24th COBEM - 2017

24th ABCM International Congress of Mechanical Engineering
December 3-8, 2017, Curitiba, PR, Brazil

COBEM-2017-0296
SIMULTANEOUS LOCALIZATION AND MAPPING OF A MOBILE ROBOT USING A KINECT V2 SENSOR IN INDOOR ENVIRONMENTS

João Carlos Virgolino Soares
Marco Antonio Meggiolaro
Pontifical Catholic University of Rio de Janeiro, Gavea - Rio de Janeiro, RJ - Brasil
joaovirgolino@aluno.puc-rio.br
meggi@puc-rio.br

Abstract. *This paper presents the development and implementation of a 3D-SLAM System for indoor applications in a commercial differential drive mobile robot equipped with a RGB-D time-of-flight sensor. A visual odometry approach is used to estimate the pose of the robot in a global coordinate system and, simultaneously, a Point Cloud Map is generated with the depth and color information from the sensor. A Graph Optimization based on Maximum Likelihood Estimation is used to reduce the errors accumulated in the visual odometry and compute a consistent map and trajectory. The Robot Operating System (ROS), an open-source framework for writing robot software, is used as middleware.*

Keywords: *Graph-SLAM, Wheeled Mobile Robot, RGB-D Sensor, ROS*

1. INTRODUCTION

Mobile robotics has a wide range of applications, including autonomous vehicles, mobile industrial robots, and home-service robots. Autonomous navigation is currently one of the most challenging subjects in mobile robotics, due to the high uncertainty and non-linearity inherent to unstructured environments, sensor measurements, and robot motion. Such uncertainties can be dealt with using Probabilistic Robotics. Probabilistic algorithms are very suitable for robotic applications to explicitly represent uncertainty using probability theory. Therefore, several error sources usually ignored are considered in these models.

To achieve full autonomy and enable navigation, the robot must have knowledge of its own pose with respect to the global coordinate system, as well as an accurate map of the environment. As a result of the interdependence of these two requirements, the problem is known as Simultaneous Localization and Mapping (SLAM).

The most broadly used sensors for SLAM applications are the laser range-finders, due to high range and precision. However, their cost is a limitation to certain research projects and commercial products. RGB-D cameras are interesting alternatives to laser scanners, because of their lower price and weight, despite having more limited measurement range. The Kinect (Microsoft, 2015) is a highly popular RGB-D sensor created for the gaming industry, and later incorporated into robotics research as a choice for SLAM, obstacle avoidance, and 3D reconstruction.

The RGB-D camera provides color and depth information that can be used to estimate the robot motion between frames, a process called Visual Odometry. With the known trajectory and sensor information, it is possible to create a global 3D map. However, a probabilistic algorithm is necessary to deal with the sensor noise. Introduced by Lu and Milios (1997), the Graph-SLAM formulation is a probabilistic approach that represents state variables as nodes in a graph and the edges are measurement relations affected by Gaussian noise (Kümmerle *et al.*, 2011). The optimization of this graph is used to obtain a Maximum Likelihood Solution for the poses and the map (Endres *et al.*, 2014).

This paper presents the development and implementation of a 3D-SLAM system for indoor applications using a commercial differential drive mobile robot iRobot Create (iRobot, 2006) equipped with a Kinect v2 RGB-D time-of-flight sensor. A visual odometry pipeline is used to estimate the pose of the robot in a global coordinate system, a Point Cloud Map is generated with the color and depth sensor information and a Graph-based maximum likelihood estimation method is applied to deal with the uncertainty. This approach uses the Robot Operating System (ROS) as middleware, an open-source framework for writing robot software (Quingley, 2009) that assures a singular integration and control of low and high-level tasks and devices through a graph-based architecture. The open-source libraries OpenCV and Point Cloud Library (Holz and Ichim, 2015) are used in the registration pipeline.

This paper is organized as follows. Section 2 introduces the theoretical background applied in the SLAM problem. Section 3 presents a numerical evaluation of the system based on a benchmark dataset. Section 4 presents the results of the analysis using real data collected from the iRobot Create in indoor environments. Section 5 presents the conclusions and suggestions for improvements and future work.

2. METHODOLOGY

The SLAM system is composed of two main parts: front-end and back-end. The front-end processes sensor information and compute geometric relations to estimate the pose of the robot between states, providing visual odometry. First, visual features are extracted from color images provided by the RGB-D sensor. These features are matched with features from the previous frame. A filter is applied to remove outliers and the remaining good features are matched. Combining both depth and color images, the transformation matrix between the two frames is estimated using the matched correspondences (Endres *et al.*, 2012). In the back-end step, a probabilistic formulation is used to optimize the poses of the robot and create a maximum likelihood estimation of the trajectory. With the estimated trajectory, it is possible to create a global map (Endres *et al.*, 2014). Figure 1 illustrates the general pipeline of the SLAM system:

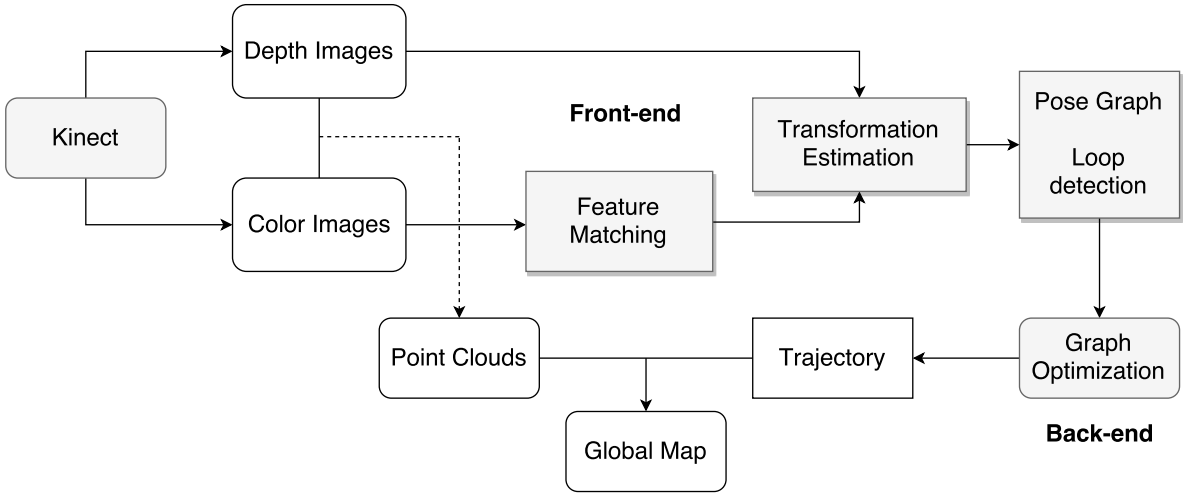

Figure 1. General Overview of the System

In this work, the library OpenCV was used for the keypoint detection, feature extraction and matching steps, and the Point Cloud Library was used for the global map registration and visualization. The library $libfreenect2$ (Wiedemeyer, 2015) was used as a driver for kinect and the library $iai2kinect$ (Blake *et al.*, 2015) as a calibration tool and a bridge between $libfreenect2$ and ROS.

ROS (Quingley, 2009) is a framework for the development of robotics software, built as a large number of small programs that communicate one another through messages. These messages can be sensor input, debug messages or control output. This setup creates a graph-based structure where the nodes are the programs (probabilistic robotics algorithms, for example) and the edges are the messages.

2.1 Front-end: Motion Estimation

2.1.1 Feature Extraction and Matching

The first step of this implementation is to use the color images provided by the RGB-D sensor to detect unique points that can be tracked, called keypoints, and extract feature descriptors that characterizes the local image region. These features need to be invariant to scale, rotation and translation, so they can be matched in sequential frames during camera motion. The Scalar Invariant Feature Transform (SIFT), presented by Lowe (2004), is a method for feature extraction that is broadly used in computer vision and robotics, and it was chosen for the present work due to its high precision.

Matching a pair of feature descriptors can be referred as a nearest neighbor matching. Given a set of points P and a query point q, the nearest neighbor search can be stated as a method to find the element in P that, under a determined metric distance, is the closest one to q (Muja and Lowe, 2014).

To match a pair of keypoint descriptors, the library FLANN (Muja and Lowe, 2009) was used for fast approximate nearest neighbor searches. In Fig. 2 it is shown two sequential color frames and its corresponding matched points.

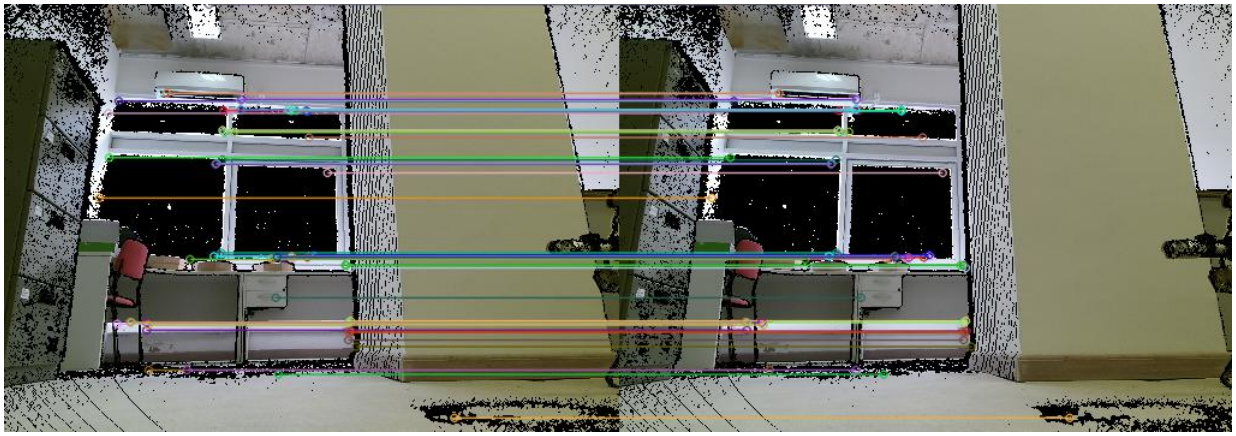

Figure 2. Feature Matching

2.1.2 Motion Estimation

Using the equations of the pinhole camera model it is possible to find the 3D correspondences between frames combining the position of the matched points from color images and their respective depth values:

$$Z = depth\_image[v, u] \tag{1}$$

$$X = (u - c_x)\frac{Z}{f_x} \tag{2}$$

$$Y = (v - c_y)\frac{Z}{f_y} \tag{3}$$

where f_x is the focal length x, f_y is the focal length y, c_x is the optical center x and c_y is the optical center y of the camera. u and v are the coordinates of the image plane.

With the established 3D correspondences, the motion estimation between the two frames is computed using a least-squares estimation method (Umeyama, 1991), which consists to find parameters R and t that minimize the mean squared error of the transformation between two sets of points X and P, as stated in Eq. (4).

$$E(R, t) = \frac{1}{N_p}\sum_{i=1}^{N_p} \|x_i - R_{P_i} - t\|^2 \tag{4}$$

The minimal value of E is found applying the singular value decomposition in the covariance matrix of X and P:

$$C_{xp} = \frac{1}{N_p}\sum_{i=1}^{N_p}(p_i - \mu_p)(x_i - \mu_x)^T \tag{5}$$

where μ_x and μ_p are the mean vector of the two sets of points.

$$\mu_x = \frac{1}{N_x}\sum_{i=1}^{N_x} x_i \tag{6}$$

$$\mu_p = \frac{1}{N_p}\sum_{i=1}^{N_p} p_i \tag{7}$$

To be robust against erroneous feature matches, the Random Sampling Consensus (RANSAC) algorithm (Fischler and Bolles, 1981) is applied to reject possible outlier features using a transformation error threshold. First, the RANSAC algorithm samples a minimal possible set of data to determine the parameters. In this case, three matched features are sampled, since it is the minimum number to compute a rigid transformation in SE(3) (Endres *et al.*, 2012). The transformation estimated from these features becomes the hypothesis. The other features are evaluated with this transformation and counted as inliers if the alignment error is below the threshold. This process is iteratively repeated and the transformation with the largest number of inliers, given the chosen threshold, is the final transformation.

2.1.3 Point Cloud Registration

Each time a transformation is computed between frames, their respective point clouds are computed using Eqs. (1-3) and the RGB information. They are stored to be further aligned using the optimized trajectory. Figure 3 shows a point cloud generated from the experiments.

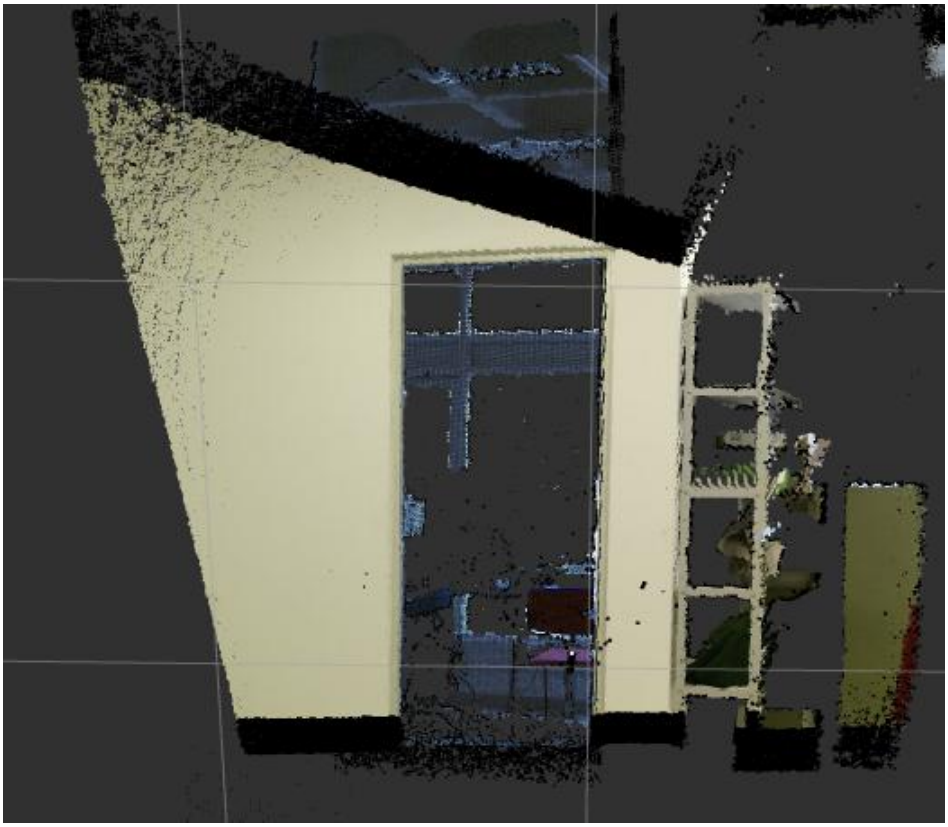

Figure 3. Point Cloud

2.1.4 Loop Closure

The motion estimation driven by the alignment between frames is inherently noisy and accumulates errors over time that cause a drift in the estimated trajectory of the robot, and, consequently, in the map. To overcome this issue, it is necessary to compare the current state of the robot with previous states to detect if the robot is in a previously visited location, a problem called loop-closure detection (Henry *et al.*, 2012).

2.2 Back-end: Graph Optimization

To deal with the uncertainty introduced by the sensor measurements and robot motion, a probabilistic formulation of the problem is necessary. The SLAM problem can be stated as an estimation of the posterior probability distribution of the state x of the robot given the measurements of the environment z, or $p(x|z)$ (Grisetti *et al.*, 2010).

The objective is to find the state $\hat{x}$ that best explains the measurements, in other words, the state x that maximizes the posterior belief $p(x|z)$: (Endres, 2015)

$$\hat{x} = argmax p(x|z) = argmax p(z|x)p(x) \tag{8}$$

If no prior knowledge is available, $p(x)$ is assumed to be constant and the problem becomes a maximum likelihood estimation. Assuming that the measurements are independent, the problem factorizes into:

$$\hat{x} = argmax \prod_{k=1}^{n} p(z_k|x) \tag{9}$$

Assuming that the measurements are locally Gaussian, the likelihoods of the measurements will be Gaussian as well:

$$\hat{x} = argmax \prod exp(-e_{ij}^T(x_i, x_j, z_{ij})\Omega_{ij}e_{ij}(x_i, x_j, z_{ij})) \tag{10}$$

where e_{ij} is the error associated with the difference between the observation and the prediction, and Ω is the inverse of the covariance matrix of the measurements. Taking the logarithm to transform the product into a sum:

$$\hat{x} = argmax \sum_{ij} -e_{ij}^T(x_i, x_j, z_{ij})\Omega_{ij}e_{ij}(x_i, x_j, z_{ij}) \tag{11}$$

$$\hat{x} = argmin \sum_{ij} e_{ij}^T(x_i, x_j, z_{ij})\Omega_{ij}e_{ij}(x_i, x_j, z_{ij}) \tag{12}$$

This problem can be represented as a graph and Eq. (12) can be minimized using the the $g2o$ framework (Kümmerle *et al.*, 2011). When the robot moves and a transformation is computed, a node is added to the graph. When a frame is matched with a previous visited frame, an edge is created and the graph is optimized.

3. DATASET EVALUATION

The numerical evaluation of the system was made using the RGB-D benchmark (Sturm *et al.*, 2012) from Technical University of Munich, which provides datasets of color and depth image sequences of a kinect sensor with the corresponding ground-truth trajectory. The sequence "freiburg1_xyz" was chosen for the evaluation, which is a sequence of 798 frames corresponding to a distance of $7.11m$ and a translational velocity of $0.24m/s$. The Absolute Trajectory Error metrics was used to evaluate the consistency of the estimated trajectory. The system achieved a root mean square error of $0.124m$ in the complete sequence.

Figure 4 compares the estimated trajectory of the robot and the ground-truth provided by the dataset for the first 300 frames of the sequence. A possible explanation for the considerably large maximum error is a wrong data association in the loop closure step. Despite that, the system was able to improve the visual odometry, as shown in Table 1.

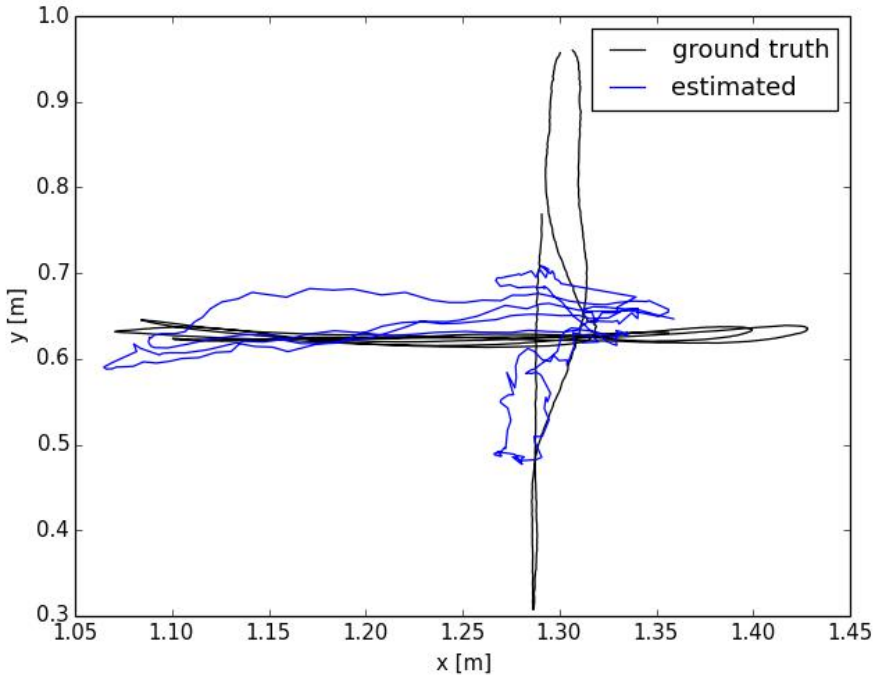

Figure 4. Dataset evaluation

Table 1. Error comparison between pure visual odometry and Graph-SLAM

	RMSE error	Mean-error
Visual Odometry	0.103	0.081
Visual Odometry + Graph-SLAM	0.088	0.070

Figure 5 shows the point cloud representation of the dataset.

Figure 5. Dataset Point Cloud

4. EXPERIMENTS

Indoor experiments were conducted in buildings from the Pontifical Catholic University of Rio de Janeiro. Figure 6 shows the two-wheeled mobile robot used in the experiments, an iRobot Create equipped with a Kinect v2. All experiments were carried out on a laptop with an Intel Core i7 6700 HQ 2.60 GHz and 16 GB of RAM running Ubuntu Linux 14.04 LTS.

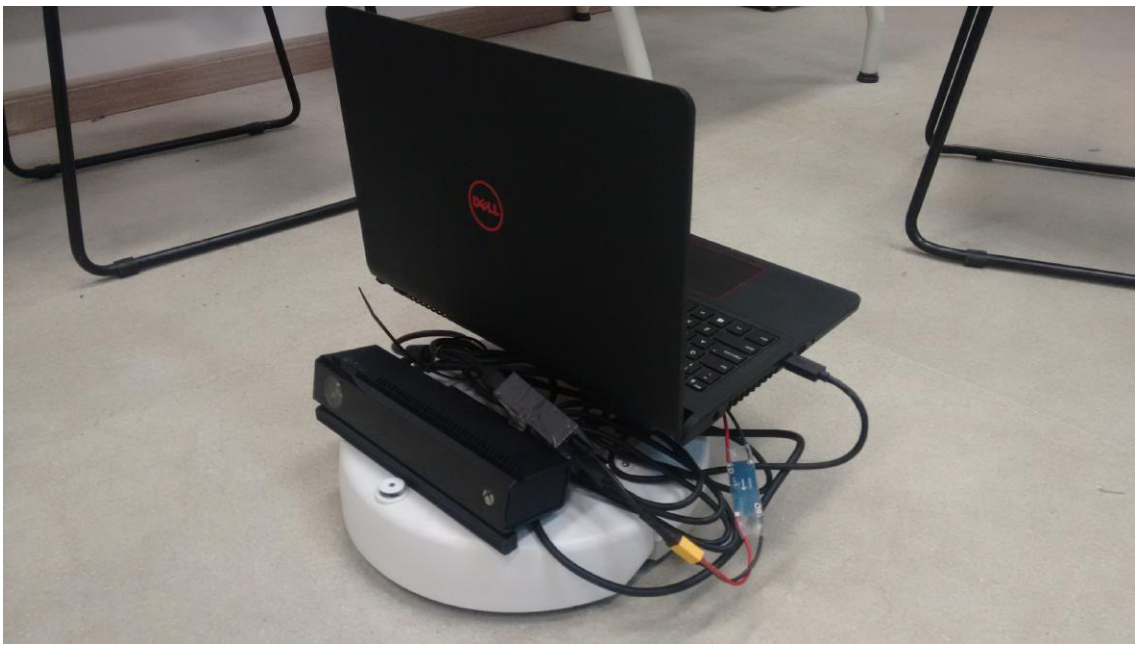

Figure 6. Mobile robot iRobot Create with Kinect v2

Figure 7 shows a global map composed of aligned point clouds after the graph optimization. The system was able to reconstruct the scene, although some drift has remained.

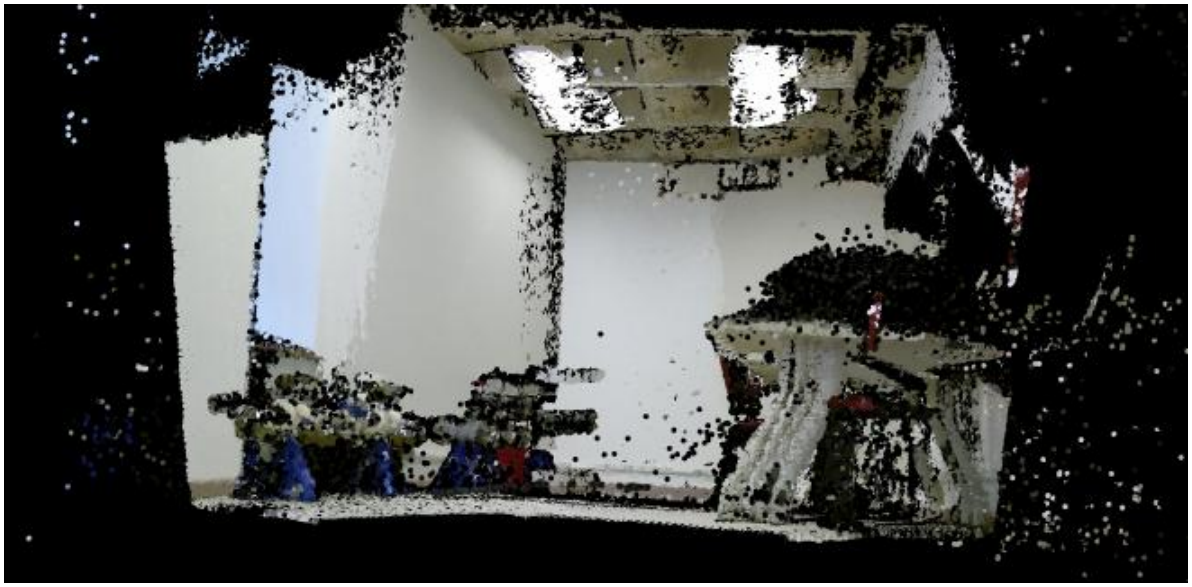

Figure 7. Point Cloud Map

5. CONCLUSIONS

In this paper, an implementation of a Graph-SLAM technique with frame-to-frame motion estimation was proposed using an RGB-D Kinect v2 sensor mounted on an iRobot Create mobile robot system. The system was able to reconstruct indoor environments and the optimization was able to improve the visual odometry. Future improvements include a better loop closure system, able to detect previous locations with more accuracy and the use of a GPU-based implementation of SIFT features to improve the computational speed.

6. ACKNOWLEDGEMENTS

This work is supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

7. REFERENCES

Blake, J., Echtler, F. and Kerl, C., 2015. "libfreenect2: Open source drivers for the kinect for windows v2 device". https://github.com/OpenKinect/ libfreenect2.

Endres, F., 2015. *Robot Perception for Indoor Navigation*. Ph.D. thesis, University of Freiburg, Freiburg.

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D. and Burgard, W., 2012. "An evaluation of the rgb-d slam system". In *IEEE International Conference on Robotics and Automation (ICRA)*. Saint Paul, MN, USA.

Endres, F., Hess, J., Sturm, J., Cremers, D. and Burgard, W., 2014. "3-d mapping with an rgb-d camera". *IEEE Transactions on Robotics*, Vol. 30.

Fischler, M. and Bolles, R., 1981. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". *Commum ACM*, Vol. 24, pp. 381 – 395.

Grisetti, G., Kümmerle, R., Stachniss, C. and Burgard, W., 2010. "A tutorial on graph-based slam". *IEEE Intelligent Transportation Systems Magazine*, Vol. 2, pp. 31–43.

Henry, P., Krainin, M., Herbst, E., Ren, X. and Fox, D., 2012. "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments". *The International Journal of Robotics Research*, Vol. 31, pp. 647–663.

Holz, D. and Ichim, A.E., 2015. "Registration with the point cloud library: A modular framework for aligning in 3-d". *IEEE Robotics & Automation Magazine*, Vol. 22, pp. 110–124.

iRobot, 2006. "irobot create". http://www.irobot.com/filelibrary/create/Create Manual_Final.pdf.

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K. and Burgard, W., 2011. "g2o: A general framework for graph optimization". In *IEEE International Conference on Robotics and Automation (ICRA)*.

Lowe, D., 2004. "Distinctive image features from scale-invariant keypoints". *International Journal of Computer Vision*, Vol. 60, pp. 91–110.

Lu, F. and Milios, E., 1997. "Globally consistent range scan alignment for environment mapping". *Autonomous Robots*, Vol. 4, pp. 333 – 349.

Microsoft, 2015. "Kinect v2". http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx.

Muja, M. and Lowe, D., 2009. "Fast approximate nearest neighbors with automatic algorithm configuration". In *Interna-*

tional Conference on Computer Vision Theory and Application.

Muja, M. and Lowe, D.G., 2014. "Scalable nearest neighbor algorithms for high dimensional data". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 36.

Quingley, M., 2009. "Ros: an open-source robot operating system". In *IEEE International Conference on Robotics and Automation (ICRA)*.

Sturm, J., Engelhard, N., Endres, F., Burgard, W. and Cremers, D., 2012. "A benchmark for the evaluation of rgb-d slam systems". In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*.

Umeyama, S., 1991. "Least-squares estimation of transformation parameters between two point patterns". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, pp. 376–380.

Wiedemeyer, T., 2015. "Iai kinect2: Tools for using the kinect one (kinect v2) in ros". https://github.com/code-iai/iai_kinect2.

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.