Building 3D Dense Reconstructions using LiDAR from a Walking Robot

Marcelo Gennari do Nascimento

Wadham College, University of Oxford

February 7, 2018

Abstract

Abstract text goes here.

Contents

1 Introduction

Autonomous robots are going to be one of the major achievements of science to the benefit of the public. Autonomy though depends on two main problems that are tightly related to each other: Localization and Mapping. The first concerns the problem of localizaing a robot in an environment given a map as a prior. The second concerns the problem of mapping the environment given a prior robot's trajectory. Most of the time though, neither the trajectory nor the map is known a priori, and they need to be built simultaneously.

Since the 1986 IEEE Robotics and Automation Conference, researchers have framed the general problem of Simultaneous Localization and Mapping (SLAM) as the "holy grail" of modern robotics [1]. A reliable solution to this problem would make autonomy one step closer to reality. Since the conference, a number of algorithms have been developed that successfully tackle SLAM, each of them with their advantages and drawbacks. Modern methods of localization are able to predict the position of a walking robot to up to 2cm [2].

In order to make the map built have significant meaning and be of use to people, it is necessary to reconstruct it in 3D (or volumetrically). A 3D reconstruction system would give geometric, semantic and graphical meaning to maps, which then can be used for augmented or virtual reality.

Volumetric reconstruction relies heavily on tracking the robot's position, since the observation accuracy is independent from other observations but bounded by the tracking accuracy. Therefore, by using modern techniques to solve the problem of SLAM, it would be possible to build a reliable and realistic map of an environment without any prior information about how the environment is structured. Obvious direct applications for such a system would be reconaissance, search and rescue, and transportation.

1.1 Aim of the Project

This project is concerned with putting together state of the art algorithms for SLAM and reconstruction systems to reliably and effiiciently build a 3D map of an environment with LiDAR data from a walking robot without any prior map or trajectory available. Since the robot is most likely to operate indoors and in situations where no off board sensor is available, it was decided to not use any wirelessly transmitted information, such as GPS (Global Positioning System) or Motion Capture Systems.

Even though many papers have been published in the individual building blocks that form the components of this project, it is harder to find academic reports that put all of the state of the art algorithms together to form a working system.

1.2 Organisation of the Report

Section 2 of the report will go through the literature review of the main building blocks of the project. In particular, we will explore the current developments in the solutions of the Simultaneous Localization and Mapping (SLAM), the Iterative Closest Points (ICP) and 3D Reconstruction problem.

Section 3 will go deeper on the system pipeline of the project. In order to get a general understanding of the system, detailed explanation of the steps of data processing, inputting and outputting will be explained. This will also introduce the two main parts of the system that will be explained in subsequent sections and analyse the tools used in the project.

Section 4 explores the solution of the SLAM problem adopted in the project. Since building an accurate map is an essential step to the success of the project, this section will explain the details of how the SLAM solution works and why this particular structure was chosen.

Section 5 shows how the output from the SLAM solution will be integrated with the 3D reconstruction system.

Section 6 will conclude the report with an overall evaluation of how the system performed. It will also indicate ways in which the system could be improved in a subsequent project. An analysis of how the outcome of this project compares with similar systems will be provided.

2 Literature Review

2.1 Simultaneous Localization and Mapping (SLAM)

SLAM is the problem of whether it is possible for a mobile robot to create a globally consistent map of an environment and localize itself on it without prior knowledge of the map [1][3]. Building a map of an environment is a crucial step towards autonomy, since planning and control assume prior knowledge of mapping and localization. Mathematically, we can frame SLAM as a Markov Chain, a Bayes Net or a Factor Graph. Defining:

- $\mathbf{x_k}$: the pose of the robot (being $\mathbf{X_{0:k}}$ as the poses from time **0** to **k**)
- $\mathbf{u_k}$: the odometry measurement ($\mathbf{U_{0:k}}$ as the historical measurements)
- $\mathbf{l_k}$: the landmark observation ($\mathbf{L_{0:k}}$ as the historical landmarks)
- $\mathbf{c_k}$: the loop closures

It is possible to formulate the problem of SLAM more formally using both a factor graph or a probabilistic framework:

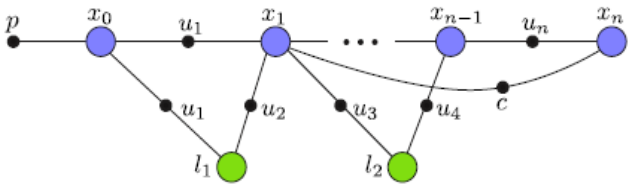

Figure 1: SLAM as a Factor Graph. The value **p** denotes a prior; $\mathbf{x_n}$ denotes the state vector; $\mathbf{u_n}$ denotes odometry measurements; **c** denotes close loops; $\mathbf{l_n}$ denotes landmark positions.

The graph shown in Figure 1 is a Factor Graph representation of the dependencies between variables and measurements. A probabilistic framework can then be extracted from it.

$$P(X, L, U, Z) \propto P(x_0) \prod_{i=1}^{M} P(x_i|x_{i-1}, u_i) \prod_{k=1}^{K} P(z_k|x_{i_k}, l_{j_k}) \tag{1}$$

The general goal of probabilistic SLAM is to find variables $\mathbf{X}^*$ and $\mathbf{L}^*$ that maximizes the posterior probability distribution $P(X, L \mid U, Z)$ of the Equation 1. Two models of the probabilities given above are used commonly in SLAM to tackle this problem: the Process Model (also known as Motion Model in mobile robotics) and Observation Model:

Process Model :

$$x_i = f(x_{i-1}, u_i) + w_i$$

Observation Model :

$$z_i = h(x_{i_k}, l_{j_k}) + v_k$$

where w_i is white noise with covariance Q and v_k is white noise with covariance R. The process (motion) model is usually a generalization of how the robot moves (kinematics) [4][5]. The observation model is a probabilistic representation of the performance of the sensors used to collect the data. Notice that both models are probabilistic and thus define the probabilitiy distributions $p(x_i|x_{i-1}, u_i)$ and $p(z_k|x_{i_k}, l_{j_k})$.

Under this framework, the mathematical analysis of SLAM has been shown that it is indeed feasible to build a nondivergent map with no prior information, so it is widely accepted that in theoretical grounds, SLAM is a solved problem [1][3][6][7]. However, there are computational and algorithmic challenges that hinders the development of a real-time implementation system that performs SLAM. This problem gets even more complicated when considering unstructured environments and large scale maps [8].

There are three main implementations of SLAM: Kalman Filtering and Extended Kalman Filtering (with early works such as proposed by R. Smith [9]), Particle Filtering (most notably with FastSLAM [4]) and Information Filter (with the now state-of-the-art work of iSAM from Michael Kaess [10]).

2.1.1 Kalman Filtering

As one of the first implementations to appear to solve the problem of SLAM, the Extended Kalman Filtering (EKF) approach makes two approximations when formulating the probabilistic SLAM: both the Process Model and the Observation Model are linearized about a suitable linearization point $\hat{x}$.

Subsequently, the probability distributions $p(x_i|x_{i-1}, u_i)$ and $p(z_i|x_i, l_i)$ are modeled as Gaussians with the mean as $\nabla f|_{x=\hat{x}}$ and $\nabla h|_{x=\hat{x}}$ and covariances Q and R. With this framework in place, a recursive two-step method can be found to update the posterior probability distribution at every iteration [1]:

$$\textbf{Prediction Phase}: \; P(x_i, L|Z_{0:i-1}, U_{0:i}) = \int P(x_i|x_{i-1}, u_i)P(x_{i-1}, L|Z_{0:i-1}, U_{0:i-1})dx_{i-1}$$

$$\textbf{Update Phase}: \; P(x_i, L|Z, U) = \frac{P(x_i, L|Z_{i-1}, U)P(z_i|x_i, L)}{P(z_i|Z, U)}$$

The Prediction Phase concerns the motion model, where an update of the estimated position of the mobile robot is made taking into account only the controls and kinematics of the robot itself. This is followed by an Update Phase, where the position of the robot is recalculated based oan the observation of a landmark, for example. This algorithm is then recursively applied for every time-step i.

Since the product of two gaussians is a gaussian, the probability density function $p(x_i, L|Z, U)$ will remain Gaussian at all times, and a closed loop solution using just the mean and the covariance matrices can be found [5].

Analysis of the EKF algorithms have shown that due to linearization of the functions f and h, assumptions about Gaussian Process and Observation Model can cause the EKF solution to perform poorly unless many loop closures are detected in frequent intervals [11].

It is also known that the Kalman Filter approach requires storage of the order of $O(N^2)$ (where N is the number of features), and for a classic implementation of the algorithm, it also requires computational power of the order of $O(N^2)$ [6]. New methods for computing the covariances (which cause the squared dependence) by exploring state augmentation, partitioned updates and sparsity in the matrices have demonstrated faster solutions, thus requiring less computational power [8].

2.1.2 Particle Filtering

In order to avoid linearlization of nonlinear models, Particle Filtering (PF) has been a popular method to integrate non-gaussian distributions in the estimation [4][5]. The basis of particle filtering comes from Dallert's proposal of a Monte Carlo Localization (MCL) algorithm [5]. In sampling methods, the probability distributions are defined in function of the density of particles along the distribution. This approach avoids the assumption of linearity and Gaussian distribution, which makes this algorithm embrace multi-modal distributions.

Just like the EKF solution, there are many ramifications of PF methods. However, most of them follow the basic structure of the FastSLAM algorithm [4]. This algorithm breaks the SLAM problem in one of localization over the robot's path, and k of landmark location, where k is the number of landmarks. The Localization problem is solved using MCL, which is composed of two parts [5]:

Prediction Phase: In this part, N number of particles are drawn from the Motion Model distribution, whose density asymptotically represents the proposal distribution $P(X, L|Z_{0:i-1}, U)$.

Update Phase: Each particle from the Prediction Phase is then given a weight which is equal to the likelihood of the particle being there given the observation. In other words, $weight = P(Z|X, L)$, which is drawn from the Observation Model.

After those two phases, the Landmark Location is solved using the classical EKF algorithm.

Particle filtering methods have advantage over the EKF for not making any assumptions about linearity or Gaussianity of the distribution. Also if implemented wisely, it can reach $O(Mlog(K))$ time, where M is the number of particles and K is the number of landmarks [4].

2.1.3 Information Filter

Information Matrices formulations of the SLAM algorithm is a technique used to compensate the quadratic dependency in computation time of the EKF by exploiting the sparsity in the Information Matrix.

It is known that the Dense Covariance Matrix in the EKF is the key to a convergent solution [1]. However, this dense matrix means that the EKF will need computational power increasing quadratically in the number of landmarks.

By adopting an information matrix formulation of the EKF (the Information Matrix is defined as the inverse of the covariance matrix, or equivalently the coefficient matrix of the least square problem), this can be reduced to constant time computation [12]. This formulation is exactly equivalent to the EKF, with the advantage of being computationally advantageous [13].

The Information Filtering and Information Smoothing approaches have had many ramifications (with special mention to the Sparse Extended Information Filter (SEIF)[12]. The main algorithm that was developed was the iSAM (Incremental Smoothing and Mapping) [10], which is the one used in this project for being light and computationally advantageous.

2.2 Stereo Visual Odometry

Visual Odometry concerns the problem of estimating the robot's pose using its camera sensors. When two callibrated cameras are used, it is called Stereo Visual Odometry. It is a useful estimation procedure that can substitute wheel (kinematics) odometry in cases it is not available or it is not accurate (e.g. in rough terrain where the wheels slip).

The standard Stereo Visual Odometry algorithm works as follows [17]:

1. Preprocessing Images: rectify images so that epipolar lines are aligned in left/right images; smooth images with an edge preserving filter such as the bilateral filter; calculate disparity map, by Block Sum of Absolute Difference (SAD) or equivalent, which indicates the inverse range map.

2. Detect Features: use either Harris [18], FAST [19] or SIFT [20] for example, to extract features in the images.

3. Match Features: by using a Score Matrix from the disparity map.

4. Estimate Motion: by means of reprojection and triangulation of the calibrated cameras.

Stereo Visual Odometry is relatively accurate when cameras are properly calibrated, and results of the order of 0.25% accuracy over 400m have been achieved using only the pure algorithm [17]. The tool used in this project was fovis [26], which is described in [25].

2.3 Iterative Closest Points (ICP)

The problem that ICP is trying to solve: given a point cloud in a sensor coordinate frame that is a subset of a complex shape of another point cloud in a model coordinate frame, what is the translation and the rotation that aligns, or registers, the clouds by finding the minimum of a distance metric?

Given a point cloud in the sensor reference frame $P = \{p_1, p_2, p_3 ... p_{N_p}\}$, which is a subset of the point cloud in the model reerence frame $X = \{x_1, x_2, x_3, ... x_{N_x}\}$, and assuming that the correspondence P to X is known and is $C = \{(x_1, p_1), (x_2, p_2), ..., (x_N, p_N)\}$, the minimum square error be formulated with the following equation:

$$f(\overrightarrow{q}) = f(\mathbf{R}, \overrightarrow{q_t}) = \frac{1}{N} \sum_{i=1}^{N} ||\overrightarrow{x_i} - \mathbf{R}(\overrightarrow{q_R})\overrightarrow{p_i} - \overrightarrow{q_t}||^2 \tag{2}$$

Equation 2 is a function of the Rotation Matrix $\mathbf{R}$ and the translation vector $\overrightarrow{q_t}$. It is computationally cheaper to define the Rotation Matrix as a function of the quaternion q_R, as it only requires 4 variables instead of the 6 needed for the Rotation Matrix. If we define the vector $\overrightarrow{q} = [\overrightarrow{q_R}|\overrightarrow{q_t}]^T$, then we can optimize the above as a function of the 7 variable vector $\overrightarrow{q}$ [14]. This vector would then define a translation from P to X that would minimize the objective function.

Using that equation, the ICP algorithm can be implemented using the following iterative operation:

1. Find the correspondeces between points X and P by evaluating closest points in each of the points of the smallest set.

2. Compute the registration by finding $\underset{\overrightarrow{q*}}{min} f(\overrightarrow{q})$ and apply the transformation to the whole set P.

3. If $f(\overrightarrow{q*}) \leq \tau$, where τ is a threshold value, then stop. Otherwise go back to step one.

The ICP algorithm formulated with the above objective function always converges monotonically to the nearest local minimum [14]. The global minimum is more difficult to find, since it depends on the relative initial pose of the model and the sensor reference frames. Thus, given an adequate set of initial translation and rotation, one can globally minimize the mean-square distance over all six degrees of freedom [14].

Since first formulated, a number of variants of the ICP appeared. They are usually alternative ways of selecting subsets of the points, finding correspondences, weigthing correspondences, rejecting specific pairs, assigning an error metric, computing the minimum of the objective function, or a combination of those [15]. The variant used in the project was the AICP, which rejects point correspondences based on the overlap between point clouds [16].

2.4 3D Reconstruction Systems

Although point clouds can be very useful for tasks such as correction of poses, they do not provide for higher level of scene understanding. 3D Reconstruction Systems are used for that: given discrete data from a sensor, the 3D Reconstruction System tries to recreate the scene geometrically by estimating surfaces and occupancy, and graphically by providing texture.

In order to represent surfaces, there have been a couple of algorithms that have been used. The Truncated Signed Distance Function was popularized by KinectFusion and uses a voxel grid with a value in each voxel which represents the distance to the nearest. When two adjacent voxels change sign (or is zero), it means that there is a surface in there, which is then reconstructed. Another popular approach is using surfels (surface elements). Another popular is quadrics.

There are mainly three classes of recunstruction systems that can be divided:

- Active vs Passive Sensors: when using an active sensor (such as Kinect to get RGB-D images), the depth estimates are accurate enough so that a simple fusion using weights is enough to get a good recontruction (example system KinectFusion); when using a passive sensor (such as in DTAM and PTAM, the depth estimates are usually less accurate and thus there is a need to use a regularizer

- Object-Centric vs Mobile-Robot-Centric Fusion: when in an object-centric situations, it can be assumed that the voxel block and the voxel grid will be seen all of the time by the sensors (such as in KinectFusion). Also, the object is seen in loops multiple times, thus allowing for fine detailed reconstructions; this is not true for a mobile-robot-centric situation, where the robot

navigates through the scene instead of around it, and new voxel blocks have to be dynamically allocated depending on the robot's trajectory (such as a bit in Kintinuous, even though it still uses fixed voxel blocks). It is also unlikely that the same scene is going to be seen multiple times in a range of different angles.

- Large vs Small Scale: memory allocation plays a huge part when large scale models are used, and that comes at a cost of usually higher voxel sizes, whereas in small scale very small voxel sizes and details can be preserved. Systems that took memory in consideration is the Hashing Voxel Grid (HVG)

Due to the RGB-D sensor becoming widely available as a commodity, most 3D reconstruction systems have focused on handheld devices for small scale reconstruction. Since this project relies on LiDAR data, a new system has been chosen.

The system used in this report is BORG-CUBES, which is a reconstruction system that uses sensor-agnostic voxel grids to recustruct the system. Since LiDAR data come at way less frequency than RGB-D data, BORG-CUBES relies on a prior regularizer to improve the quality of the reconstruction.

2.5 Past Work on 3D Reconstruction using LiDAR

3 System Pipeline

The pipeline for the overall system can be seen below. It consists of 4 modules: (I) First odometry estimates (based on either Visual Odometry or Wheel Odometry); (II) Laser Odometry (using AICP [16]); (III) Loop closure detection and Graph Optimization (using iSAM [10]); (IV) 3D dense volumetric reconstruction (using BOR^{2}G-CUBES [21][22]).

These 4 modules are broken down in two parts: the first part is concerned about solving the SLAM problem, which outputs a reliable trajectory and map. The second part is concerned about inputting that to a 3D reconstruction system (which in this case if BOR^{2}G-CUBES) to build the environment volumetrically. Figure 2 shows the overall system built.

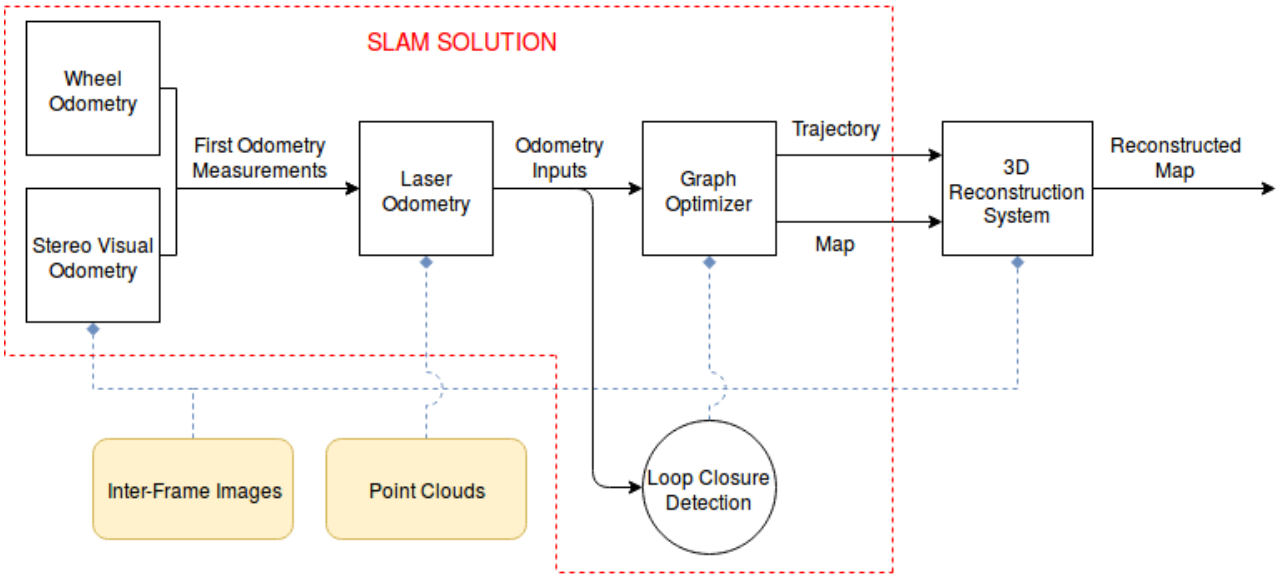

Figure 2: Diagram of the overall system pipeline. The beige boxes represent data collected from a sensor, the white boxes represent data being processed and the output being input to the next box. The rounded box is a module built for data association.

The data was collected using a LiDAR sensor (MultiSense SL or Velodyne) to get the Point Clouds at every iteration.

4 SLAM Solution

As already introduced in Section 3, the SLAM solution has 4 independent building blocks. This offers flexibility enough to test the effect that each part has in the outcome of the map. This section is going to discuss deeper how the solution works with all the building blocks working together.

4.1 First Odometry Measurements

The first odometry measurements offer a relatively inaccurate estimation of the pose of the robot at a certain time. It is used as the input pose to the AICP algorithm in order for it to process laser odometry based on the LiDAR scan.

During the project, two mehods to get the first odometry measurements were used. When getting data with the mobile robot Husky [23], the wheel odometry provided by the system served as a good first estimate of its position. When getting the data with the MultiSense SL [24], Stereo Visual Odometry from MultiSense's stereo camera was performed to get the first odometry estimations.

Stereo Visual Odometry

The Stereo Visual Odometry used is based on the system described at [25] and implemented with the fovis library [26]. Figure 3 shows one frame of the result of the visual odometry system when applied to the dataset collected for this project. The inputs came from the stereo camera in the MultiSense SL, collected at 30Hz in a relatively illuminated indoors scenario.

Wheel Odometry

Remember to put the pictures of the Wheel Odometry and the corrected pose using AICP.

Just like with the stereo odometry and as common with systems based on proprioceptive sensors without loop closing mechanism, this method of locomotion drifts significantly. After applying this algorithm to the pose of the robot, the results of the

Also a good idea would be to put the Wheel Odometry and the Visual Odometry as comparison using the same Dataset (the one that Simona Collected at Edinburgh would be perfect)

4.2 Laser Odometry

Given the first odometry measurements, the AICP algorithm was used to incorporate the LiDAR scan of each of the paths to the estimation problem. For the Husky example, the Velodyne [27] LiDAR

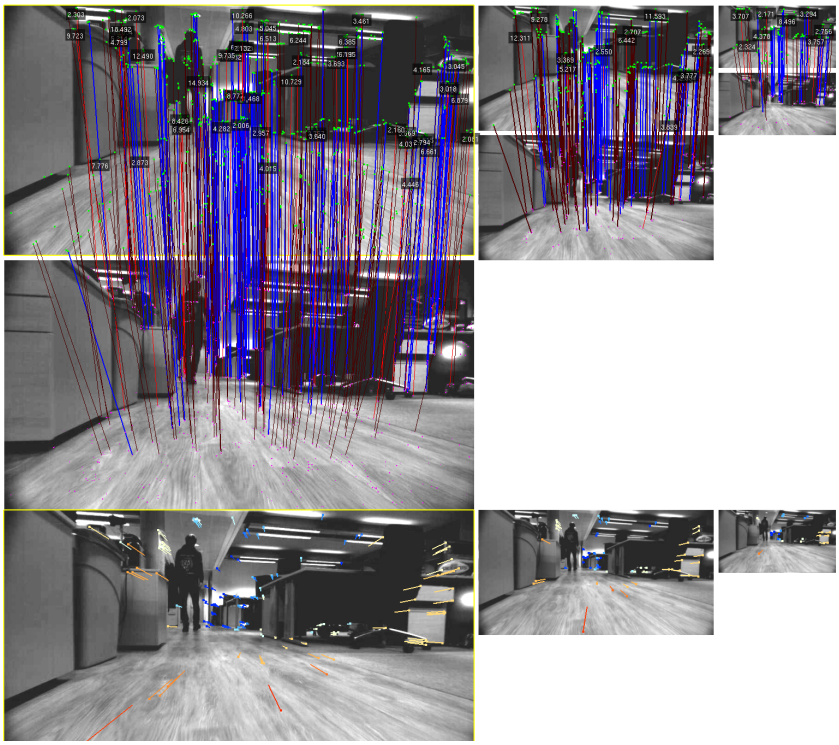

Figure 3: Result of fovis library to the data collected at the Information Engineering Building at University of Oxford. The first row of images show a Gaussian Piramide to extract point features. The Key Frame is shown in the second row where features extracted are matched back to the first row images. Red lines indicate outliers and blue lines indicate inliners (bad and good correspondence respectively). The last row shows the result of the rotation matrix in the pictures. The scale of the movement is indicated as a colour scale from blue to red.

was used, whereas for the Oxford Dataset, the MultiSense SL LiDAR was used.

The results for each of the paths are shown below for comparison. Figure 4 shows the results for the data collected with Husky using Wheel Odometry. It is noticeable the difference between the accuracy of the First Odometry Measurements to the Laser Odometry. However, it can be seen that there is drift due to how the AICP algorithm works. This can be further corrected by implementing Loop Closures.

4.3 Loop Closure Detection

In order to adjust for the innacuracies of the path estimate, a loop closure detection algorithm was placed. Since it was assumed that the robot would not overdrift, three simple heuristics were used to detect loop closure: Time Filter Sampling, Euclidean Distance, and Pose Alignment.

- **Filter Sampling:** In order to avoid the problem of loop closing two state estimates that are in the same room but did not leave the room, a time filter sampling was applied. The idea is to sample every S state estimates, so that the candidate loop closing states would be sparse enough so that the loop just occurs between big time frames, but dense enough to apply euclidean distance.

- **Euclidean Distance:** Once the estimates have been sampled, the euclidean distance between each of them is applied. If the distance between any two states is less than a threshold value E, then the two estimates form a candidate pair and they are added in the candidate pair list.

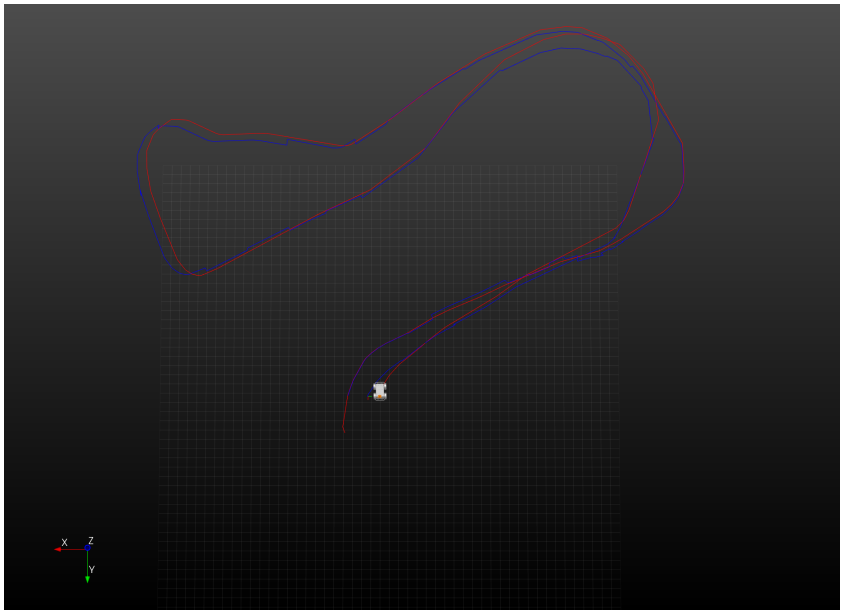

Figure 4: Result of the Wheel Odometry feeding the laser odometry system. The red path shows the Wheel Odometry and the green path shows the corrected poses after being computed by the AICP algorithm. Notice that the corrections are more obvious during the turns, when the wheel odometry is particularly innacurate.

- **Pose Alignment:** After the resampling, the candidates pairs are tested for their relative pose. Since the loop closure correction relies on alignment of point clouds using ICP, it is a good strategy to select poses that would maximize the overlap between the point clouds associated with the candidates. In order to do that, another resampling is done based on the overlap between the field of view of the estimates. If the overlap is less than a threshold value O, then the candidate pair is dropped out from the list, and the remaining are the loop closures that are considered.

The parameters S, E and O are chosen by hand depending on what dataset is used in the algorithm. Changing the parameters can affect the outcome of the map significantly, so it is important to spend some time tuning them to the specific requirements of the test.

Figure 5 shows the results of the Loop Closure detection when applied to the Husky dataset. Due to the relatively low velocity of the mobile robot, the parameters were adjusted to match the time that the mobile robot would take to leave a room before coming back to the same room.

It is also important to notice that marginally different parameters have an impact on the number of loop closures and on the outcome of the map. Figure X shows the results of different maps according to different settings of the loop closure parameters.

Ideally, the number of loop closures should be neither too big nor too small. Loop closure can be thought of strings "tightening" the Factor Graph in Figure 1 by adding constraints to it. If many loop closures are made, the graph becomes to constraint and new corrections to the map would be

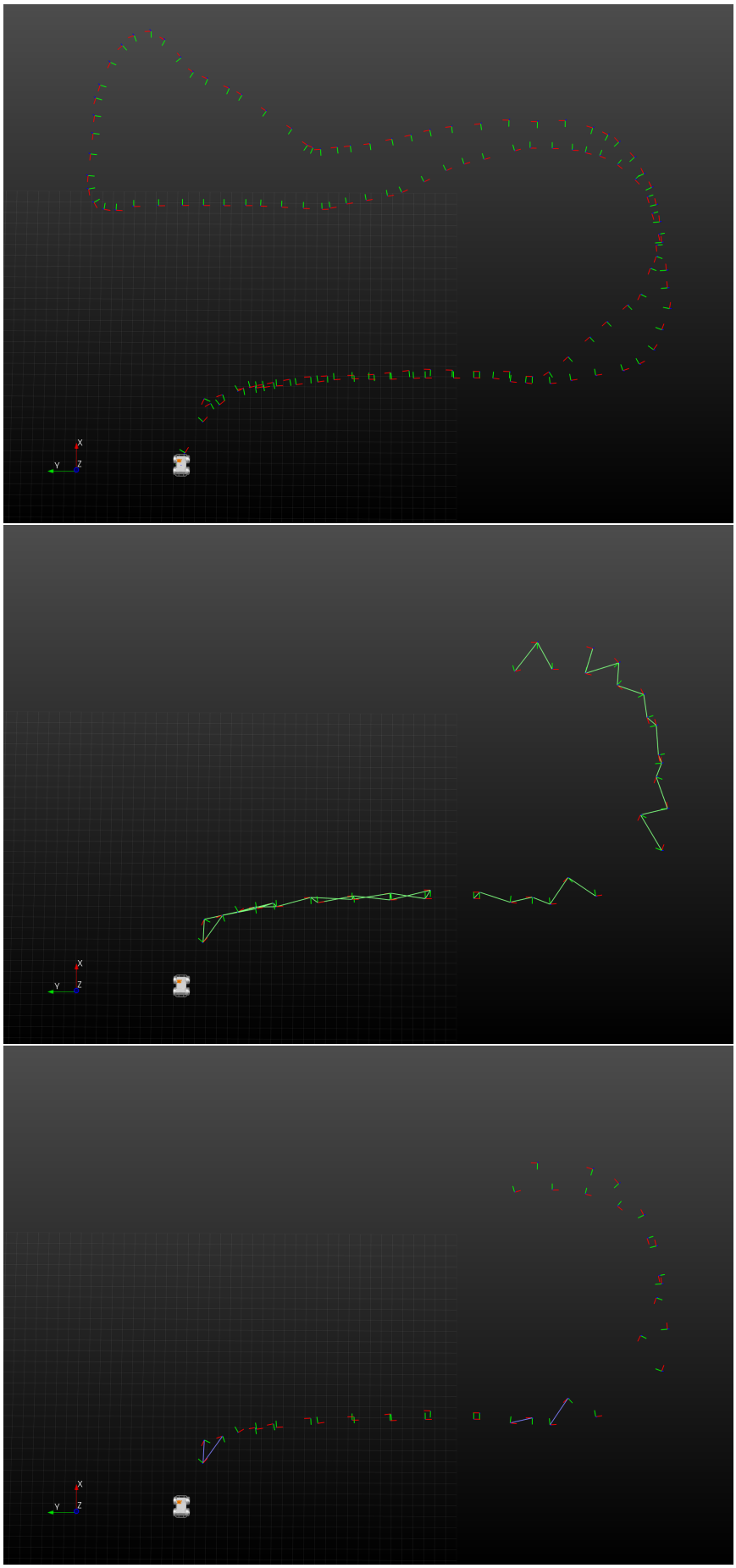

Figure 5: The three heuristics for loop closure detection. Data shown was collected using the Husky Mobie Robot and the Velodyne LiDAR, visualized in bird's-eye view with red and green coordinates representing the path of the robot. The figure on the top shows the result of the time sampling. The plot in the middle shows the Euclidean Distance Pairings. The last figure is the result of the Pose (overlap) filter. In this experiment, $S = 2$, $E = 2$ and $O = 0.5$.

insignificant. This can be naively desirable since it means that the certainty of the graph is high, but if one loop closure is a bit off, then the effect cannot be corrected subsequently. In the other hand, if few loop closures are made, then the drift between poses are not corrected substantially and the overall error in the graph is high.

Therefore it is crucial to tune the parameters so that meaningful loop closures are selected.

4.4 Graph Optimization

The Graph Optimization was conducted using the iSAM library. Since the exteroceptive model used was LiDAR data instead of any landmark localizer (like Visual Data or Fiducial Systems for example), a Pose-Only graph is generated. It is important to notice that all of the SLAM properties for landmarked based systems also hold for Pose-Only based systems [lacking citation]. The results of the Wheel Odometry system when applied the Optimization are shown in the Figure 6.

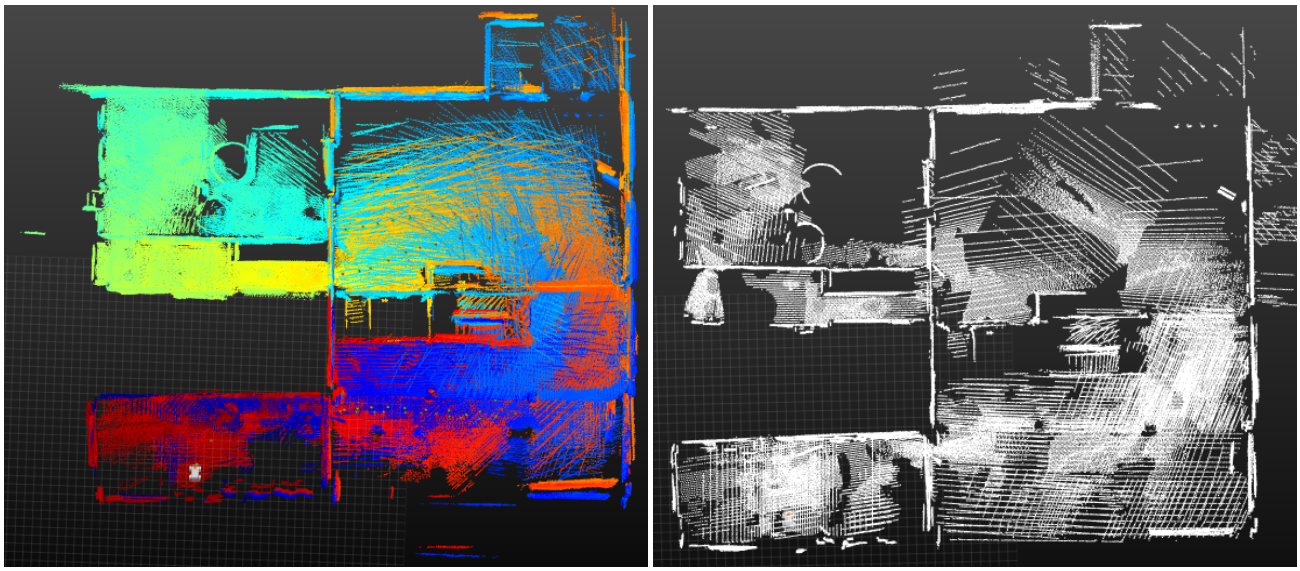

Figure 6: Result of the Graph Optimization solution. The map in the left shows the Point Clouds collected without Graph Optimization and Loop Closures applied. The map in the right shows the result of the map after the loop closures have been implemented in the iSAM algorithm. Notice the alignment of the wall at the top of both images and at the very bottom to compare the overall drift of both systems. The point clouds in the picture in the left are coloured with timestamps.

5 Integration with BOR^{2}G-CUBES

6 Conclusion

References

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics Automation Magazine*, vol. 13, pp. 99–110, June 2006.

[2] M. F. Fallon, M. Antone, N. Roy, and S. Teller, "Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing," in *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 112–119, Nov 2014.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past,present, and future of simultaneous localization and mapping: Towards the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[4] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 593–598, AAAI, 2002.

[5] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, pp. 1322–1328 vol.2, 1999.

[6] M. Csorba, "Simultaneous localisationa and map building," 1997.

[7] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 229–241, Jun 2001.

[8] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): part ii," *IEEE Robotics Automation Magazine*, vol. 13, pp. 108–117, Sept 2006.

[9] R. Smith, M. Self, and P. Cheeseman, "Autonomous robot vehicles," ch. Estimating Uncertain Spatial Relationships in Robotics, pp. 167–193, New York, NY, USA: Springer-Verlag New York, Inc., 1990.

[10] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. on Robotics (TRO)*, vol. 24, pp. 1365–1378, Dec. 2008.

[11] S. Huang and G. Dissanayake, "A critique of current developments in simultaneous localization and mapping," *International Journal of Advanced Robotic Systems*, vol. 13, no. 5, p. 1729881416669482, 2016.

[12] *Closed form solutions to the multiple-platform simultaneous localization and map building (SLAM) problem*, vol. 4051, 2000.

[13] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research, IJRR*, vol. 25, pp. 1181–1204, December 2006.

[14] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," vol. 14, pp. 239–256, 04 1992.

[15] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152, 2001.

[16] S. Nobili, R. Scona, M. Caravagna, and M. Fallon, "Overlap-based icp tuning for robust localization of a humanoid robot," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4721–4728, May 2017.

[17] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," pp. 3946 – 3952, 10 2008.

[18] C. Harris and M. Stephens, "A combined corner and edge detector," p. 50, 01 1988.

[19] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006*, (Berlin, Heidelberg), pp. 430–443, Springer Berlin Heidelberg, 2006.

[20] D. Lowe, "Object recognition from local scale-invariant features," vol. 2, pp. 1150 – 1157 vol.2, 02 1999.

[21] M. Tanner, P. Piniés, L. M. Paz, and P. Newman, "BOR2G: Building Optimal Regularised Reconstructions with GPUs (in cubes)," in *International Conference on Field and Service Robotics (FSR)*, (Toronto, ON, Canada), June 2015.

[22] M. Tanner, P. Piniés, L. M. Paz, and P. Newman, "DENSER Cities: A System for Dense Efficient Reconstructions of Cities," *ArXiv e-prints*, Apr. 2016.

[23] Clearpath Robotics, "Husky a200," 2018.

[24] Carnegie Robotics, "Multisense sl," 2018.

[25] P. H. M. K. D. M. D. F. Albert S. Huang, Abraham Bachrach and N. Roy, "Visual odometry and mapping for autonomous flight using an rgb-d camera," *Int. Symposium on Robotics Research (ISRR)*, August 2011.

[26] "fovis." http://fovis.github.io/. Accessed: 04-02-2018.

[27] Velodyne LiDAR, "Velodyne," 2018.