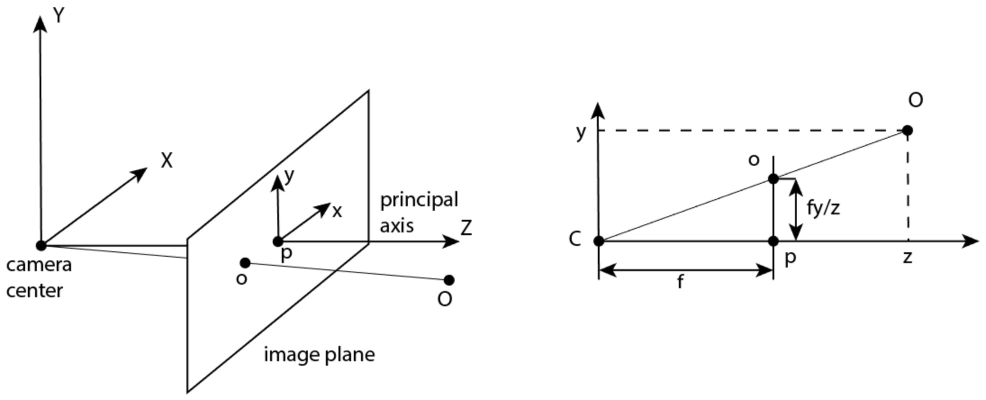

Fig. 4.12: Geometry of light ray projection on sensor plane

With knowledge of triangle similarity rule, the 3D spatial point $(x, y, z)^T$ will be projected to sensor as a point with coordinates $(\frac{fx}{z}, \frac{fy}{z}, f)^T$. Now this homogeneous transformation between the two coordinate systems can be expressed by a matrix equation.

$$\begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.16}$$

which can be generalized as follows

$$\boldsymbol{w} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \boldsymbol{P} \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{Z} \\ \boldsymbol{1} \end{bmatrix} \tag{4.17}$$

where X,Y,Z are coordinates of 3D point in global coordinate system, **P** is camera matrix, x,y are coordinates on projection plane and$w$ is scale.

Now we can expand problematic from model of Camera Obscura to real camera model.

$$\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}] \tag{4.18}$$

P is camera matrix, which is product of intrinsic camera parameters (intrinsic matrix) **K** and external camera parameters of rotation **R** and translation **t**.

If we look closer at the intrinsic matrix **K**, called calibration sensor matrix, we can find following intrinsic parameters

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{f/s_x} & \boldsymbol{k} & \boldsymbol{o_x} \\ \boldsymbol{0} & \boldsymbol{f/s_y} & \boldsymbol{o_y} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \tag{4.19}$$

which express focal length f, coordinates of image center o_x, o_y and pixel's dimensions s_x, s_y expressed in millimeters. Parameter k express the angle between x and y axis of projection plane. It is non-zero, when axes are non-perpendicular to each other.

$$k = \frac{f}{s_y} * tan(\alpha) \tag{4.20}$$

where α is complement to the $\frac{\pi}{2}$ of angle between x and y axis.

After the sensor matrix **P** is established, we are able to calculate optical center **C** of sensor with equation

$$\boldsymbol{C} = -\boldsymbol{P}_{\boldsymbol{3x3}}^{\boldsymbol{-1}}\boldsymbol{P}_{\boldsymbol{:,4}} \tag{4.21}$$

where P_{3x3}^{-1} represents matrix constructed of first three rows and columns of matrix **P** and $\mathbf{P}_{(:,4)}$ is the forth column of matrix **P**.

According to projection geometry the relation between **C** and **P** is expressed by following equation

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{1} \end{bmatrix} = \boldsymbol{0} \tag{4.22}$$

4.8.3 Sensor Distortion

In this case, we distinguish two kinds of distortions. The first one is called radial. It is caused by beam bend during the pass through the lens or set of lenses in sensor's optical system. The smaller the lens is the larger is the distortion. The radial distortion caused that image grid is deformed from the center distortion radially to the edges and conversely.

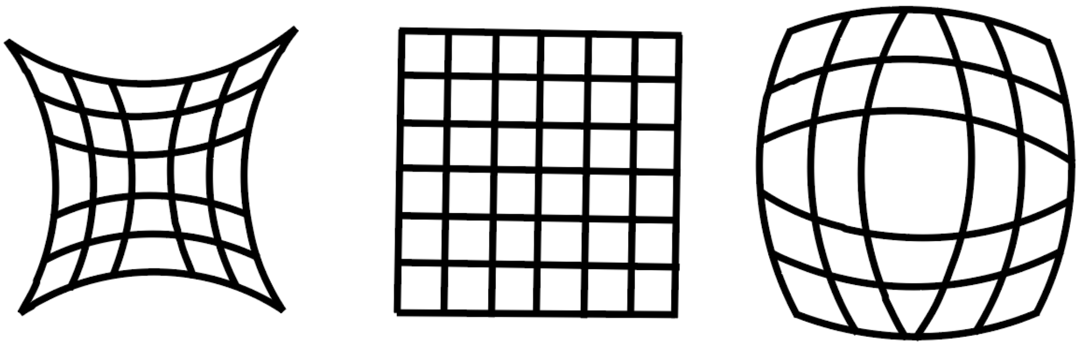

Fig. 4.13: Example of radial distortion a) negative radial distortion (Barrel), b) non-distorted grid, c) positive radial distortion (Pincushion)

The distortion is expressed as follows

$$\begin{aligned} x' &= x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) \\ y' &= y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) \end{aligned} \tag{4.23}$$

where x',y' are pixel's coordinates of distorted image, k_1, k_2 a k_3 are radial distortion coefficients and r is Euclidean distance of original pixel from the center of distortion.

The second kind of distortion is called tangential distortion or often also the perspective. It is caused by non-parallelly mounted the optical system and sensor chip.

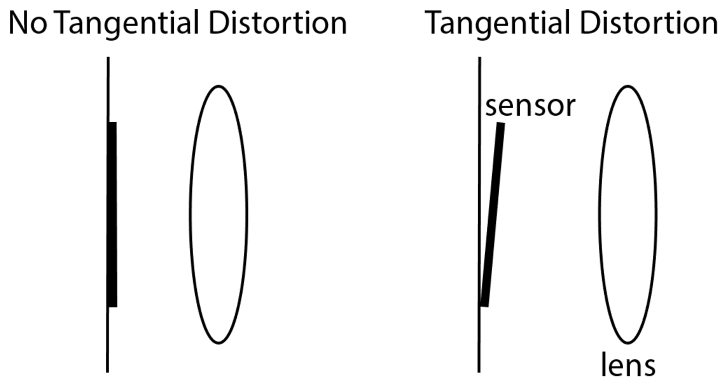

Fig. 4.14: The cause of tangential distortion

And can be mathematically expressed as follows

$$\begin{aligned} x' &= x + [2 * p_1 * x * y + p_2 * (r^2 + 2 * x^2)] \\ y' &= y + [p_1 * (r^2 + 2 * y^2) + 2 * p_2 * x * y] \end{aligned} \tag{4.24}$$

where x',y' are coordinates of distorted pixels with origin coordinates of x,y. p_1, p_2 are tangential distortion coefficients and r is Euclidean distance of original pixel from the center of distortion.

4.8.4 Calibration Process

The calibration of intrinsic parameters [24] of Kinect v2 was calculated on images of large-scale calibration checkerboard pattern of size approximately A3. It was taken about 20 frames by RGB and the infrared cameras, so that the calibration sample is always located in the different position and orientation relatively to the Kinectu while both channels took pictures at the same time.

To determine the intrinsic parameters and distortion coefficients the MATLAB tool "Single Camera Calibration App" was used [25]. It leads user through the whole process and allows him to configure and optimize the calibration procedure.

In the first step of calibration images are loaded into MATLAB environment and the calibration pattern is detected. Images where patter was not successfully detected are excluded. Following the calculation of the intrinsic parameters and coefficients of distortion. Then the software calculates the error of established model against each one individual calibration images and the resulting images compensated.

If any of the images has significantly larger error compared to the rest of the samples, it is recommended to remove this image from calibration set and perform new camera model calculation.

This process is iterated until the result is accurate enough. We can then export parameters and apply it to correcting output data from the Kinect v2.

Final calibration of Kinect v2 for RGB camera looks as follows.

$$K = \begin{bmatrix} \boldsymbol{f/s_x} & \boldsymbol{k} & \boldsymbol{o_x} \\ \mathbf{0} & \boldsymbol{f/s_y} & \boldsymbol{o_y} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{1065.1} & \mathbf{0.6527} & \mathbf{996.8} \\ \mathbf{0} & \mathbf{1066.9} & \mathbf{536.1} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$$f = 3.3mm$$

$$s_x = s_y = 3.1\mu m$$

$$[k_1, k_2] = [-0.0011, 0.115]$$

$$[p_1, p_2] = [-0.0024, 0, 0086]]$$

As is apparent from plots above, that tangential distortion is considerably smaller than the radial and has almost no effect on the total distortion. Therefore, it can be neglected.

4.8.5 Depth Camera Parameters

In case to resolve the parameters of depth camera of Kinect v2 few second-long static scene recording was taken. Later there was taken out one hundred frames from this record, on which there was calculated the average distance at which Kinect v2 has measured the average depth of the scene. The result is evident in the figure 4.19. Average pixel is always calculated as the mean value of the pixel at position x, y assuming that during the timeline t there were no pixel with a zero value, which means measurement fail. All such that pixels of the time series, that contain zero

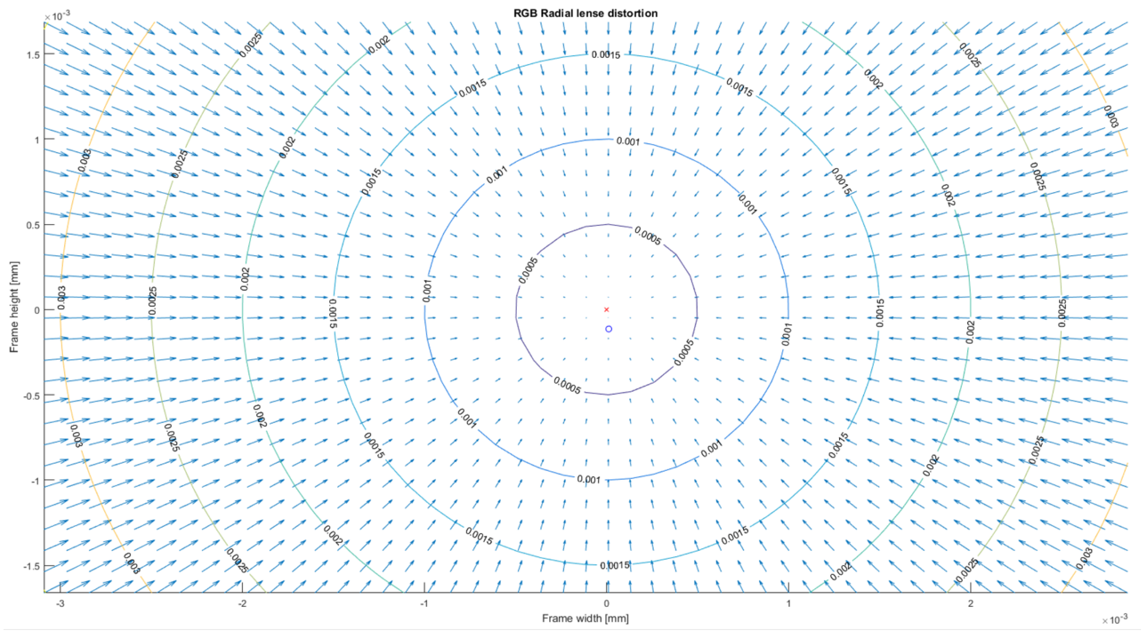

Fig. 4.15: Radial distortion of RGB cemara

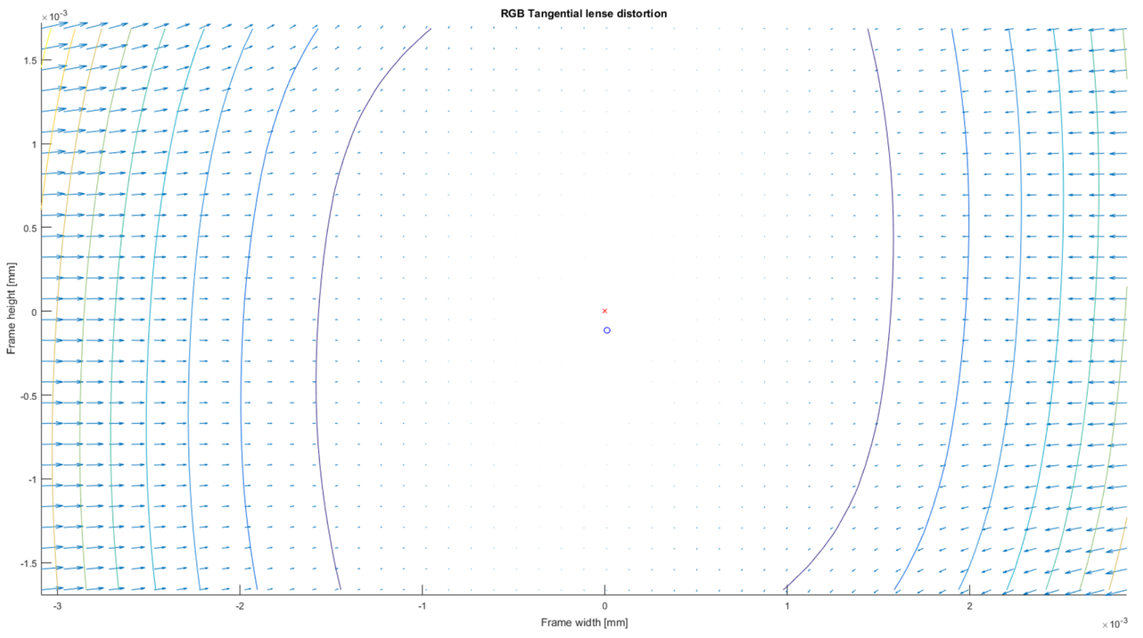

Fig. 4.16: Tangential distortion of RGB cemara

value has been excluded from the standard deviation calculation. Thus, the black areas represent measurement failure.

However, more important parameter is a reliability of measured data. For this

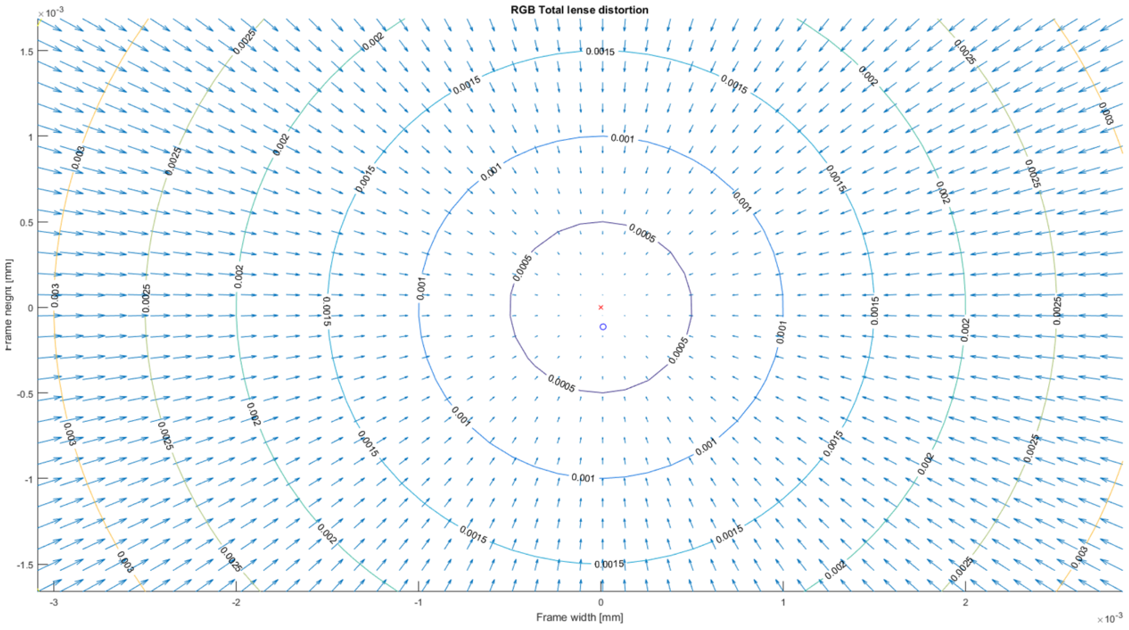

Fig. 4.17: Total RGB camera distortion

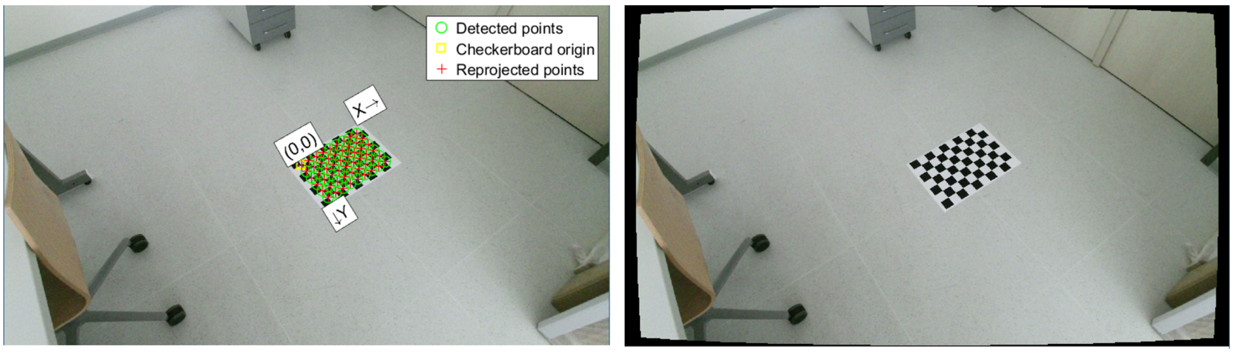

Fig. 4.18: Example of RGB image before and after distortion correction

purpose, for each pixel the standard deviation has been calculated. The result is shown in the figure 4.20. The standard deviation of the principal field of view is oscillating about two, up to five millimeters. It coincides with the results of [22]

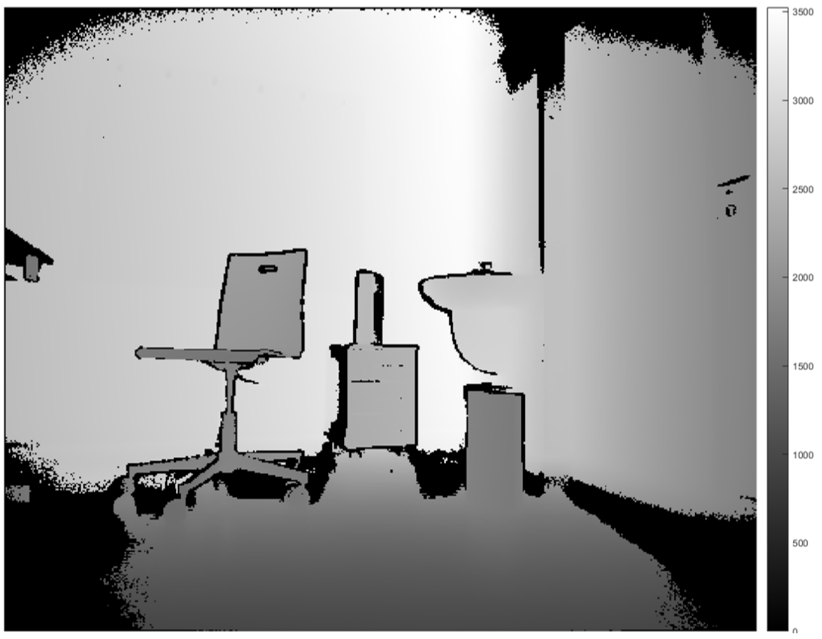

Fig. 4.19: Average value of depth of static scene represented by one hundred frames in row. One illumination unit represents one millimetre of scene depth

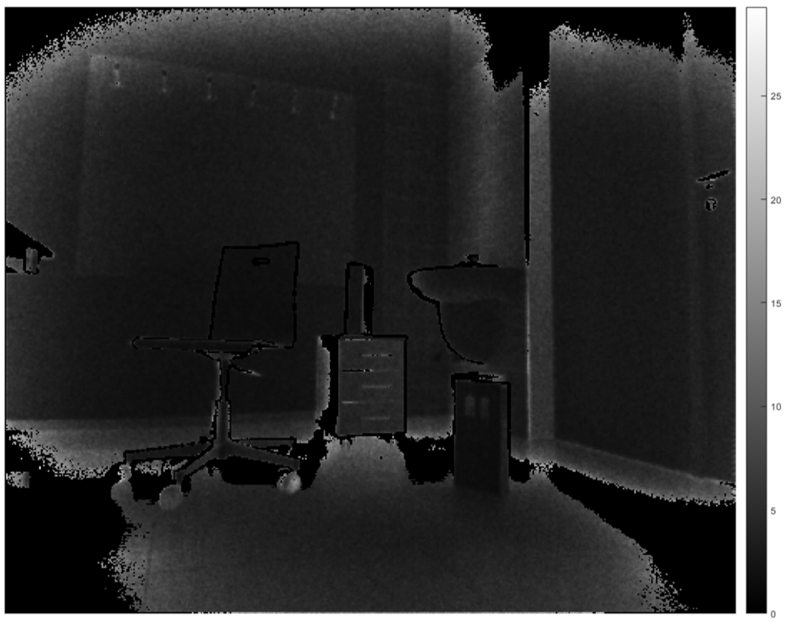

Fig. 4.20: Standard deviation of depth scene measurement in millimetrers, over one hundred frames

5 SLAM PROJECTS

This chapter introduces brief overview over existing projects in field of RGBD camera SLAM algorithms. At the very beginning the Kinect Fusion [4] is mentioned as a very first implementation of this problem with low cost RGBD camera and later the Kintinuous [5] and Elastic Fusion [6] projects are described. Both of them has been created by the same authors and both has been deeply studied for purposes of this thesis.

5.1 Kinect Fusion

Method described in [4] is the very first SLAM implementation with low cost RGBD camera usage. It was firstly introduced in 2011 as an example and developer demonstration of possibilities of new Microsoft's device. The source codes are not public, so we are not able to exactly determinate the way, how algorithm works, but according the released papers and according the open source clone of mentioned algorithm under the hat of PCL [26], we are able to get a brief idea how it works inside.

The Kinect Fusion code functionality is divided into four processes. All of them are implemented using quite new technology in 2011, the NVidia's CUDA. Because of this, the project's performance is scalable with improvement of this day's graphic cards.

In first step the raw depth data from Kinect depth camera are transformed into 3D space. The idea of this step is to take the depth image, which defines distances of all visible obstacles and using the Kinect's calibration matrix it performs projection of all pixels into 3D space. The result is a single vertex map. Also, the orientation of each vertex is calculated by comparing the self-vertex position with its nearest neighbors.

In the second step the camera position is estimated. For this purpose, the well-known ICP algorithm with 6DOF is used. In each iteration of ICP calculate the sum of all square distances between currently scanned scene and in-memory modelled environment and try to find the transformation H which minimize mentioned distance.

$$H = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.1}$$

In the third step the currently scanned scene is integrated into global environment model. For this purposes the SDFs functions are used. The idea is to interpolate existing voxels with spatial functions, that take negative values inside the defined 3D shape, zero-value for shape border and positive values for space outside the shape.

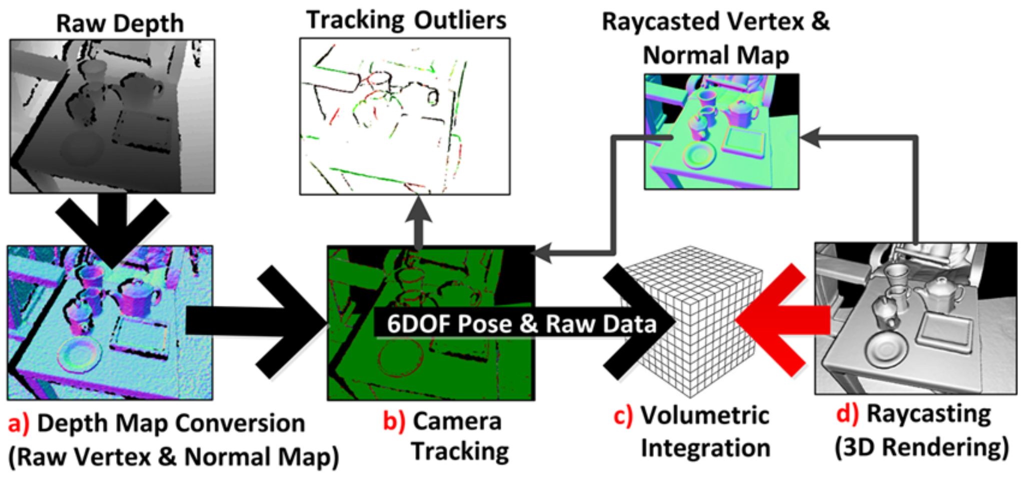
Raw Depth
Tracking Outliers
Raycasted Vertex & Normal Map
6DOF Pose & Raw Data
a) Depth Map Conversion (Raw Vertex & Normal Map)
b) Camera Tracking
c) Volumetric Integration
d) Raycasting (3D Rendering)

Fig. 5.1: Kinect Fusion's workflow [4]

This method has two advantages. First, the unexplored areas and noisy surfaces are interpolated with smooth surface and second, this space representation is well applied for graphics rendering techniques.

In the last step, called "Raycasting for Rendering and Tracking" the GPU calculates the beam for pixel in view frustum paced in current viewpoint. Every beam the GPU calculates if it crosses zero-value in its travel direction and if so, the zero-cross distance is estimated and relevant pixel in rendered image takes the equivalent intensity. This is how the SDFs defined environment is transformed into 2D image.

5.2 Kintinuous

Kintinuous project [27] is strongly based on techniques and ideas of previously mentioned Kinect Fusion. Also, it mentions three major disadvantages of Kinect Fusion. It is not able to scale the size of scanned environment, so the only limited space volume can be captured, the position estimation of Kinect Fusion is based only on depth camera information and there is no loop closing mechanism in this project.

Kintinuous is trying to introduce innovative approach in three mentioned fields and expand the idea, how RGBD SALM could work.

As first improvement, the shifting TSDF window is introduced. As so in Kinect Fusion the basic four steps had been performed over the entire volume, in Kintinuous these four operations are performed over the window, that is cutted out from global model. This window is static for infinite period of time, until the RGBD sensor moves behind the defined threshold and then new TSDF window in created around current position and the old one is released. This allows the algorithm to continuously work

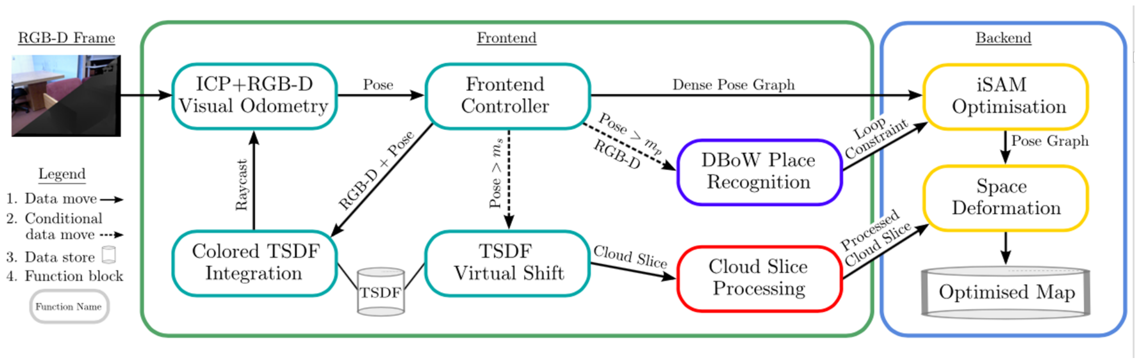

Fig. 5.2: Data-flow schematic in Kintinuous. Different colors runs of different threads [5]

with limited amount of data stored in GPU memory and also, algorithm is capable to move TSDF window behind the borders of initial location, so the environment model can be scaled to any size.

The second interesting improvement is the introducing the position estimation by using the RGB camera information. The process is very similar to ICP algorithm. The basic idea is to take two RGB images in time t and time $t-1$, calculate grayscale image and try to estimate the rotation R and translation T that minimize difference between image in time t and homogeneously transformed image in time $t-1$. The transformation with least difference between these two images is accepted as a representation of RGBD camera movement in last time period.

After both, the RGB and the depth positions are estimated, the linear combination of results of both method is done and the final transformation is propagated into next steps of algorithm.

The Kintinuous has very common in these days architecture. It is divided into two main block called frontend and backend. In frontend part, there are performed all up to now mentioned algorithms. The backend takes care about next introduced innovation, the Loop Closing mechanism.

As RGBD camera passes the space. In the backend memory, there is constructed graph that represents RGBD past positions and surrounding key-points (the landmarks). For purpose of key-points searching in RGB image, the SURF [28] method has been used. For every frame the SURF points are compared against the most probable candidates in global model and if correspondence is large enough, the RANSAC transformation estimation is done to validate loop closing. If this step pass, the ICP is used to calculate final deformation estimation. Because of the backend graph is non-rigid, the deformation of one point is distributed through entire graph and the global model is reorganized. .

5.3 Elastic Fusion

„Real-time dense visual SLAM system capable of capturing comprehensive dense globally consistent surfel-based maps of room scale environments explored using an RGB-D camera.“ [29]

This project has been created by the same authors as Kintinuous does, but the way, how the SLAM problematic is solved is different in many ways. The most interesting change is, that Elastic Fusion leaves the concept of Graph Based SLAM, which has been the state-of-art approach for nearly two decades.

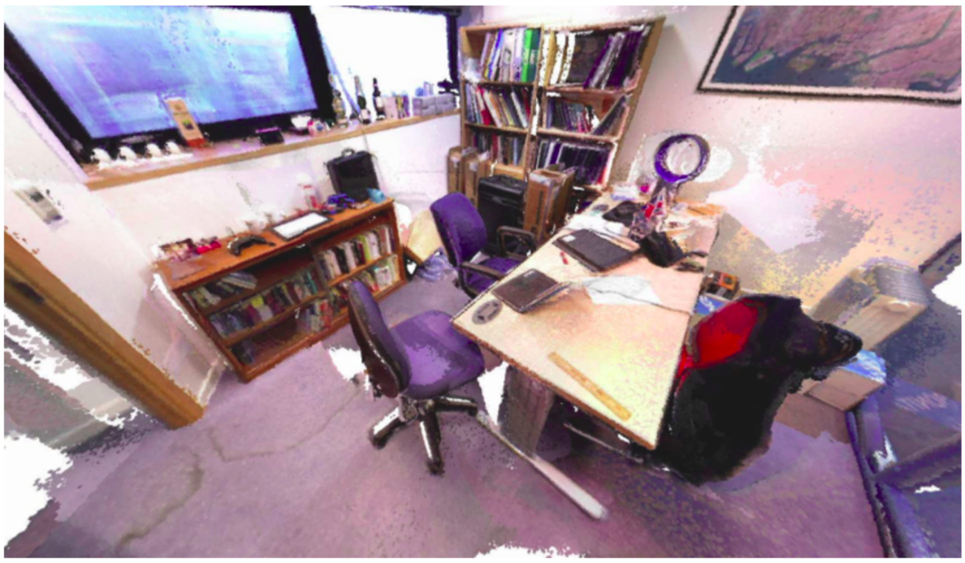

Fig. 5.3: Example of "Dense model" created by Elastic Fusion. It is composed of 4,5mil surfels

This project has been chosen as base of this thesis, because it introduces a new and interesting ways of solving SLAM problems and it is in active development process, so the future improvements can be expected. Another purpose is, that this program gives very precise output models in internet demonstration videos so good quality models could be expected during my work. Last, but not least important is fact, that in this project many modern programming techniques are used such as CUDA or OpenGL shader language, so I can learn many new technologies.

The Elastic Fusion’s model is defined by unordered set of surfels, where each surfel has defined its position $\mathbf{p} \in R^3$, normal $\mathbf{n} \in R^3$, color $\mathbf{c} \in R^3$, weight $w \in R$ and radius $r \in R$. Additionally, each surfel has its internal state (active//inactive). The surfel is active in the moment of creation and keeps active until the RGBD sensor is able to see it. If surfel has not been seen for defined period of time it comes inactive. Over such a define environment model the following operations are performed during SLAM runtime. Position estimation is done in a very same way as Kintinuous project does it. There are separated processes to estimate position from RGB and depth domains and then they are linearly combined together. In depth domain, the well-known ICP is used and in color domain there is iteratively minimized square difference of two in row coming RGB frames.

Interesting improvement is in the fact that local position estimation is performed only against active surfels, these are the ones that had been tracked during short period of time. In this way, the local operation window is created, so the frame-to-model process of local position estimation do not have to handle large amount of data.

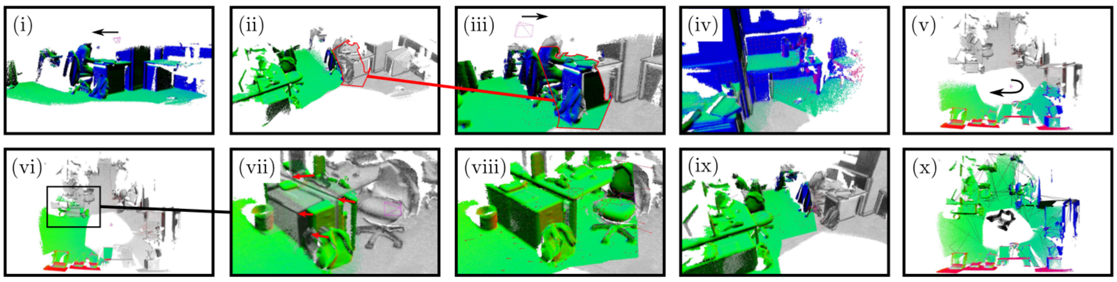

Fig. 5.4: [6] (i) initial view, (ii) leaving initial view, surfels become inactive, (iii) returning to inactive surfels, become active again, (iv) camera continues local loops closing, (v) exploring new areas, (vi) loop close detection, (vii) loop close detail, (viii) global loop closing activates surrounding inactive surfels, (ix) local loop closing continuous, (x) entire overview

In field of global graph deformation, the main innovation comes. There is no such a backend position graph that is processed during entire runtime. However, for each incoming frame the entire new graph is constructed. It is created out of several picked surfels from global model and each of these key-surfels is connected to its four neighbors. The key-surfels are picked randomly from whole set of existing ones, but the chosen is distributed uniformly through timeline of surfels creation timestamps, so the whole model is covered. This graph is used in next step to perform local and global loop closing.

During loop closing at first the global loop closing (mentioned in next paragraph) is performed. If no global loop closing has been detected, the local loop closing performed. It is done only above the set of currently active surfels and it helps to keep currently visible part of model consistent. It is done in way that transformation between active surfels and visible surfels is done. If transformation overcome defined threshold, model is deformed.

Global loop close matching is performed as follows. There are randomly selected views over entire model and current view is compared to selected ones. To improve performance, the views are down sampled to 80x60. If loop is detected, the deformation is done and it distribute through whole non-rigid model.

6 DATA FUSION

In this chapter reader is introduced into problematics of data fusion and the way, how this thesis handle data merging and what tools are used for this purpose.

6.1 Input Data Characteristics

At the beginning of this thesis there is a project which takes data from RGBD camera and performance complex SLAM algorithms to estimates camera's position and expands 3D environment model construction. Visual odometry is on its own not very accurate because of pixel resolution, cameras output picture deformation, ICP algorithm uncertainty and so on.

This thesis is aiming on feeding existing solution with more precise data to make results of running real-time SLAM more precise.

The amount of position information hidden in single image is called information gain.

Every camera's image keeps some information about current position. When we are comparing two different images of the same scene the different information makes possible to estimate position change. But for example, if first image is taken to calculate position, it gives us limited amount of information gain, but if second image is taken in the same position with the same orientation, it gives us no new usable knowledge, so the information gain of second image is equal to zero.

There are two ways how to improve SLAM runtime position information gain. The first one is to use more precise sensor that gives us better information with higher information gain. Also, more sensors can be used and the position measurement can be oversampled and the result can come out of averaging all sensors measurements together to reduce noise. But this is not the way that has been chosen in this thesis.

The second possibility is to estimate position by different measurement method, that has different characteristic of measurement noise and measurement method uncertainty. This gives us two independent streams of position information where each one has different measurement error and by combining this information gains together the characteristic error for each method can be suppressed.

As an example of different measurement principles, we could mention visual odometry, dead reckoning (described bellows), GPS position estimation, LIDAR scanning and so on. Every mentioned method uses different physical principle and gives different error characteristic.

In this thesis, visual odometry is combined with vehicle odometry.

Visual odometry is quite inaccurate in short time scale. Because of image noise the estimated position is oscillating around ground true position. But in long time scale it is able to suppress position estimation drift because of visual feedback to environment.

On the other hand, the dead reckoning method is very precise in short time scale (method shows very low noise amplitude), but even very low noise integrated during long period of time with missing correction feedback causes dramatic position estimation drift.

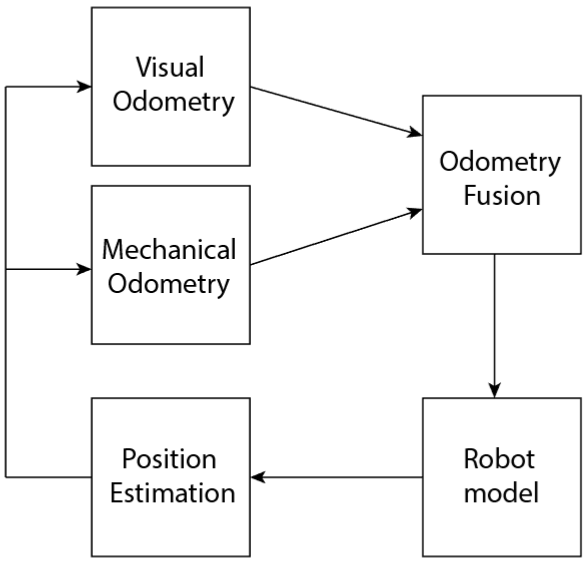

Fig. 6.1: Principle of different position data merging

6.2 Kalman Filter

Kalman filter [30], [31], sometimes called quadratic estimator, is a method of optimal dynamic data filtration, where noise has Gaussian distribution.

The entire filtration process is based on modelling the tracked system and estimating all its hidden states so the method is able to predict future changes and correct the predictions by available measurement.

Let's have a following system

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{6.1}$$

$$z_k = Hx_{k-1} + v_k \tag{6.2}$$

where A is state matrix or system's Jacobian, B is input matrix, H is model of internal system's states measurement (the measurement matrix). x_k is vector expressing system internal states z_k is measurement output. Constants w and v express the noise of system and measurement.

The system defined by equations above can be filtered by Kalman filter.

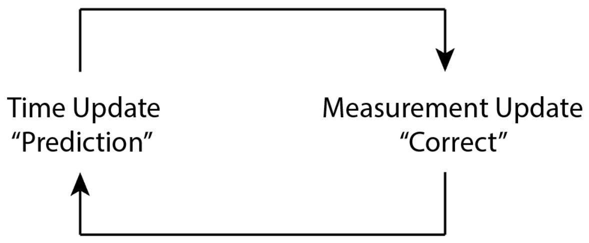

Fig. 6.2: Kalman filter iteration

As it is shown on image, each iteration of data filtrations is composed of two steps. In first one the future system state is estimated based on previous system states knowledge.

$$x_k^- = Ax_{k-1} + Bu_{k-1} \tag{6.3}$$

$$P_k^- = AP_kA^T + Q \tag{6.4}$$

The first equation expresses the mentioned prediction of system's internal states in step k based on knowledge of internal steps in previous step $k-1$. Q is system noise with Gaussian distribution.

The second equation expresses the prediction of system's covariant matrix, which express measurement uncertainty.

In the second step the prediction is corrected by measurement.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{6.5}$$

$$x_k = x_k^- + K_k(z_k - Hx_k^-) \tag{6.6}$$

$$P_k = (I - K_kH)P_k^- \tag{6.7}$$

K_k is called Kalman gain. It expresses how much should we trust to newly measured data and how much should we use them to estimate final position.

From equations above we can assume

$$\lim_{R_k \to 0} K_k = H^{-1} \tag{6.8}$$

which says that with decreasing measurement covariance R_k (decreasing measurement error) is rising the measurement trustfulness. And with decreasing model's covariant error P_k^- decreasing also the Kalman gain K_k.

$$\lim_{P_k^- \to 0} K_k = 0 \tag{6.9}$$

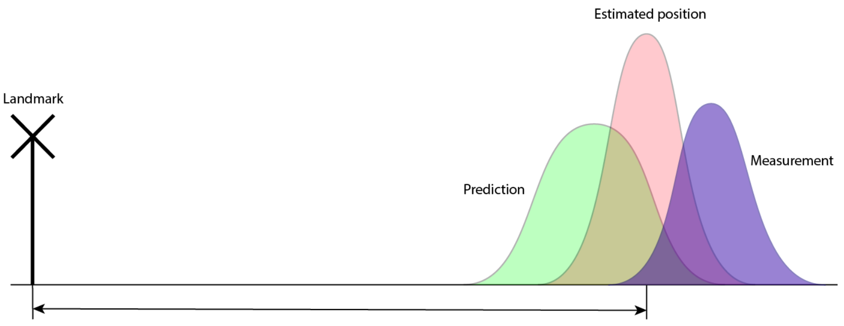

Fig. 6.3: Graphic interpretation of Kalman filtration (prediction, measurement, correction)

The largest problem of creation the properly working Kalman filter is to estimate initial coefficients. The measurement noise matrix R could be deduced from sensors parameters. But system noise Q estimation if very difficult and very often is estimated experimentally or some auto-covariant method, which estimates the level of noise based on model's statistics.

6.3 Extended Kalman Filter

The method described in previous chapter gives great results but only in case of modelling linear stochastic systems, which could be described by Gaussian statistic model. But if tracked system exhibits nonlinear behavior, the model has to be linearized for every step for every current state x_k in R^n dimensional space. This method is called Extended Kalman Filter.

In other words, for every iteration method has to calculate new model that exhibits linear behavior for near neighbor of its current internal states..

The system description is quite different in contrast with simple Kalman filter.

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \tag{6.10}$$

$$z_k = h(x_{k-1}, v_k) \tag{6.11}$$

6.4 Wheel Odometry (Dead Reckoning)

Dead reckoning is a method that measure the traveled distance by continuously integrating vehicle speed or wheels angular position change.

$$x_k = \sum_{i=1}^{k} \Delta x_i = \sum_{i=1}^{k} \Delta \dot{x}_i * \Delta t \tag{6.12}$$

Let us imagine a robot, that lives in one dimensional space and it moves in one direction with specified speed. If we assume, that at the beginning of measurement robot was in the space coordinate 0, than the position in time t can by expressed as the time velocity integral, where robot's speed can be estimated out of it's wheels rotation. The method is very precise, but during time there is negative effect of cumulating and integrating small error, which causes large drift during long period of time. The main sources of error are wheel rotation measurement error and wheel skid.

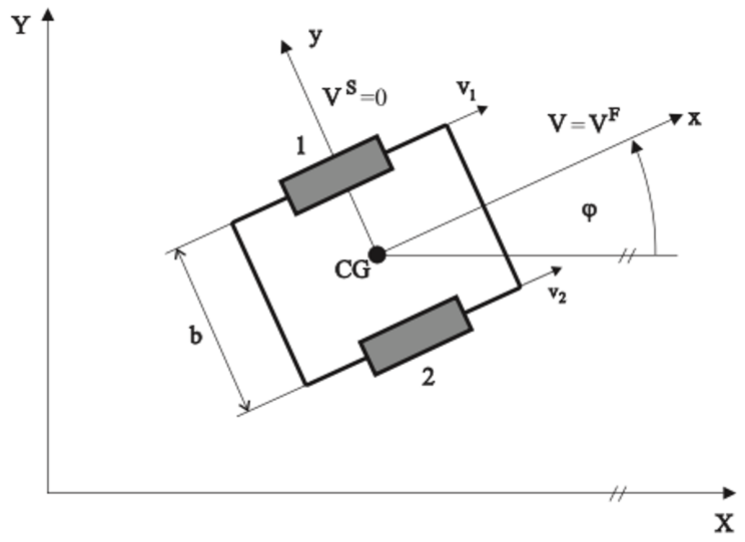

Fig. 6.4: Differential chassis model [32]

This method has been used for example in marine to estimate ships position on the see, where passed distance has been estimated out of number of propeller's rotations or continuous ship speed integration.

Now we are able to adapt this method on simple differential chassis model. Let's imagine a tank, which could be described by following model. The disadvantage of this model is that it is not able to model wheel's skid, which is always present with even very low amplitude.

By discretization all models equations we are able to create discrete model of Dead Reckoning method.

$$V_F = \frac{v_1 + v_2}{2} \tag{6.13}$$

$$V_S = 0 \tag{6.14}$$

$$\dot{\phi} = \frac{v_1 + v_2}{b} \tag{6.15}$$

$$\dot{x} = V_F cos\phi - V_S sin\phi \tag{6.16}$$

$$\dot{x} = V_F sin\phi + V_S cos\phi \tag{6.17}$$

6.5 Measurement vehicle

As part of my thesis the high precise measuring vehicle has been constructed. It has two wheels on the main axle, both with RI58-O / 5000AS.41RB quadrature encoders mounted. Furthermore, the vehicle is equipped with STM32F4 Discovery module, which is fitted into the extending shield. This computing unit functions as a collector of data from the encoders, which are then passed to the PC through the "serial line to TCP socket" network bridge connected to the same local network as SLAM PC does.

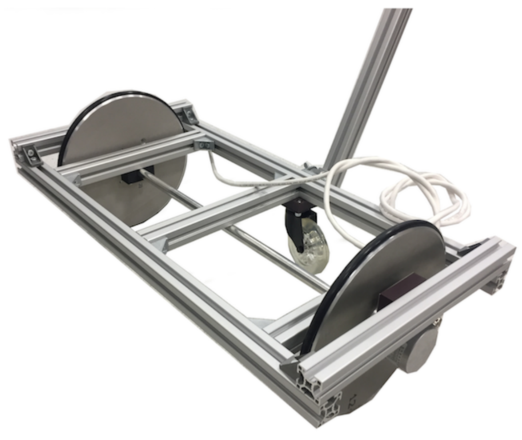

Fig. 6.5: Measurement vehicle construction

The PC is running the background process, which performs a model calculation of the vehicle odometry and through a buffer's queue it provides output of Dead Reckoning telemetry to the Visual SLAM thread, which is able to reduce cumulative position estimation error.

$$d' = f(x, y, \Phi, \Delta d_r, \Delta d_l) = \begin{bmatrix} x \\ y \\ \Phi \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} cos(\Phi + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \frac{\Delta s_r + \Delta s_l}{2} sin(\Phi + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix} \tag{6.18}$$

where x and y are robot's space coordinations, Φ is robot's orientation. Δd_r and Δd_l are right and left wheel distance changes and b is robot's chassis base.

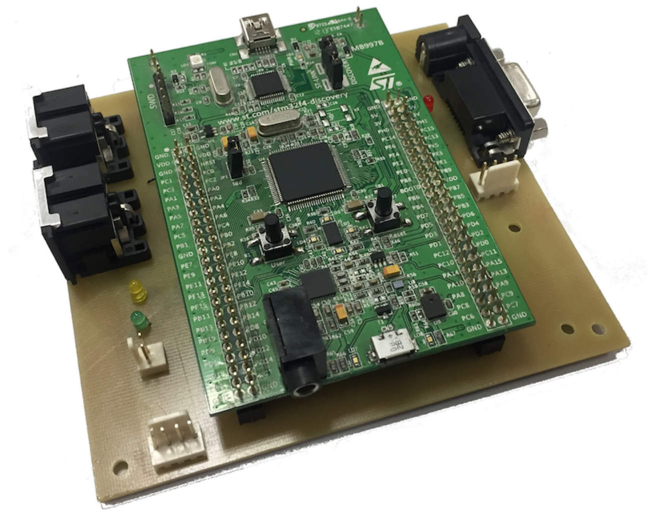

Fig. 6.6: Vehicle STM32F4 Discovery board with extension shield

The vehicle electronics, as previously mentioned, is based on STM32F4 Discovery board that is extended with expansion shield that provides power supply to entire vehicle electronics, the interface to interconnect microcontroller with quadrature encoders, three UARTs, one with level-shift to RS232 to communicate with another computer, one to interconnect with GPS module and one to communicate with Bluetooth module. Also, external interrupt pins are prepared to make possible external time synchronization.

The schematics and board of extension shield is placed in appendix.

6.6 Data Merge

The main idea of this thesis is to provide a differential chassis odometry into Visual SLAM process to improve accuracy of this method. On image 6.7 there is shown the way, how does the information about wheel's rotation and vehicle movement are transferred into SLAM process.

The wheel's rotation is sensed by two quadrature encoders. These two sensors are providing measured information into microprocessor and it integrates the number of incoming pulses and passes it via UART line into TCP/IP WiFi bridge with serial line input. The mentioned bride is connected into local Wi-Fi network and is accessible as a TCP server.

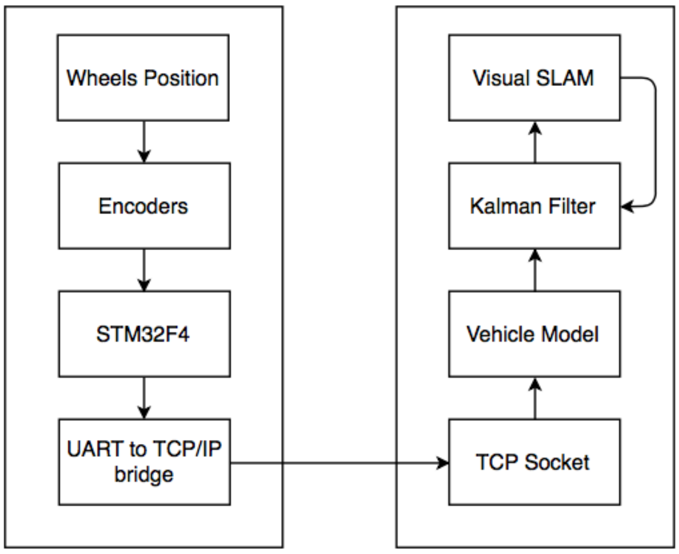

Fig. 6.7: Communication pipeline schematic

The SLAM hosting computer is connected to the same local network as vehicle's bridge does. The SLAM process contains TCP socket that connects to the bridge at the startup of the program and TCP socket begins to continuously receiving string messages, that contains information about number of encoder pulses captured by microprocessor.

This incoming information is parsed line-by-line and then it is passed into process that simulates runtime vehicle movement model. Based on differential equations that describe differential chassis (mentioned in previous chapter) the actual position in planar, two-dimensional world is estimated and this position information is provided into Kalman Filter module, where it is fused with output position from Visual SLAM.

The final result position information is forced into Elastic Fusion runtime.

6.6.1 Data Fusion Pipeline

To provide vehicle odometry data into Elastic Fusion's visual position estimation process, the one feature original Elastic Fusion API's feature has been used. For every frame the actual 6DoF position can be read out and also for each frame the same 6DoF position can be forced.

The vehicle odometry feeding pipeline works in the way showed on image 6.8. The Kinect v2 provides RGBD images with 60 fps framerate. The first frame is left without any modifications, to let Elastic Fusion performs initial position estimation. Before the second frame will be processed, the Visual odometry's position is taken as well as parallel vehicle position does. Both data packages are passed into Kalman filter, where the fusion is performed and after that the new estimated position is forced into Elastic Fusion's and vehicle's odometry pipelines.

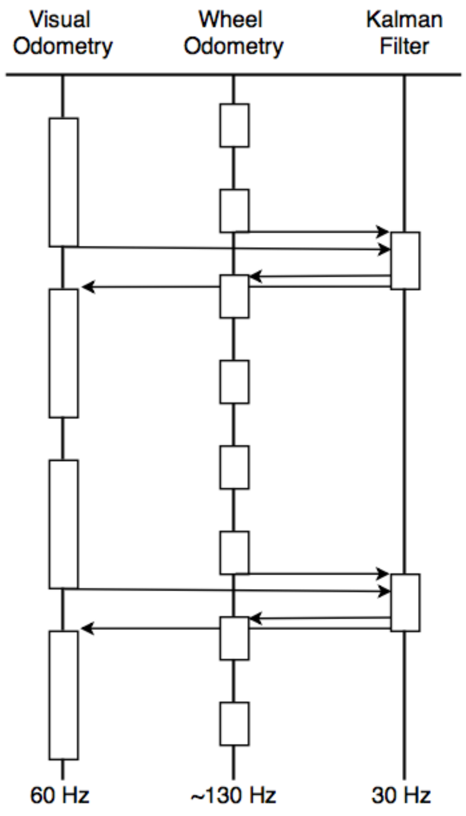

Fig. 6.8: Odometries fusion pipeline

This process is looping during entire SLAM runtime

Also, many parameters can be tuned. For example, the frequency of forcing position into Elastic Fusion's odometry can be done not only every second frame, but this number can be chosen arbitrarily. Even larger the period of position forcing will be chosen, the less the vehicle odometry will affect the position estimation.

7 RESULTS AND ACCURACY

In this chapter, there is described a way, how the results of this thesis has been tested. Firstly, there are mentioned all the software modifications, that has been finally used, next there is described whole measurement set, then reader is introduced into measurement experiences itself and in the last section the results are mentioned and they are compared to original Elastic Fusion performance.

7.1 Software Modifications

The entire project is composed out of two main parts. The first one is Elastic Fusion Core, which takes care about visual odometry estimation, map building, memory management and so on.

The second part is called GUI. There is running the main loop, which creates all instances, that handle graphic interface, Kinect's input data stream and use's interaction. This GUI parts include previously mentioned Core as a compiled static library.

Because of this software architecture all my modifications have been done only on GUI part. The Core part has not been changed.

The brief overview of all modification can be seen on 7.1

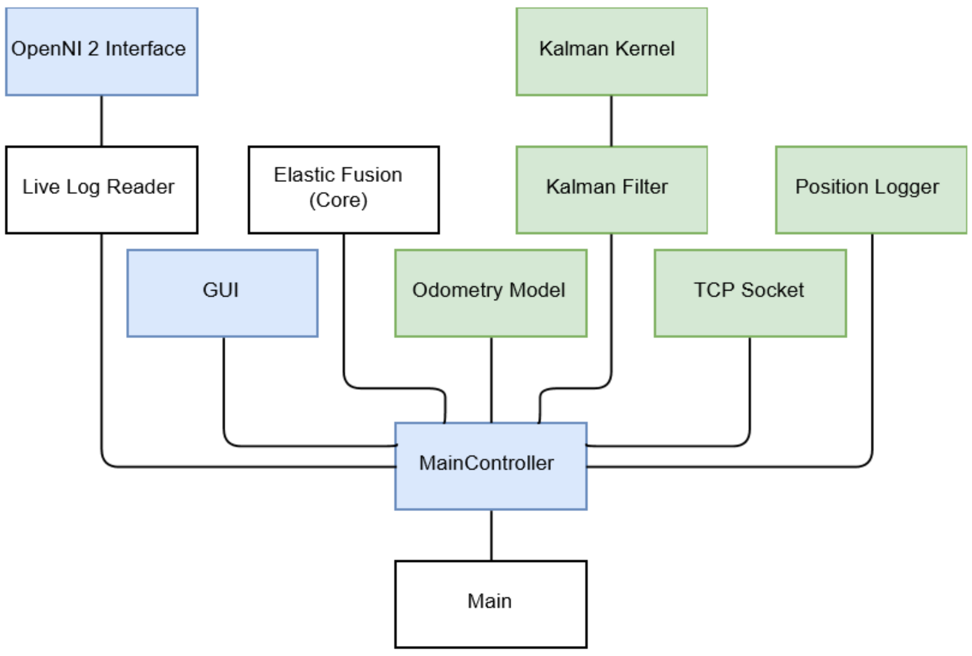

Fig. 7.1: UML Class diagram of modified Elastic Fusion project

This image represents only these parts of the project, that had been modified. Some includes of original project are not shown.

All the green classes had been created from the beginning for purpose of this thesis. The blue ones had been already present in original project and their functionality was modified. White ones weren't changed.

Main - contains main loop. Its only task is to call Main Controller and pass on runtime scope.

Main Controller - works as a bridge between all submodules. It performs data exchange between Kinect input, visual odometry core, odometry motion model, GUI rendering, etc.

OpenNi 2 Interface - handles input stream from Kinect v2 and transforms it into frame format (for both color domain and depth space). These frames are then passed into Elastic Fusion Core. Also, Kinect calibration is done in this block.

GUI - This class contains all graphic interface staffs and map rendering. For my purpose, some graphic elements has been added to be able to modify Elastic Fusion parameters during runtime.

Odometry Model - Instance of this class takes incoming encoders data from measurement vehicle and transform them into vehicle's position.

Kalman Filter - It is a wrapper, that encapsulates all filtrations into one instance. Every single degree of freedom (axis x and y and yaw) are calculated separately.

Kalman Kernel - Implementation of Kalman Filter. Every instance of this class is used for filtration of one degree of freedom

TCP Socker - This class represents TCP client implementation. During startup, it connects to the TCP server, that provides encoder's data. Incoming string stream is stripped into lines and every single line is passed into Odometry Model.

Position Logger - This class is used to logging robot's position. Vehicle odometry, visual odometry and fused position are logged separately. Every log also has time stamp.

Elastic Fusion Core - Takes RGB and depth frames and returns Kinect's translation and rotation.

The entire project is linked by using CMake and compiled by GCC compiler.

7.2 Measurement Set

During the measurement experiment the entire hardware set has been placed on measurement vehicle.

The vehicle itself has been equipped with STM32F4 board, battery pack, TCP bridge and Kinect v2. STM board and battery pack are placed in 3D printed black boxes, that are screwed to aluminum construction. The Kinect v2 is mounted also by 3D printed holders. The visualization of this parts is available in appendix.

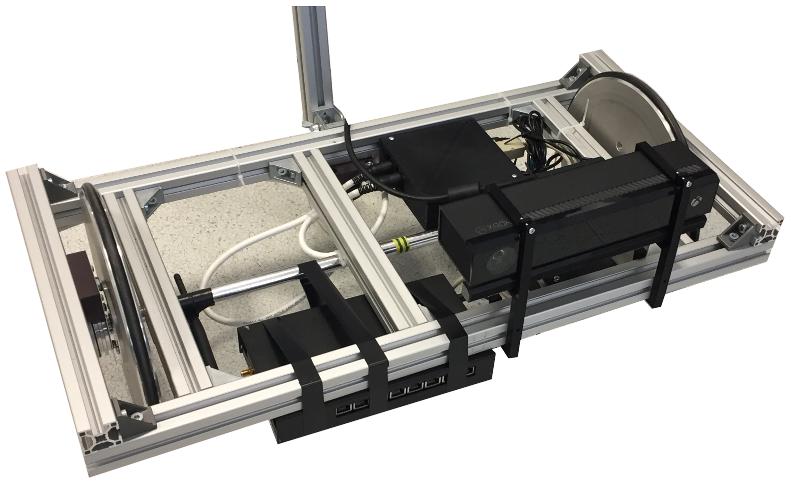

Fig. 7.2: Measurement vehicle equipped with Kinect v2, STM32F4 board, TCP bridge and battery pack

Because of Elastic Fusion runtime requires high compute performance (according authors 3,5 TFLOPS on GPU) and Kinect v2 has large data stream that is hard to pass through radio channel, so the entire PC had to be mounted on measurement vehicle. For this purpose, I have used Mini ITX PC with dimensions of 25x21x37 cm. This computer contains i5-6400 Intel CPU and 960 GTX NVidia graphics card. With this hardware, the whole experiment runs fluently with no freezing and no lags. The GPU memory usage was about 20 percent when model was build out of one million surfels.

7.3 Validation Experiments

The validation experiments have been performed in three different environments. Every environment have different character, so even different odometry methods gets different accuracy in 3D model costruction process.

The first experimental environment was the laboratory room. As the laboratory contains large number of different furniture, devices and other stuff, it creates a highly contrast environment in which even visual odometry gives very good results. Also loop closing works well, anyway it is better to minimalize active scene window timeout. In this case wheel odometry does not improve SLAM results in any useful way.

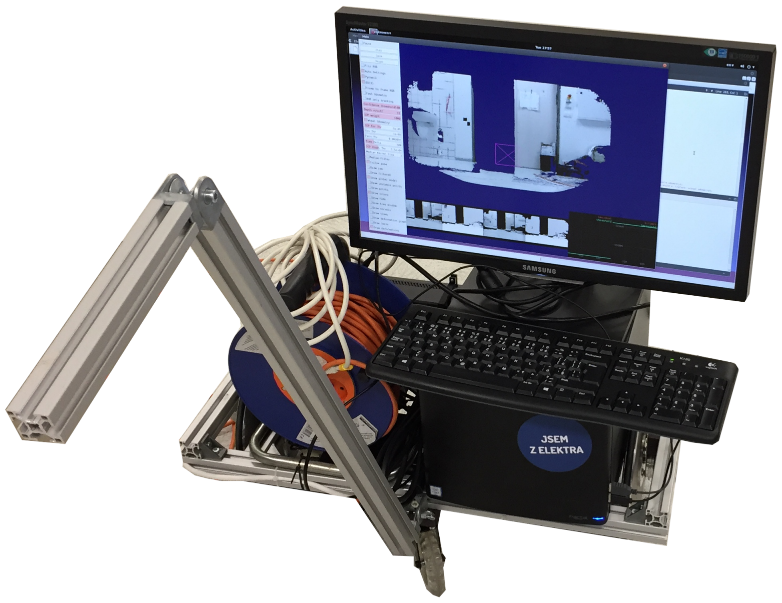

Fig. 7.3: Measurement vehicle with all hardware mounted

The purpose is, that visual odometry very well handle position estimation in highly rugged environment. Also, the accuracy of wheel odometry is strongly dependent on precision of calibration. During my thesis, there was no possibility to calibrate wheel odometry with higher precision, than 0.2% relative error in linear direction movement and more than 1% relative error in rotation movement.

Because of this uncertainty of wheel odometry the measurement error which was occurred in this experiment by fused odometry was larger than uncertainty of pure visual odometry.

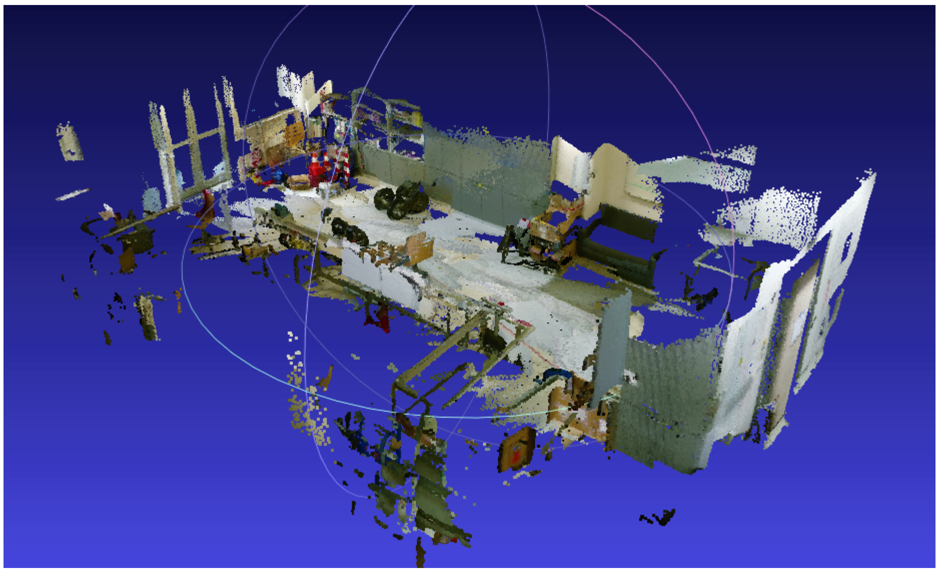

Fig. 7.4: Example of laboratory scan

The second experiment has been conceived as a pass between two laboratories through short section of corridor, when direction was changed two times. Experiment starts The hardest part for SLAM was to orientate during corridor pass through, because there are no contrast shapes. The pure visual SLAM usually gets lost during rotation. With fusion of both data sources inputs I was able to always pass this route without losing orientation.

The main benefit in this experiment shows, that operator, that controls robot with installed SLAM does not need to take care about ruggedness of environment, and if visual odometry gets lost, the position estimation by wheel odometry still works, until visual odometry starts to orientate.

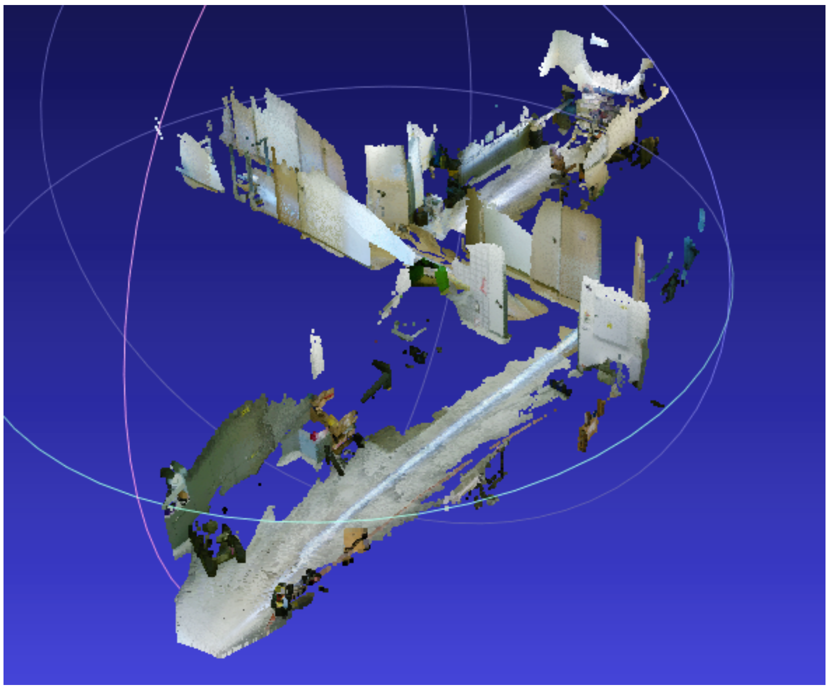

Fig. 7.5: Corridor between two laboratories scan

The third experiment was to straight pass through approximately forty meters long corridor. Also, it this experiment the most interesting results were achieved. The detail description is mentioned in following section.

7.4 Results

One of possible criterions to determinate accuracy of the SLAM method could be expression of relative error of straight corridor measurement.

Ground true value of the corridor length was measured from the middle of the doors on the one side of corridor to the middle of the doors on the second side parallely to the ground. This ground thrue value has been established by Lecia DISTO D8 laser scanner, that has measurement accuracy ± 1mm.

Lecia DISTO D8 laser scanner was used as reference measurement, because during my thesis there was no possibility to apply method or device with higher precision.

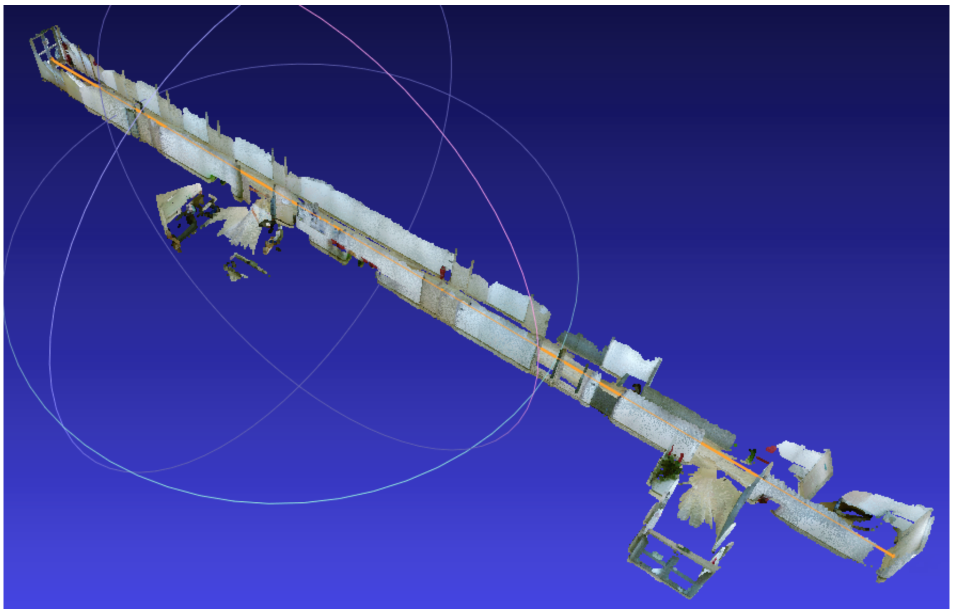
Fig. 7.6: The orange line shows measured distance

As it has been already mentioned, that each odometry method shows different type of measurement uncertainty. The wheel odometry is very accurate during small scale measurement, but has disadvantage of continuous integrated drift from ground true. On the other hand, the visual odometry has large uncertainty of current position estimation, but because of feedback it is able to compensate the mentioned wheel odometry drift.

In detail, the experiment of corridor length measurement has been performed in following way. SLAM was started looking on the first doors and slightly moved backward, to established 3D model of near surrounding. After this, it returned to original position and turn around by $180deg$. Usually in this moment the pure visual odometry get lost looking on non-contrast white wall, when combined and pure wheel odometry done well and was able to keep tracking surrounding. In last part of experiment the vehicle with SLAM was moving straight forward to the end of corridor with velocity about $0.5ms^{-1}$. At the end of corridor measurement has been terminated and results has been saved.

The following image 7.7 shows the different between visual odometry based 3D model construction and 3D model based on fused data. It is easy to see, that the pure visual odometry model shows large amount of geometrical distortion, meanwhile model constructed by fused data SLAM has much better geometrical expression of entire straight corridor.

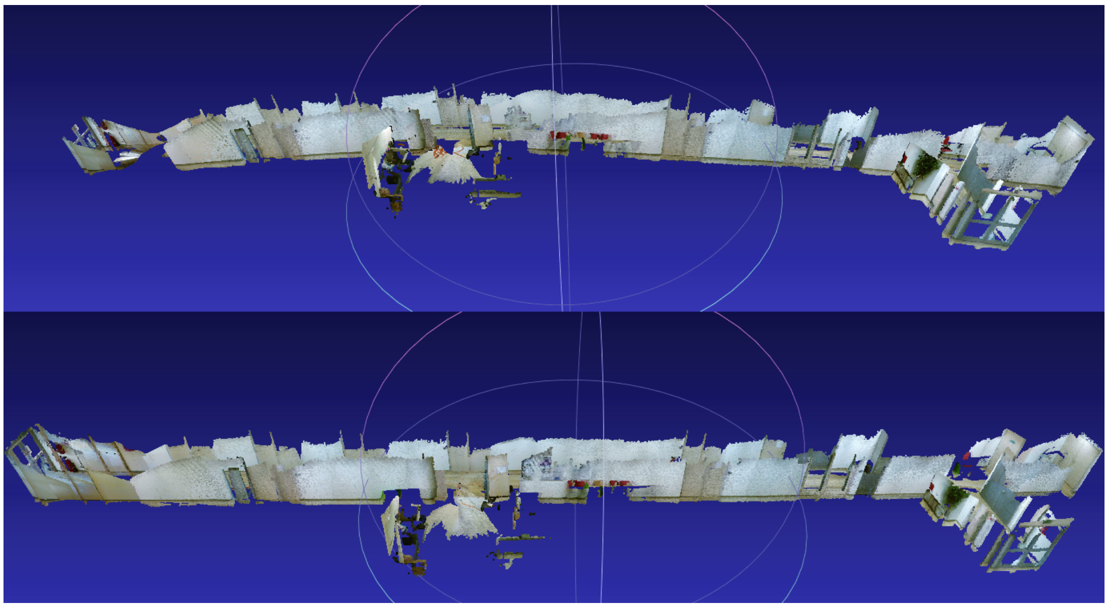

Fig. 7.7: 3D models build by Elastic Fusion (top - visual odometry only, bottom - fused odometry)

The relative error is expressed as follows

$$\delta = \frac{M - S}{S} * 100[\%] \tag{7.1}$$

where M is measured value and S is ground true value.

Method	Measured Length [m]	Relative Error [%]
Visual odometry	42.30	2.70
Wheel odometry	43.02	1.04
Fused data	43.29	0.41
Ground true	43.47	-

Tab. 7.1: Measurement methods error comparation

From this table we can assume, that results of Elastic Fusion with additional data source achieves few times better results (6 times lower relative error) than the original solution does. The wheel odometry helps to keep high accuracy on short time scale and visual SLAM kept the wheel's odometry drift at a low value, even on a quite long trace.

In state-of-art papers as a SALM benchmark, there is often used comparison of SLAM estimated trajectory against for example laser measured ground true, but during my thesis there was no possibility to perform this measurement.

Of course, there are much more criterions, than only relative error of measured distances. Let's mention for example the capability of keeping straight line. In other words, make SLAM to minimize geometrical distortion of straight planes (see 7.8).

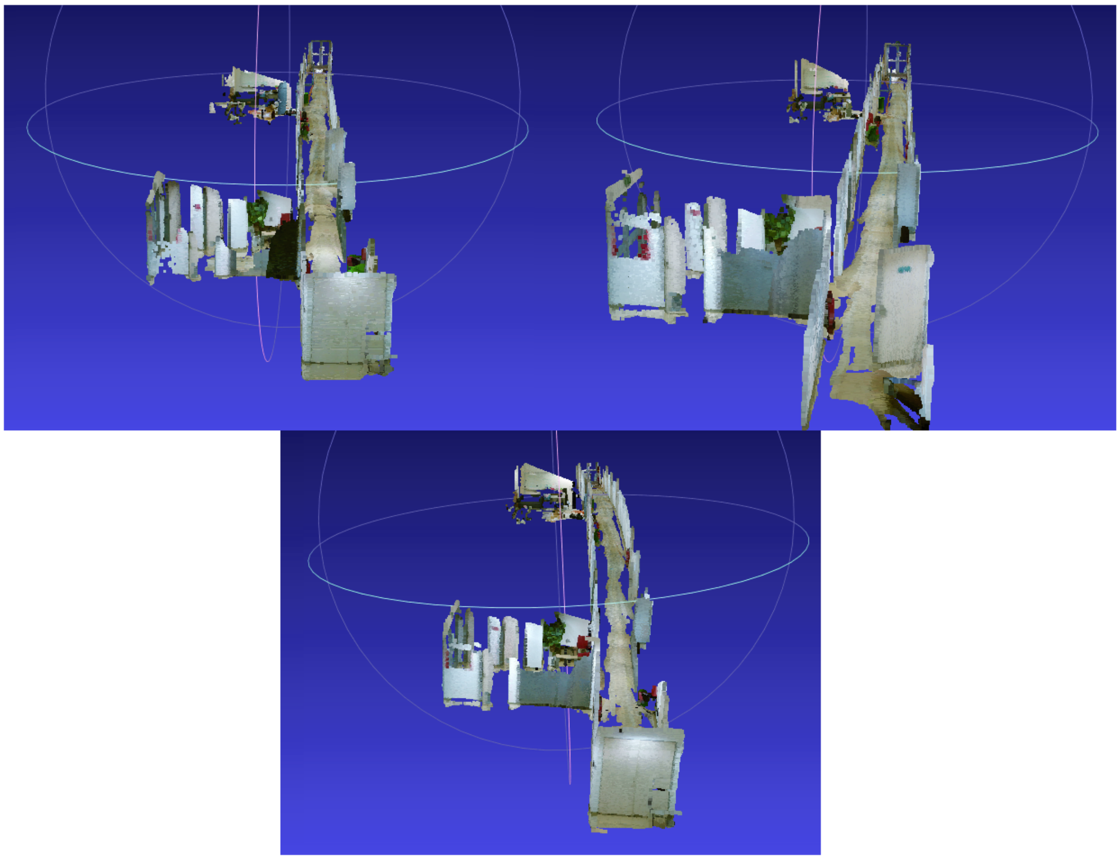

Fig. 7.8: 3D model's geometrical distortion, left top - wheel odometry only, right top - fused odometry, bottom - visual odometry only

8 CONCLUSION

This thesis dealt with visual SLAM and the methods to improve its accuracy. The entire work is based on the open source project "Elastic Fusion", which was then modified and the results were verified in several validation experiments.

During my work, I have forked the before mentioned open source SLAM project called Elastic Fusion and run it on Ubuntu 14.04. Next I had ported it from original Asus RGBD camera to newer Kinect v2 one. Then the electronics and firmware for measurement differential vehicle were created and entire wheel odometry was calibrated. In last step, I have modified the Elastic Fusion project to accept wheel odometry output data and fuse them with its own visual SLAM position estimation to achieve higher estimation accuracy.

To understand the motivation behind the fusion of visual and wheel odometry, it is necessary to understand that both methods have different measurement uncertainty. The visual odometry does not estimate current position very well, but because of the feedback can correct cumulative position estimation errors, on the other hand the wheel odometry has high accuracy for short distance measurements, but on large distances it integrates cumulative error. By combining these two methods, the disadvantages of both approaches could be eliminated.

The final results show the achieved improvements. The accuracy improvement has been tested in three different scenarios. The most interesting result was the one, when SLAM had to scan forty meters long corridor. The pure visual SLAM shows 2.7% relative error in corridor reconstruction measurement. The pure wheel odometry show 1% relative error and fused method creates 3D model with 0.4% error. It means over six times smaller measurement error compared to the original pure visual SLAM.

All the results mentioned above had been realized as a part of my master degree thesis. Also in future, there is possibility to expand this work during my Ph.D. studies.

At first there is necessary to encapsulate entire solution into for example Docker container, so that the project would be able to be deployed to any hardware that fulfil GPU performance requirements.

Then of course there is also task to deploy solution on Orpheus platform, that is developed by robotics team of FEEC of BUT. The robot is able to carry small-sized computer and Kinect v2 can be mounted for example on camera arm, so it would be able to rotate in all three axes.

By mounting Kinect to moving camera arm there would be possibility to also integrate inertial unit sensor, so the SLAM algorithm could lock all three rotation degrees of freedom and so position estimation could be much easier to solve and

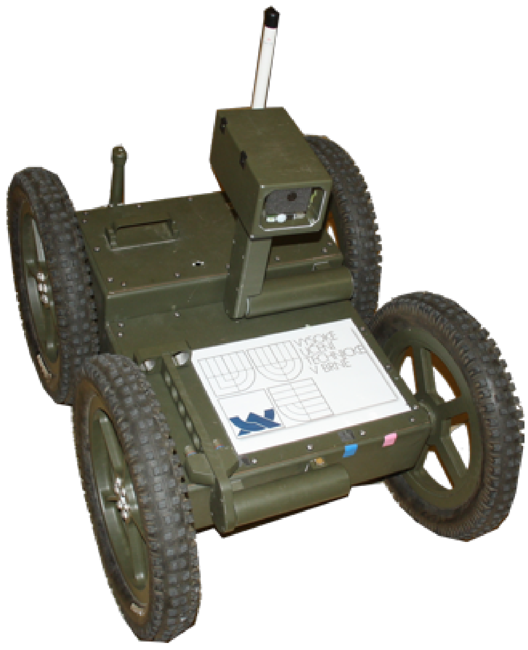
Fig. 8.1: Orpheus robot

estimated position could be more precise.

The next possible extension could be done in field of virtual reality. The entire 3D constructed model could be rendered into two framebuffers, so the illusion of 3D space could be created. With VR headset the operator, that controls the robot on its mission could get much better experience during exploration of unknown areas and would be able to comprehend much better, how is robot situated in its environment.

BIBLIOGRAPHY

[1] Waymo [online]. [cit. 2017-01-05]. Available from URL: <`https://waymo.com/`>.

[2] Tesla [online]. [cit. 2017-01-05]. Available from URL: <`https://www.tesla.com/`>.

[3] Průzkumný robotický systém Cassandra [online]. [cit. 2017-01-05]. Available from URL: <`https://www.ceitec.cz/02-2014-pruzkumny-roboticky-system-cassandra-automa/f1398`>.

[4] Richard A. Newcombe, Shahram Izadi *KinectFusion: Real-Time Dense Surface Mapping and Tracking* Imperial College London, Microsoft Research. Available from URL: <`https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/ismar2011.pdf`>.

[5] Thomas Whelan, Michael Kaess *Real-time large scale dense RGB-D SLAM with volumetric fusion.* Available from URL: <`http://thomaswhelan.ie/Whelan14ijrr.pdf`>.

[6] ElasticFusion: Dense SLAM Without A Pose Graph [online]. [cit. 2017-01-03]. Available from URL: <`http://thomaswhelan.ie/Whelan15rss.pdf`>.

[7] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, W. Burgard *An Evaluation of the RGB-D SLAM System.* Available from URL: <`http://www2.informatik.uni-freiburg.de/~endres/files/publications/endres12icra.pdf`>.

[8] ElasticFusion: Dense SLAM Without A Pose Graph [online]. [cit. 2017-01-03]. Available from URL: <`http://thomaswhelan.ie/Whelan15rss.pdf`>.

[9] Orpheus Robotic System [online]. [cit. 2017-01-05]. Available from URL: <`REFERENCEhttp://www.uamtold.feec.vutbr.cz/robotics`>.

[10] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, Wolfram Burgard *A Tutorial on Graph-Based SLAM* . University of Freiburg. Available from URL: <`http://www2.informatik.uni-freiburg.de/~stachnis/pdf/grisetti10titsmag.pdf`>.

[11] MAIER, Robert, Jurgen STURM a Daniel CREMERS. *Submap-based Bundle Adjustment for 3D Reconstruction from RGB-D Data* TU Munich, Germany. Available from URL: <`https://vision.in.tum.de/_media/spezial/bib/maier2014gcpr.pdf`>.

[12] Point Cloud Library [online]. [cit. 2017-01-05]. Available from URL: <`http://pointclouds.org`>.

[13] Wikipedia - Octree [online]. [cit. 2017-01-05]. Available from URL: <`https://en.wikipedia.org/wiki/Octree#/media/File:Octree2.svg`>.

[14] BURIAN, F. *TVORBA MULTISPEKTRÁLNÍCH MAP V MOBILNÍ ROBOTICE*. Brno, 2014. VUT Brno. Vedoucí práce Doc. Ing. LUDĚK ŽALUD, Ph.D. Available from URL: <`https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=93104`>

[15] P.J. Besl; H.D. McKay *A method for registration of 3-D shapes.* Available from URL: <`http://ieeexplore.ieee.org/document/121791/media`>

[16] BAZEILLE, S a D FILLIAT. *Combining Odometry and Visual Loop-Closure Detection for Consistent Topo-Metrical Mapping* ENSTA ParisTech. Available from URL: <`http://perso.ensta-paristech.fr/~filliat/papers/Bazeille_COGIS09.pdf`>.

[17] Kin Leong Ho, Paul Newman *Loop closure detection in SLAM by combining visual and spatial appearance* Oxford Robotics Research Group . Available from URL: <`http://www.robots.ox.ac.uk/~mobile/Papers/VisualLoopClosureSLAM.pdf`>.

[18] Martin A. Fischler, Robert C. Bolles *Random Sample Consensus: A Paradigm for Model Fitting with Apphcatlons to Image Analysis and Automated Cartography* SRI International. Available from URL: <`http://www.cs.columbia.edu/~belhumeur/courses/compPhoto/ransac.pdf`>.

[19] Wikipedia - RANSAC [online]. [cit. 2017-01-03]. Available from URL: <`https://en.wikipedia.org/wiki/Random_sample_consensusl`>.

[20] Rémi Paucher, Matthew Turk Location-based augmented reality on cell phones [online]. [cit. 2017-01-03]. Available from URL: <`https://ilab.cs.ucsb.edu/projects/remi/remi.html`>.

[21] Distance-Varying Illumination and Imaging Techniques for Depth Mapping [online]. [cit. 2017-01-03]. Available from URL: <`https://www.google.com/patents/US20100290698`>.

[22] Jan Smisek, Michal Jancosek and Tomas Pajdla *3D with Kinect.* Available from URL: <`http://cmp.felk.cvut.cz/ftp/articles/pajdla/Smisek-CDC4CV-2011.pdf`>.

[23] libfreenect2 - GitHub [online]. [cit. 2017-01-03]. Available from URL: <https://github.com/OpenKinect/libfreenect2>.

[24] SCHREER, KAUFF a SIKORA *3D video communication.* ISBN 13 978-0-470-02271-9 (HB), 2005, s 365

[25] Single Camera Calibration App [online]. [cit. 2017-01-03]. Available from URL: <https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html>.

[26] An open source implementation of KinectFusion [online]. [cit. 2017-01-03]. Available from URL: <http://play.pointclouds.org/news/2011/12/08/kinectfusion-open-source/>.

[27] Kintinuous - GitHub [online]. [cit. 2017-01-03]. Available from URL: <https://github.com/mp3guy/Kintinuous>.

[28] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool *SURF: Speeded Up Robust Features*. Available from URL: <SURF:SpeededUpRobustFeatures>.

[29] Elastic Fusion - GitHub [online]. [cit. 2017-01-03]. Available from URL: <https://github.com/mp3guy/ElasticFusion>.

[30] Ramsey Faragher *Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation* . Available from URL: <https://www.cl.cam.ac.uk/~rmf25/papers/Understanding%20the%20Basis%20of%20the%20Kalman%20Filter.pdf>

[31] Greg Welch, Gary Bishop *An Introduction to the Kalman Filter*. Department of Computer Science University of North Carolina at Chapel Hill. Available from URL: <http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf>.

[32] Šolc František, Tomáš Neužil, Jakub Hrabec, Jaroslav Šemberar *Kinematický model kolového, smykem řízeného robota.* Available from URL: <http://www.atpjournal.sk/buxus/docs/casopisy/atp_plus/plus_2008_1/plus01_05.pdf>.

LIST OF SYMBOLS, PHYSICAL CONSTANTS AND ABBREVIATIONS

2D	Two Dimensional Space
3D	Three Dimensional Space
API	Application Interface
deg	degree
EKF	Extended Kalman Filter
fps	frames per second
FoV	Filed of View
GPS	Global Positioning System
ICP	Iterative Closest Point
LIDAR	Light Imaging, Detection And Ranging
ORB	oFAST + rBRIEF
PCL	Point Cloud Library
px	Picture Element (pixel)
RGB	Red, Green, Blue
RGB-D	Red, Green, Blue, Depth
SDF	Signed Distance Function
SDK	Software Development Kit
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localisation And Mapping
SURF	Speeded Up Robust Features
ToF	Time of Flight
TSDF	Truncated Signed Distance Function
vx	Volume Element (voxel)

LIST OF APPENDICES

A STM32F4 EXTENSION SHIELD

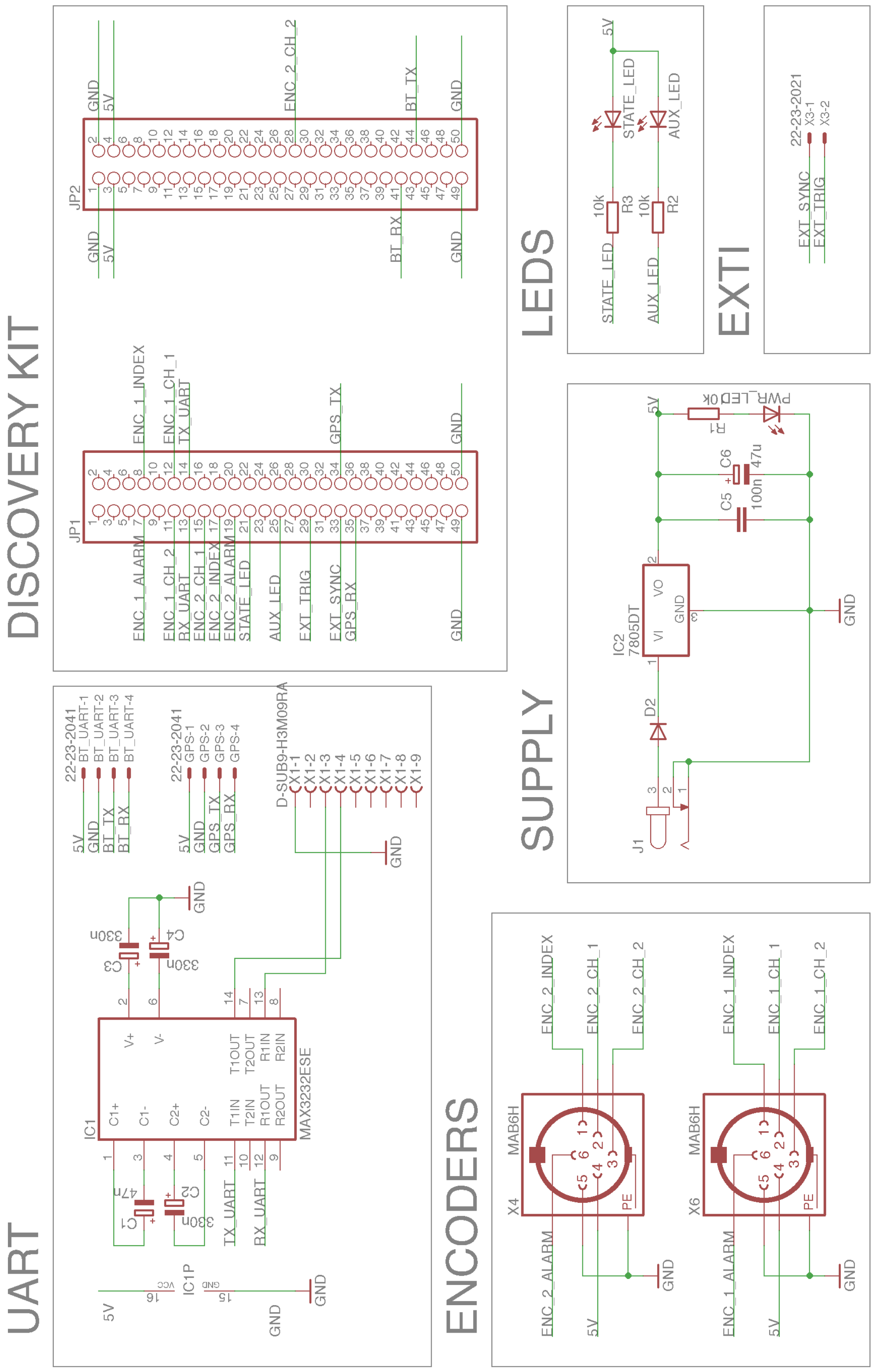

Fig. A.1: STM32F4 Extension Shield - Schematic

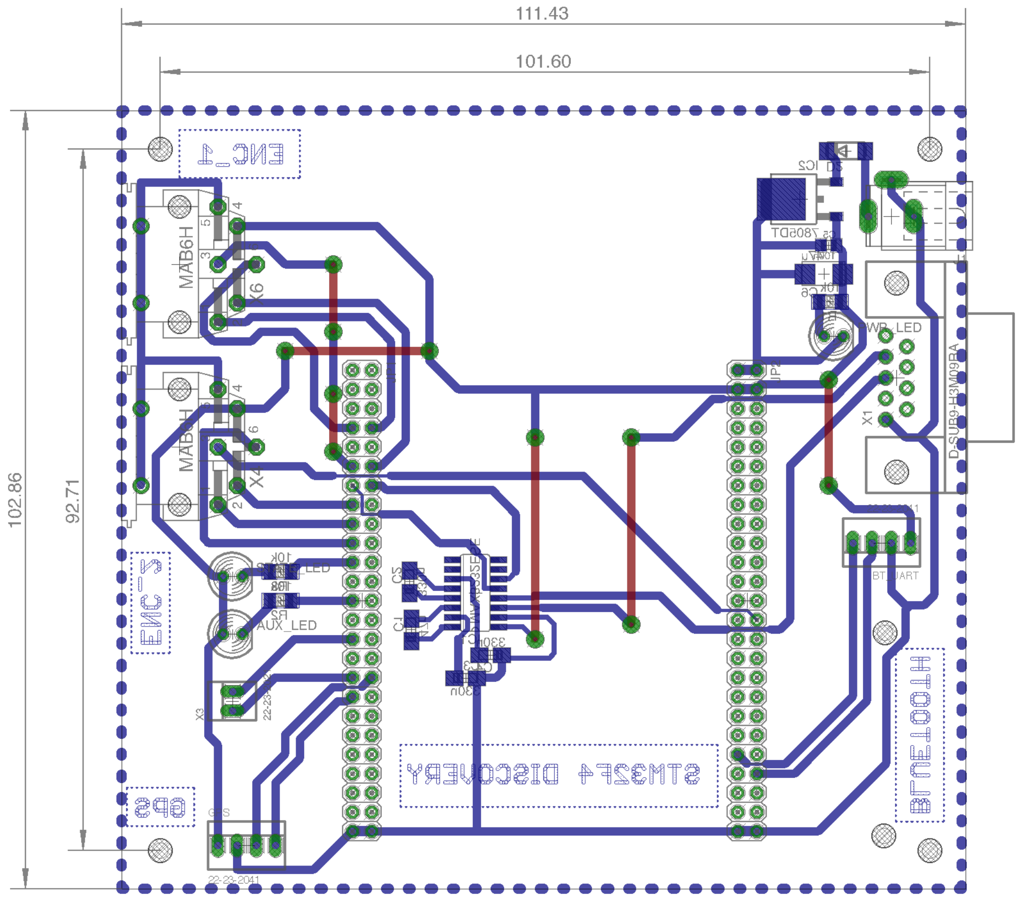

Fig. A.2: STM32F4 Extension Shield - PCB

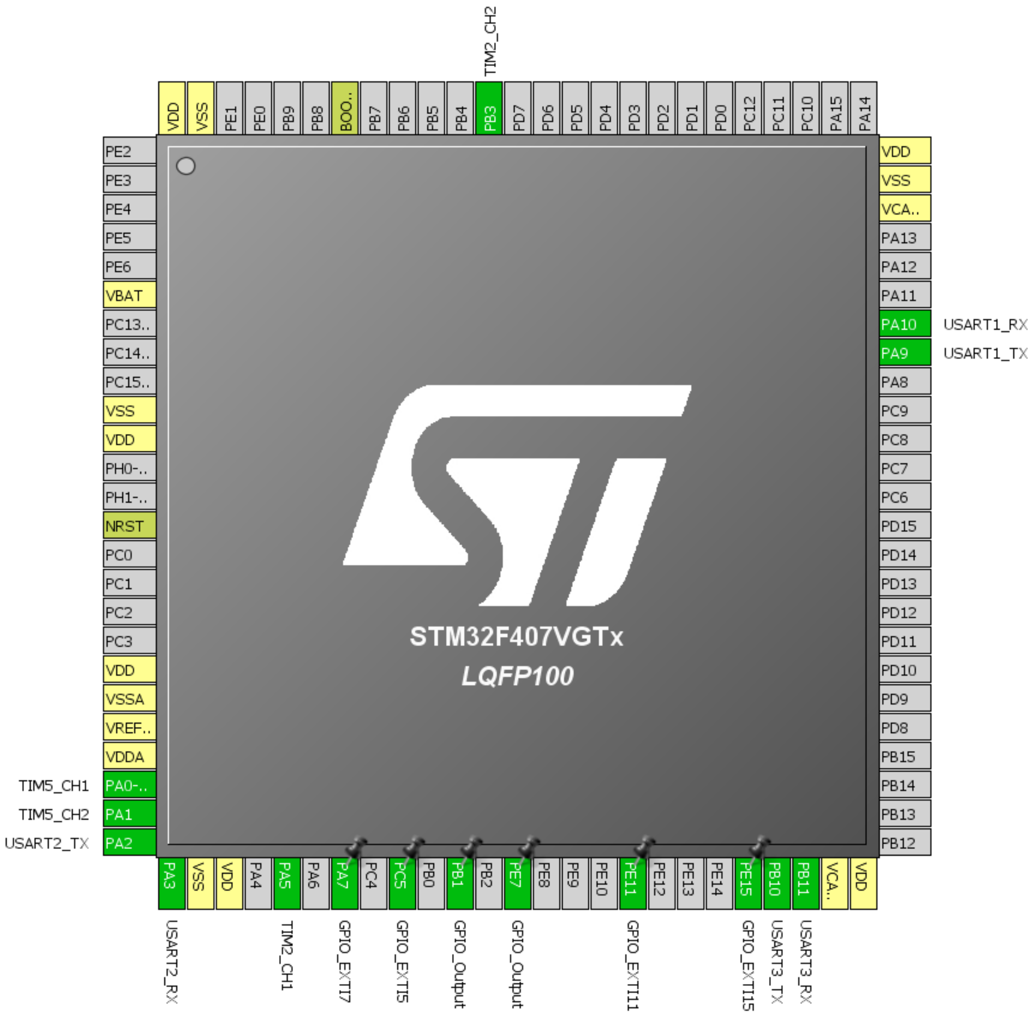

Fig. A.3: STM32F4 pin assignment

B 3D-PRINTED INSTALATION PARTS

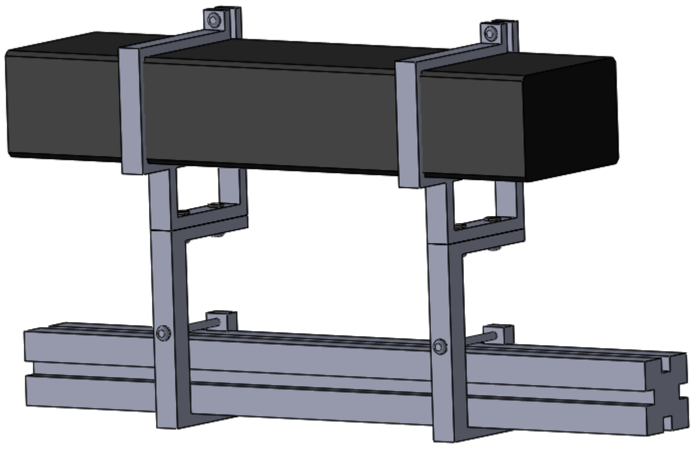

Fig. B.1: Kinect v2 holder visualization

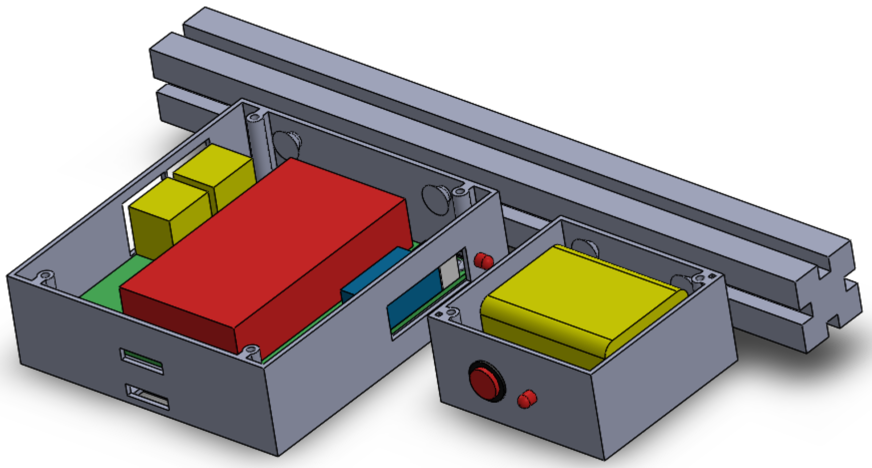

Fig. B.2: Measurement vehicle electronics visualization

C ENCLOSED MEDIA DEVICE CONTENT

Entire content of this thesis that is stored on enclosed media device.

```
/                           root
  apendix                   all support files of thesis
    batery_pack_model       3D visualisation of differential chassis
    Encoder                 STM32F4 source codes, Kein v5 IDE
    images                  images used in thesis appendix
    kalman_filter           experimental implementation of Kalman filter
    kinect_v2_holder        3D visualisation of differential chassis
    matlab_scripts          differential chassis motion model and visualization tools
    PCB_odometry_vehicle    STM32F4 extention shield design
  latex                     latex source codes
    images                  images used in thesis text
    loga                    support images
    pdf                     support pdfs
    text                    thesis chapters
  papers                    all papers studied during thesis
  scans                     3D models and and telemetry of SLAM runtime
```