International Journal
of Applied Information Technology
http://journals.telkomuniversity.ac.id/ijait/

Survey of SLAM in Low-Resource Hardware

Ismail

School of Applied Science, Telkom University, Indonesia
ismailrusli@telkomuniversity.ac.id

ARTICLE INFO

Received 04 June 2018
Revised 12 October 2018
Accepted 22 October 2018
Available online 26 October 2019

Keywords
SLAM, embedded system, low-resourced hardware

ABSTRACT

Many of researches in Simultaneous Localization and Mapping (SLAM) are targeting desktops or laptop computers. Mounted in a robot platform such as Pioneer, these high computational power hardware do all the processing in SLAM. Still others, SLAM algorithms exploit GPU power to provide deep details in map reconstruction. Yet, it is desirable to deploy SLAM in a small robot without advantages from high computational power hardware. Single board computers with limited power supply and low computing power are the main boards that are often available in small robots. Therefore, it is important to consider the design solution of SLAM that targets such a system. This research presents a survey paper of SLAM in low-resource hardware. The main question to be answered is "How researchers deal with hardware limitation when implementing SLAM?" Classification based on a method to tackle the problem is presented as the conclusion of this paper.

* Corresponding author at:
School of Applied Science, Telkom University,
Jl. Telekomunikasi No. 1, Terusan Buah Batu, Bandung, 40257, Indonesia.
E-mail address: ismail@tass.telkomuniverity.ac.id

ORCID ID:
Author: 0000-0002-2714-1810

https://doi.org/10.25124/ijait.v3i01.1307
Paper_reg_number IJAIT000030101 2019

1. Introduction

In order to navigate autonomously, a mobile robot should have knowledge of its surrounding environment. Otherwise, it must have the capability to develop a model of the environment while localizing itself simultaneously. This problem is known as Simultaneous Localization and Mapping (SLAM).

Theoretically, SLAM is considered solved. However, several important issues have to be considered [1]:

1. Can there be dynamic objects?
2. Can we not only ignore them but include them in the model in some useful way?
3. Can it be done in real time for very large environments?
4. Can we monitor and update changes in the environment?
5. Can we compute semantically meaningful models, not just geometric models?

Typically, new SLAM algorithms run in mid to high processing power computer, i.e., desktop or laptop. Some even use GPGPU to satisfy high processing demand. In other hands, SLAM can be implemented in low-resource hardware (low-clocked processor, megabytes of RAM, and low-cost sensors) that is desirable for consumer products such as a robot to mop or sweep the floor in an indoor environment.

There are only a small number of works that are targeting SLAM in low-resource hardware specifically. This is reflected in a number of links returned from Google Scholar search with the keyword: "intitle:+slam low resourced hardware low cost". After manual selection, this paper reviewed 16 from those papers which work specifically on SLAM in low-resource hardware.

This paper continues as follows. Section 2 describes the mathematical framework of SLAM. Section 3 contains short descriptions of works in SLAM targeting constrained hardware. Finally, the conclusion is described in Section 4.

2. Formulation of SLAM

Mathematically, SLAM is formulated as a problem to calculate following joint probability distribution:

$$p(x_{0:k}, m | z_{1:k}, u_{1:k}) \tag{1}$$

With $x_{0:k} = \{x_i\}_{i=0}^{k}$ is the history of robot pose. m is the representation of the environment inform of a map. Then, $z_{1:k} = \{z_i\}_{i=1}^{k}$ is a history of measurements, and $u_{1:k} = \{u\}_{i=1}^{k}$ is a history of control given to the robot.

Probability distribution in (1) estimates the robot's pose (position and orientation) in each time step while simultaneously estimates a map of the robot's environment. By considering the Dynamic Bayesian Network in Figure 1, the probability distribution could be factorized [2]

$$p(x_{0:k}, m | z_{1:k}, u_{1:k}) \propto p(x_0) \prod_{i=1}^{k} p(x_i | x_{i-1}, u_i) \prod_{i=1}^{k} p(z_i | x_i, m_{O(i)}) \tag{2}$$

With $m_{O(i)}$ represents the set of all map elements observed at *i*-th time step.

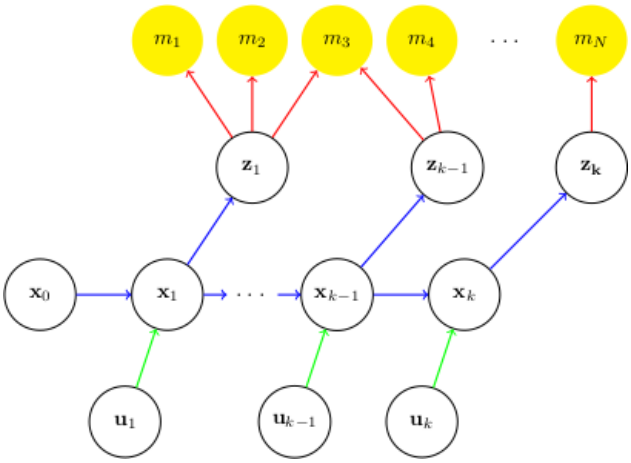

Figure 1 Dynamic Bayesian Network for SLAM Problem

We could filter the problem by estimating only the current robot's pose instead of the full path of the robot. This filtering problem is formulated in (3).

$$p(x_k, m \mid z_{1:k}, u_{1:k}, x_0) \tag{3}$$

Assuming the Markovian process and using Bayesian theorem, (3) could be formulated in a recursive form, i.e. Recursive Bayesian Filter

$$p(x_k, m|z_{1:k}, u_{1:k}) \propto p(z_k|x_k, m) \int_{x_{k-1}} p(x_k|x_{k-1}, u_k) p(x_{k-1}, m|z_{1:k-1}, u_{1:k-1}) dx_{k-1} \tag{4}$$

with $p(z_k|x_k, m)$ is sensor model and $p(x_k|x_{k-1}, u_k)$ is the robot's motion model.

In general, there is no analytical solution to (4) due to arbitrary form of distribution inside an integral part. However, by assuming that the form of $p(x_{k-1}, m|z_{1:k-1}, u_{1:k-1})$ is Gaussian and the system is linear (or approximated as linear), the problem could be solved analytically. This is Kalman Filter (Extended Kalman Filter). Relaxing the assumptions, we could get the sub-optimal solution of SLAM by numerical methods [3][4].

Typically, EKF-based SLAM has $O(L^3)$ complexity [2] with L denotes the number of landmarks. This is undesirable since the requirement of real-time then will not be achievable in a large-scale environment.

FastSLAM or SLAM with particle filter performs better in term of time complexity. According to [2], SLAM with Rao-Blackwellized Particle Filter (RBPF) has $O(LN)$, with L is the number of map elements, and N is the number of the particle. It seems that particle filter is a candidate to choose if one wants to implement SLAM in low-resource hardware. However, SLAM based on particle filter also has its own drawback, because ideally, the number of the particles should be exponentially increased if the map is getting larger. This makes particle filter faces a similar problem with EKF-based SLAM.

Bayesian filtering method was state-of-the-art in the early years of SLAM researches. However, as camera getting popular in robotics, visual SLAM attracted many researchers and they found a way to use techniques such as Bundle Adjustment to attack SLAM problems. This technique, for example, was popularized by works of Klein [5]. Recently, the availability of RGB-D camera, also introduced novel dense technique to SLAM, dubbed Dense SLAM [6][7].

3. SLAM in Low-Resource Hardware

Table 1 shows hardware usages for several most-cited SLAM algorithms. The table shows that SLAM algorithms are designed with no consideration of optimization for low-resource hardware (except for LSD-SLAM which has been implemented in smartphone although works only for visual odometry).

Table 1 Hardware Usage in SLAM with Relevant Notes on Time and Space Complexities

Paper	Hardware	Note
MonoSLAM EKF [8]	1.6GHz Pentium M Processor	Total processing of each frame is 19ms
FastSLAM[3][4]	1GHz Pentium PC	With 100 particles and 1M landmarks, the memory needed to store map is about hundreds of megabytes [4]
PTAM[5]	Intel Core 2 Duo 2.66GHz	Used in a small workspace for AR application, the algorithm is able to operate real-time in a smartphone (2.3GHz quad-core CPU).
KinectFusion[9]	Kinect sensor and commodity GPGPU	The time needed to reconstruct the environment of the volume $3m^3$ with 512^3 voxels resolution is about 25ms
LSD-SLAM[7]	CPU	Real-time. The visual odometry algorithm is optimized for NEON architecture [10]
ORB-SLAM[11]	Intel Core-i7-4700 MQ (4 cores @2.4GHz) with 8GB RAM	In average, for New College Dataset, ORB-SLAM takes 31.60ms for tracking and 464.27ms for local mapping

The following paragraphs review the papers in SLAM for low-resource hardware. Table 2 shows the summary.

Abrate, et. al. [12] experimented with an IR-only sensor to equip a mobile mini robot with EKF-SLAM. There was no optimization in the algorithm and the results showed that the sparse and noisy nature of the data acquired from the IR sensor limit the success of SLAM.

Bonato, et. al [13] implemented the EKF-SLAM that was targeting an embedded system based on an FPGA (Field-Programmable Gate Array) device. This work was not aimed at designing a new algorithm. After profiling EKF-SLAM, the authors implemented the most time-consuming part of the algorithm in hardware by using C2H (C-to-Hardware) module or design custom instruction set attached directly to ALU of the main microprocessor (softcore processor). The result is still considerable poor compare to that of standard PC work on 3 GHz clock CPU (The experiment used Altera NIOS II microprocessor with the clock of 50 MHz).

Beevers, et. al. [14] tried to cut the computational cost of SLAM by implementing a particle filter with fixed-point constraint. As Abrate, et. al. [12], they were targeting small mini consumer-oriented robot. With infrared rangefinders and microcontroller as the main processor, the result showed that the capability of the robot to close to the loop in the area of 2m ×2.5m. However, the SLAM would take as long as half a minute for every step of prediction and update which is far from real-time capability.

Gifford et. al [15] used multi-robot to handle area larger than a small robot with SLAM could handle. The author claimed that each of their robot cost as low as $1250. The author made the scenario of planetary exploration by using the multi small low-cost robot. Several sensors were employed obviously to reach the

goal of making a low-cost robot. For example, instead of using a costly rangefinder, the author used 6 IR sensors separated 30^o to cover 180^oFOV. To tackle uneven terrain typical in the planetary surface, the robots also were equipped by gyroscope and accelerometer in addition to odometry to track robot's pose. All those sensors provide data to DP-SLAM algorithm [16] [17]. With GumstixVerdex XL6P (600 MHz, 128 MB RAM, 32 MB Flash) as the main processor, the system was able to process every timestep in 3 seconds and 10 seconds for 15 and 25 particles respectively.

The success of SLAM usually relies on accurate sensors. In contrast to that, Yap [18] considered the low-cost and noisy sensors, i.e. sonar, to build SLAM. Although this research was not about the usage of low computational power, using low-cost sonar is in line with our aim to provide the survey in an attempt to make SLAM accessible in low-resource hardware.

Based on particle filter, the research resulted in accurate tracking and mapping task. One main assumption was orthogonality of the shape of the environment.

Four types of test were done in 3 test environments, i.e. tracking with odometry, tracking and mapping without the assumption of orthogonality, and using occupancy grid map instead of the particle filter.

Eade [19] proposed techniques for bounding the SLAM graph complexity during operation, using variable elimination and constraint pruning with heuristic schedules. The system consisted of a graph SLAM optimization engine in the back end and the view creation/recognition engine at the front end. As time goes by, the created view grows larger and burden both the storage of the system and the recognition process if the robot visited the previous place. The key ideas of the author were to bind this growth in the graph by removing nodes and limiting connectivity between nodes. The nodes were removed by doing marginalization while limiting connectivity was done by queuing priority nodes with degrees (number of edges) exceeding a predetermined value and pruning the edge from the queued nodes until no node degrees exceed the bound. The author did not show the performance of the system presented in the form of how fast the algorithm works in the robot platform. Instead, they showed how the proposed key ideas could reduce graph complexity while keep maintaining the accuracy of SLAM.

Vinckeet. al. [20] presented a solution to the implementation of SLAM in an embedded system by co-design hardware architecture, feature detector, SLAM algorithm, and optimization methodology. The SLAM algorithm was used based on EKF-SLAM. The optimization was done by trying several feature detectors and use the best one. The author developed system architecture with different processors (RISC processor, DSP, GPU) and a co-processor for data pre-processing. The conclusion of the research is that FAST feature detector surpasses Shitomasi and Harris corner detector in term of performance.

Magnenat [21] proposed a solution to SLAM problem for affordable hardware by co-design a slim rotating distance scanner, a lightweight SLAM software, and an optimization technology. The SLAM was based on FastSLAM 2.0 that run in real-time on a miniature robot.

Spampinatoet. al. [22] used FPGA to create an embedded stereo vision module for 6D pose estimation and mapping. The stereo vision system was made of two 5-megapixels CMOS digital image sensors from Micron (MT9P031) and a Spartan-3A-DSP FPGA with 1800K system gates, 84 block RAM (18KB each) and 84 DSP 48A blocks.

SLAM algorithm used was based on EKF. With the ability to estimate pose in 6DoF, and the system was able to operate in unlevel terrain.

Buonocore, et. al. [23] used a particle filter with sensor data fusion algorithm. He used a web camera with laser-pointer, an infrared sensor, and an ultrasonic sensor. The robot is equipped with autonomously exploration algorithm that makes the path that was taken by the robot follows a certain rule. The experiment was conducted in an indoor area with the size of 8.8m×2.8m. The results were comparable to the robot without the automatic exploration algorithm.

Gutmannet. al. [24] presented the method of localization based on the spatial variation of continuous signals coming from beacons. The signal was modeled by using SLAM, which the author has experimented with 3 algorithms, i.e. EKF-SLAM, graphSLAM, and Exactly Sparse Extended Information Filter SLAM (ESEIF-SLAM). The proposed method was evaluated on an embedded on an ARM 7 embedded board with 64 KB RAM connected to Roomba 510 vacuum cleaner.

Although the processing power was frequently limited, several single board computers were equipped with the SIMD architecture. It was desirable for SLAM to be able to get benefit from this architecture. Vinckeet. al. [25] presented the efficient EKF-SLAM algorithm in multi-core embedded system. They evaluated the algorithm by partitioning it into several functional blocks (FB), then determined each processing time by integrated cycle counter register (CCNT) of the ARM processor. The FB with high processing time then reimplemented into the multi-core architecture. The author exploited the SIMD architecture in ARM processor (NEON) and its DSP-module. The author showed that the matching and estimation tasks that have been executed within NEON environment gained significant performance boosting against similar FB executed in ARM processor.

Lee and Lee [26] used the upward looking camera for visual SLAM along with odometry. This configuration was adopted in ARM11 hardware targeting SLAM in low-cost consumer robots. Constructed for indoor environment, the pose graph optimization was used as it was known as a successful method in SLAM for the large-scale environment. However, with the proposed method, called visual compass, the author showed that the pose graph optimization complexity has been reduced to a significant amount makes it suitable for the embedded system implementation.

Tripicchio [27] used modified FastSLAM 2.0 [4] to build SLAMCGS, an algorithm targeting SLAM in low-resource hardware. They used laser rangefinders and IMU as sensors in Micro Aerial Vehicle (MAV) making a survey of the indoor environment. The results showed that their algorithm was robust against sensor noise and the delay between signals acquired was comparable to that of FastSLAM 2.0 in terms of execution speed and CPU load.

Nikolicet. al. [28] equipped their MAV with 4 cameras as well as IMU sensor to get robust and accurate pose-estimation and mapping. The real-time performance achieved by mitigating the most time-consuming part of the algorithm, i.e. image processing, to FPGA hardware.

Dine et. al. [29] presented a temporal analysis in graph-based SLAM in OMAP embedded architecture. The authors compared they work to known graph optimization framework, i.e. g2o [30]. The key to achieving efficacy in the algorithm was reached by the optimized data structure and efficient memory access management. Implemented in multi-core architecture, the proposed optimized algorithm was comparable and even perform better than the g2o framework.

Table 2 SLAM in Low-Resourced Hardware

Paper	Hardware	Sensors	Description
[12]	Khepera II	IR	Achieved limited success in SLAM
[13]	FPGA (main microprocessor is Altera NIOS II 50MHz)	1-4 CMOS Camera Modul	Implementing the EKF-SLAM algorithm in hardware and in the custom instruction set
[14]	Ratbot (ATMega64 88-bit 16MHz)	Infrared rangefinders	Success in detecting loop closures. SLAM takes half a minute each step.
[15]	GumstixVerdex XL6P (600MHz, 128 MB RAM, 32MB Flash)	6 Infra Reds, 3-axis accelerometer, and 3-axis gyroscope	Implemented in the multi-robot system in the scenario to explore a planetary surface
[18]	IBM ThinkPad X32	16 sonars	Able to track and map accurately with orthogonality of the environment's shape assumption
[19]	N/A	Web camera, wheel odometry	Reducing graph complexity while maintaining its accuracy
[20]	GPU, ARM Cortex A8, DSP	Inertial Measurement Unit, odometry, camera	Using EKF-SLAM with different feature detectors
[21]	ARM 11 533MHz	Rotating scanner (infrared)	-
[22]	FPGA	CMOS Camera	EKF-SLAM and able to operate in unlevel terrain
[23]	PC	IR, webcam, sonar	Low-cost sensors with a data fusion algorithm
[24]	ARM 7, 64KB RAM	Norhstar (optical sensing)	Using active beacons to do localization
[25]	ARM Cortex-A 500MHz, NEON, DSP 64x processor, 3D graphics accelerometer	Camera	EKF-SLAM. The result of implementation in multi-core is compared to a single core.
[26]	ARM11	Looking upward camera	graphSLAM with a visual compass
[27]	OMAP4430, dual-core 1GHz ARM Cortex-A9	Laser rangefinders	Comparable in speed to that of FastSLAM 2.0
[28]	XILINX FPGA	Camera, IMU	To cut complexity, image processing is processed using FPGA hardware
[29]	OMAP4430 (Dual core ARM Cortex-A9)	N/A	GraphSLAM

4. Conclusions

Survey of several papers for SLAM in low-resourced hardware have been presented. In general, those works could be categorized into two broad categories, i.e., hardware and software optimization.

1. Hardware optimization
   a. Customizing system architecture (which present in [13] [22] [28]).
   b. Using low-cost low accuracy sensors ([21] [15] [18] [14] [23] [12]).
2. Software optimization
   a. Reducing complexity ([19]).
   b. Design new algorithm ([24] [26]).
   c. Tweaking software part ([25] [20] [29] [27]).

All the works surveyed here have succeeded in part of solving the problem of implementing SLAM in low-resource hardware. Most of the works run in small environments and producing a sparse map that is limited in usage. Following the trend in SLAM which is using RGBD sensors to produce a high-quality dense map, which is more suitable for robotic applications, is still a great challenge. However, the increase of computing power owned by single-board computer opens an avenue to fully implement SLAM in a mobile robot with limited resources.

Bibliography

[1] Udo Frese, "Interview: Is SLAM Solved?," *KI - Kunstliche Intelligenz*, vol. 24, pp. 255-257, 2010.

[2] Juan-Antonio Fernandez-Madrigal, Simultaneous localization and mapping for mobile robots: introduction and methods.: IGI Global, 2012.

[3] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, and others, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," 2002.

[4] Michael Montemerlo and Sebastian Thrun, FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics.: Springer, 2007, vol. 27.

[5] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*.

[6] Christian Kerl, Jurgen Sturm, and Daniel Cremers, "Dense visual SLAM for RGB-D cameras," in Intelligent Robots and Systems (IROS), *2013 IEEE/RSJ International Conference on, 2013*, pp. 2100-2106.

[7] Jakob Engel, Thomas Schops, and Daniel Cremers, "LSD-SLAM: Large-scale Direct Monocular SLAM," in *Computer Vision--ECCV 2014*.: Springer, 2014, pp. 834-849.

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29*, pp. 1052-1067, 2007.

[9] Richard A. Newcombe et al., "KinectFusion: Real-time dense surface mapping and tracking," in 2011 10th

[10] T. Schops, J. Engel, and D. Cremers, "Semi-dense visual odometry for AR on a smartphone," in *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*

[11] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos, "ORB-SLAM: a Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, *vol. 31*, pp. 1147-1163, 2015, 00025 arXiv: 1502.00956.

[12] Fabrizio Abrate, Basilio Bona, and Marina Indri, "Experimental EKF-based SLAM for Mini-Rovers with IR Sensors Only.," in *3rd European Conference on Mobile Robots*, 2007.

[13] Vanderlei Bonato et al., "An FPGA implementation for a kalman filter with application to mobile robotics," in *2007 International Symposium on Industrial Embedded Systems*, pp. 148-155.

[14] Kristopher R. Beevers and Wesley H. Huang, "An Embedded Implementation of SLAM with Sparse Sensing," in *IEEE International Conference on Robotics \& Automation (ICRA 2008)*, 2008.

[15] Christopher M. Gifford et al., "Low-cost multi-robot exploration and mapping," in *2008 IEEE International Conference on Technologies for Practical Robot Applications*, pp. 74-79.

[16] Austin Eliazar and Ronald Parr, "DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks" in *2003 Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*.

[17] Austin Eliazar, Ronald Parr, and others, "DP-SLAM 2.0," in *IEEE International Conference on Robotics and Automation*, 2004.

[18] T. N. Yap and C. R. Shelton, "SLAM in large indoor environments with low-cost, noisy, and sparse sonars," in *2009 IEEE International Conference on Robotics and Automation*.

[19] Ethan Eade, Philip Fong, and Mario E. Munich, "Monocular graph SLAM with complexity reduction," *in Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 3017-3024.

[20] Bastien Vincke, Abdelhafid Elouardi, and Alain Lambert, "Design and evaluation of an embedded system based SLAM applications," in *System Integration (SII), 2010 IEEE/SICE International Symposium on*, 2010, pp. 224-229.

[21] Stephane Magnenat et al., "Affordable slam through the co-design of hardware and methodology," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 5395-5401.

[22] Giacomo Spampinato et al., "An embedded stereo vision module for 6D pose estimation and mapping," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011, pp. 1626-1631.

[23] Luciano Buonocore, Cairo Lucio Nascimento Junior, and Areolino Almeida Neto, "Solving the Indoor SLAM Problem for a Low-Cost Robot using Sensor Data Fusion and Autonomous Feature-based Exploration.," in *ICINCO (2)*, 2012, pp. 407-414.

[24] J.-S. Gutmann, E. Eade, P. Fong, and M. E. Munich, "Vector Field SLAM - Localization by Learning the Spatial Variation of Continuous Signals," *IEEE Transactions on Robotics, vol. 28*, pp. 650-667, 2012.

[25] B. Vincke, A. Elouardi, A. Lambert, and A. Merigot, "Efficient implementation of EKF-SLAM on a multi-core embedded system," *in IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*

[26] Seongsoo Lee and Sukhan Lee, "Embedded visual SLAM: Applications for low-cost consumer robots," Robotics & Automation Magazine, IEEE, vol. 20, pp. 83-95, 2013.

[27] Paolo Tripicchio, Matteo Unetti, Nicola Giordani, Carlo A. Avizzano, and Massimo Satler, "A lightweight slam algorithm for indoor autonomous navigation," in *Australasian Conference on Robotics and Automation, ACRA*, 2014.

[28] Janosch Nikolic et al., "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014, pp. 431-437.

[29] Abdelhamid Dine, Abdelhafid Elouardi, Bastien Vincke, and Samir Bouaziz, "Graph-based SLAM embedded implementation on low-cost architectures: A practical approach," in Robotics and Automation (ICRA), *2015 IEEE International Conference on*, 2015, pp. 4612-4619.

[30] Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard, "g2o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 3607-3613.