Evaluation of Bio-inspired SLAM algorithm based on a Heterogeneous System CPU-GPU

Rachid Latif[1], *Kaoutar Dahmane*[1], *Monir Amraoui*[1], *Amine Saddik*[1], and *Abdelouahed Elouardi*[2]

[1]LISTI, ENSA Ibn Zohr University Agadir, 80000, Morocco
[2]SATIE, Digiteo Labs, Paris-Sud University, Paris Saclay University, Orsay, France

Abstract. Localization and mapping are a real problem in robotics which has led the robotics community to propose solutions for this problem... Among the competitive axes of mobile robotics there is the autonomous navigation based on simultaneous localization and mapping (SLAM) algorithms: in order to have the capacity to track the localization and the cartography of robots, that give the machines the power to move in an autonomous environment. In this work we propose an implementation of the bio-inspired SLAM algorithm RatSLAM based on a heterogeneous system type CPU-GPU. The evaluation of the algorithm showed that with C/C++ we have an executing time of 170.611 ms with a processing of 5 frames/s and for the implementation on a heterogeneous system we used CUDA as language with an execution time of 160.43 ms.

Keywords.SLAM, RatSLAM, Heterogeneous system, CPU-GPU, C/C++, CUDA.

1 Introduction

Embedded systems have the advantage of small size and low power, are alsopopularlyused in mobile robots to decreaseweight and increase endurance [1]. In the promisingfield of mobile robotics, localization has been a hot topic in recentyears. For navigation thatisdivided in three phases: mapping, localization, and planning, the robots needmaps, and in the same time they have to build a map. The questions that arise are: in the absence of a map, how to locateoneself, and withoutknowingone's position, how to build a map? That meansthat a robot needs to know its position to be able to map an environmentaccurately [2]. Nevertheless, it must absolutely have a preestablished map of itsenvironment to be able to locateitselfthere. Location technologies depend on the environment and cost, accuracy, frequency and robustness, which can beobtained by the absolute and relative positioningmethodssuch as global positioning system (GPS), inertialmeasurement unit (IMU) and wireless signal [3]. However, GPS technology can onlyworkoutdoors ,it does not limit the location error for indoor use. and the IMU system has a cumulative errorisused to measurelinear and rotationalacceleration of robots,WiFilocalization uses a WiFicardbased on a graph by collecting signal strength in the field. In thismethod, the mean and standard deviations of WiFi RSSI observations are approximated by linear interpolation on a graph [4][5]. Eachmethod assumes major limitations,underthiseffect, we are pushed to use algorithmsthatallow robots to map theirenvironmentswhile locating in the generated map. SLAM allows the robot to position itself by aligning the data collected by the sensorswith the data thatisalreadyavailable. The sensor data collectedwith the data already collected allows the construction of a navigation map representing a set of distinctive points in the environment, otherwiseknown as points of interest in order to have an orientation and planning of the trajectorywhilelimitingthe errorthat can be made by the robot generating the diversion.The process of solving the problembeginswithodometry techniques. Odometryis the measurement of the robot'sability to estimateitsown position. This isnormallycalculated by the robot from the position of itswheels. One of the key elements of the SLAM process is the acquisition of data about the robot environment. A robot will use differentreference points for differentenvironments. The reference points must bestationary, and the waypoints must be unique in relation to the surroundingenvironment. The waypoints must alsobenumerous and must be able to beseenfrommanydifferent angles. By extracting the sensory input and identifying the differentwaypoints, once a robot has detected a waypoint, it can thendetermineitsown position. A method must be in place for the robot to do this. This landmark extraction can beperformed in differentways, fromalgorithmssuch as peak extraction to scan matching. The important factor to rememberisthat the robot needs a method to identify a waypoint. Robots can also use previouslyscannedlandmark data and match them to eachother to determinetheir location [6].

In this context, Abouzahir et al, 2017 to ensure real-time performance of SLAM algorithms with their computational complexity have been executed on high-performance machines. The use of embedded systems is necessary to have an architecture that allows efficient implementation to ensure real-time constraints. There are attempts to implement SLAM algorithms on embedded systems. However, the implementation of SLAM algorithms still limited and strongly depends on the nature of the algorithm and the purpose of the embedded architecture [7]. The authors in Ma et al, 2016 implement large scale SLAM system that combines dense stereo vision with inertial tracking of using off on a high-end NVidia TITAN GPU and an

Intel i7 quad- CPU desktop [8]. Also, the authors in [7] provided a case study of the FastSLAM 2.0 algorithm dedicated to largescale environments implemented in different embedded architectures such as the Tegra X1 system-on-chip (SoC) which integrates processor 4x ARM Cortex A57 and 4x ARM Cortex A53 CPUs @ 1.9 GHz, also SLAM algorithmimplemented on a high-performance desktop Core 2 Quad Q6600, @ 2.40 GHz and on the T4300 dual-core, @ 2.10 GHz laptop computer also the algorithm is implemented on the ODROID-XU4 which uses a Quad-Core ARM Cortex15, and on The Panda board, ES includes an ARM Dual Core Cortex A9 processor @ 1 GHz [7].

In recentwork, NGUYEN et al, 2018 proposes a vision system implementing a SLAM algorithm on a heterogeneous architecture. The HOOFR-SLAM algorithm uses images acquired by a stereo camera to performsimultaneouslocalization and mapping, the implementationwasbased on a CPU-GPU architecture using CUDA and OpenCL. The embedded platforms used are JETSON Tegra X1 equippedwith 4- Core ARM and A57 4-Core ARM A53 @ 1.3-1.9 GHz and Intel core i7 laptop @ 3.40 GHz [9].Recently, the authors in [10] based on HOOFR extractor , designed a wholefeature extraction system, dedicated for SLAM application takingintoaccount the bucketingmethod. also, theyprovided A hardware-software codesignapproach in order to implement the system on FPGA-basedheterogeneous architecture usingOpenCLprogramming .and using a publiclydatasetthey can reach the performance of evaluation of FPGA-basedimplementation versus embedded GPU-basedimplementation.

Our workaims to achieve an implementation of a bio-inspired SLAM algorithm in a heterogeneous CPU-GPU system for the mobile robot, whichis able to perform real-time localization and mapping. The evaluation of the algorithmisperformed in laptop thatcontain the NVIDIA GeForce 940MX and Intel Core i7 @ 2.70 GHz. The results of the evaluationshowed us that the processing time with C/C++ on robot operating system (ROS) is 160.43 ms using CUDA, evaluated on the New College data set recorded by a stereo camera.

The presentedworkiscomposed of 4 parts. The first section for the introduction, the second part provides an overview on SLAM algorithmembedded for mobile ro- botic. The third part isdevoted to present the evaluation and resultobtained by the use of embedded CPU-GPU system and the last section isconsecrate to the conclusion.

2 SLAM: overview and approach

In roboticsfieldwe have a diversity of sensorsused for SLAM algorithmtrackits mission theirdifferentcapabilities and theirweak points push us to develop new algorithms. In recentwork, Latif et al. 2019, statedthat SLAM algorithms are classifiedaccording to the type of sensors and the nature of the mathematicalapproachused. He alsoprovided a study on someproposedmethods to solve the SLAM problem. The first proposed solution to solve the problem of localization and simultaneous mapping is Extended Kalman Filter (EKF) in [11], EKF presents an extension of the Kalman filterthattakes non-linearsystemsintoaccount. The advantage of thisalgorithmisgiving the uncertainty on the position of the robot also the landmarks in the course of time. ConcerningtheirDisadvantage, wefind the high algorithmiccomplexitypresented a lot of problems, mostlywhenweaim to achieve the real-time implementation, in another hand. This solution suffersfrom a consistencyproblem, such as another solution. FastSLAM [12] 00based on the particulatefilter and developedunder the name FastSLAM 2.0 [13-14], however, the FastSLAM has an advantagethathereducesalgorithmiccomplexitycompared to EKF-SLAM. The GraphSLAM, based on smoothingapproaches, using all the sensormeasurements can estimate in addition to the map, the full trajectory of the robot. The strong point of graph slam isthatallowsavoiding the propagation of linearizationerrors, despite GraphSLAM gives more preciseresults, itsalgorithmiccomplexityremainsrestrained due to the smoothingcharacteristic. We have also a totallyvisualalgorithmcalled the ORB SLAM whichpresent a monocular system based on SLAM characteristics for small and large, in door, and outdoorenvironments [15-16]. Stefano et al, 2019 presented the method to modify and customize the open source SLAM algorithm ORB-SLAM2 in order to run thisalgorithm in real-time using the NVIDIA Jetson TX2 board and theyadopted a data flow paradigm for images processing, achieving an efficient CPU-GPU load distribution, whichresults in a processing speed equal a 30 fps. The evaluation of resultsisbased on KITTI datasets [17]. The algorithmspresentedabove are probabilisticalgorithms and westill have the bio-inspiredalgorithmsthat focus on emulatingbiologicalsystemsthat are supposed to beresponsible for mapping and navigation in the animal and humanbrain, can also solve the SLAM problem. Rodents, in particular, are better at dealingwithnavigationalproblems: rats can navigateand updatetheir pose representationevenwithoutexternalsignals, using the estimation of self-movement, called path integration. Bio-inspired SLAM algorithmsinclude the RatSLAM provided by Milford et al. 2015 [18], based on the visual SLAM algorithm. It uses a simplified computer model of the rodent hippocampus to build a real-time map consistently and stablyusing a single camera. RatSLAM corrects cumulative errors in odometry by a map correction algorithm in internal and externalenvironmentsaccording to [19,20] expertise.

3 EVALUATION AND RESULT

In this work we using the bio-inspired RatSLAM inspired from hippocampus of rat, the fact that rodents are able to memorize the location of reference objects and the store as a virtual map [21]. The RatSLAM based on the visual SLAM algorithm, Visual SLAM refers to the process of calculating the position and orientation of a device with respect to its surroundings, while mapping the environment at the same time, using only visual inputs from a camera. Visual SLAM uses

only visual inputs to perform location and mapping, meaning that the only sensor required is a camera that has to be mounted on board of the device. No other external sensors are required [22].

The types of cameras are varied, Omnidirectional cameras are gaining in popularity: they have a 360° view of the environment and as features stay longer in the field of view, it is easier to find and track them. To improve the accuracy of features, some work relies on a multi-sensor system. The system of Castellanos et al consists of a 2D laser scanner and a camera, however, a monocular system has certain weaknesses in certain situations, for example, it requires additional calculations for depth estimates, scale propagation problems, or may lead to failure modes due to non-observability. Stereo systems are widely adopted in different environments, both for landmark detection and motion estimation in indoor and outdoor environments [23]. The identification of places is reached by using a neural network, RatSLAM able to generating topological representations of outdoor and indoor environments, the RatSLAM algorithm is composed of 4 blocks, local view, pose cell and the experience map block, and odometry, data that can be extracted from the bag file. We evaluated this algorithm on The New College Dataset [24] includes, laser, odometry, stereo camera images, panoramic images, and GPS recordings in a custom format. Data collection was performed outdoors on the 2.2km path. In order to run the dataset with OpenRatSLAM the panoramic images and odometric information have been re-encoded into a ROS bag file. Timestamps were extracted from the original dataset to ensure proper timing. The odometric information has been integrated to match the panoramic image rate of 3Hz [25], recorded by a robot using a stereo camera with resolution of 512 × 382 pixels, a stereo camera system consists of two cameras separated by a fixed distance which presents the simplest ways to directly get depth information; in the same method, that humans do with our eyes, observations of the position of the same 3D point in the two cameras provide the depth to be calculated by triangulation. It can reduce the constraint that depth information will be inaccessible without the cameras moving, as is the case with monocular cameras. Despite, the depth measurement range is limited by the baseline and resolution [5], with the version of the RatSLAM code Open Source. Open RatSLAM algorithm has provided by Ball. 2019 [26], using an open source, meta-operating system ROS system (Robot Operating System) is a set of self-service software that is a meta-operating system for robots. We can also represent it as a framework for writing robotic software. Its goal is to create a standardization of programming in robotics.

The operation of ROS is similar to that of a client-server. The master represents the server and the different nodes of the client. A node is an executable that can be, the data of a sensor. The master is the server to which all nodes must subscribe in order to be able to talk to each other. Once subscribed to the master, the nodes discuss with topics that are information transport services. The nodes can either publish information on these topics or read the information published in them or publish and read them at the same time. ROS offers a set of programs that allow the use of various sensors, visualization software, inter-machine connections, and simulation software. It can be used with several programming languages such as C++ [27]. ROS can ensure package management, also low-level device control given the necessary libraries, and allow message passing between processes, which the RatSLAM uses for communication between its blocks. Figure 1 shows the blocks that comprise the Open RatSLAM.

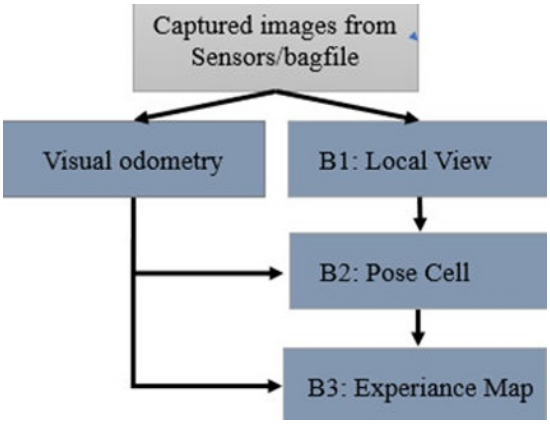

Fig. 1. Open RatSLAM blocks

Firstly, the images captured from sensors or provided by the dataset will be sent to the local view block to preprocesse the current image into a visual template representation in order to determine whether a scene given by the current view is a new or previously seen visual template by using image comparison techniques. The first step is converted image into a mono grayscale format. and the image then be cropped to bias the templates towards visually interesting areas of the camera images. Figure 2 shows the operation of matching the current view to all of the stored view templatesto bythe Local View node.

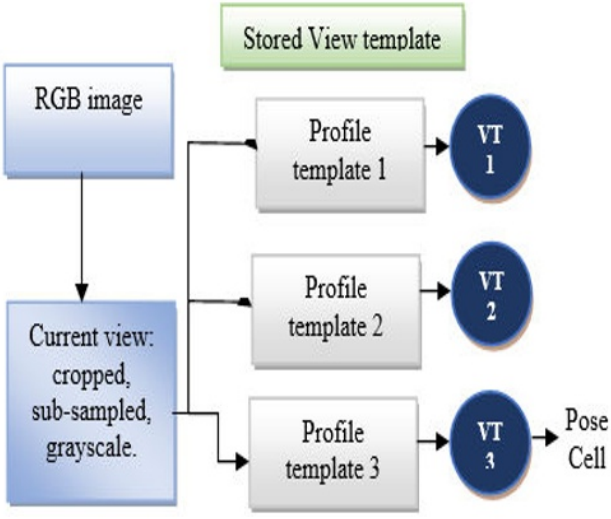

Fig. 2. Matching the current view to all of the stored view

The cropped region may then be subsampled to defined height and width parameters, also it may undergo global and local normalization steps which attempt to alleviate changes in illumination. Global normalization considers the mean and range of the entire image and addresses global changes in illumination. Local normalization preserves contrast in small patch regions. After pre-processing, the local view match node compares the visual template that represents the current camera image with all previously learned templates, the pose cell corresponding cell, if the view is new we add it to the templates stored previously, the block of pose cell networks of position, forms three dimensional hypotheses of location and orientation (x′,y′,θ′) for the position of the robot in the real environment (x, y, θ), responds to two types of input; odometry and view templates. The action on a view template input depends on whether this is a new or existing view template. For new view templates, the id is associated with the centroid of the current peak activity packet in the pose cell network. For existing view templates, activity is injected into the previously associated location in the pose cells. The injected activity for consecutive matches of the same view template decays rapidly but is gradually restored over time. this node manages the energy packet that represents pose in response to odometric and local view connections. In this implementation, this node handles the decision on when to create new nodes and links because it requires knowledge of the internal workings of the Pose Cell Network, which is no longer available due to the split into separate nodes. RatSLAM is based on the iterations of the CAN (Competitive Attractors Network) of pose cells block, the block called experience map is a topological representation encoding the pose cells and local view cells in nodes and links uses the received actions to create nodes and links, or to set the current node, each experience has an associated position and orientation. Creating a new node also creates a link to the previously active node. Experience map manages graph building, graph relaxation, and path planning [25]. Each of these 3 main blocks represents a process executing simultaneously, the first local view block is converted into CUDA language with the heterogeneous architecture CPU-GPU, our choice is based on the analysis of the code and extraction the execution times of each functions and methods. This block contains a function which we can implement on the GPU in order to be able to complement the CPU architecture by giving capacity to achieve repetitive calculation involving massive amounts of data. Heterogeneous System allows using more than one kind of processor, to work efficiently and cooperatively. The parallel programming language CUDA used on GPUs and CPUs is supported by the heterogeneous system. Compute Unified Device Architecture (CUDA) based on the standard C/C++ language represents a parallel programming paradigm allowing to use of GPU resources, CUDA is a proprietary framework created by NVIDIA, it generates better performance results.

CPU and GPU cores cooperate with each other . They assume that a parallel part of a code: CUDA cores run on a GPU and a serial part of a code; the rest of the C program runs on a CPU [28]. In CUDA applications, a crucial question is therefore how to structure a certain part of the code to expose so much data parallelism. We have observed that CUDA applications are generally designed to exploit massive parallelism only with GPUs. Figure 3 shows the proposed CUDA architecture CPU-GPU of the Open RatSLAM algorithm.

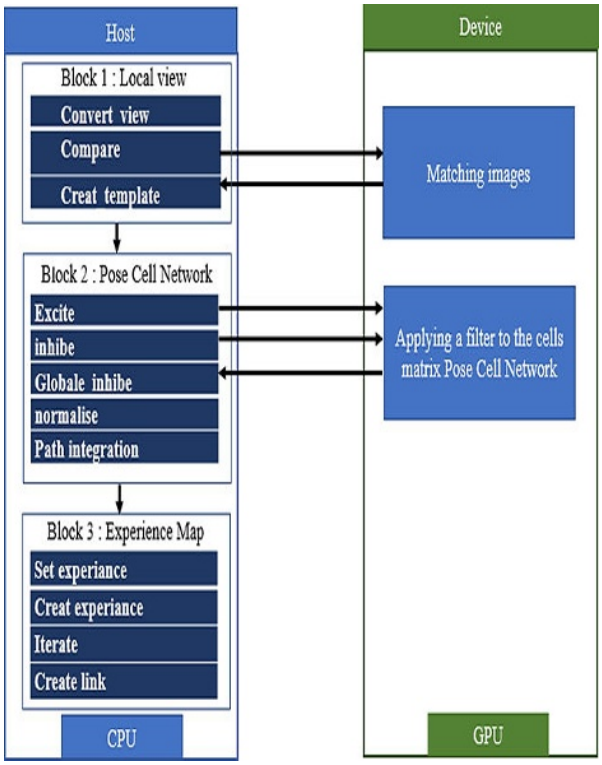

Fig. 3. Block diagram of the proposed CUDA architecture

When each new image is collected, the algorithm checks if the current view is identical or similar to the stored images models, model by model, so that it decides whether or not to add a new model to the vector of the different models, then the size of the visual template increases and the similarity calculation time also increases. In this work the calculation of the matching operation is performed in parallel so that the similarity is calculated for the images collected with all the templates stored at the same time. In the second block we apply a filter to the cells of the Pose Cells matrix to excite these cells, the multiplication of the matrix by the Pose cell matrix is carried out box by box, each box represents a laying cell then we have a large number of iterations for this operation for 204 image sequences there are 129925 iterations. Also the inhibit function responsible for inhibiting the activation of cells by the use of convolution matrix, the multiplication of the two cell-to-cell matrices: the original and mask is executed several times for 204 image sequences we have 1208636 iterations, then for both methods, we can apply the 3D multiplication (x, y, θ) in parallel by the GPU. At the end, the Experience

Map block creates the map based on the observations of the robot. The map is represented by a graph in which each node is an experiment. Table 1 shows our execution time, for the blocks on homogeneous CPU and heterogeneous CPU-GPU architectures.

Table 1.TOTAL EXECUTION TIME (MS)

Tools		***Laptop, CPU (C/C++)***	***Laptop, Nvidia GeForce 340MX CPU-GPU (CUDA)***
Functional blocks	Local View	170.611	160.43
	PoseCell Network	79.7	78.0
	Experience Map	54.3	-
Total time (ms)	304.611		292.73

As shown in the table above, we have 3 principal functional blocks, the first block named local view the use of C/C++ gave us a time of 170.611 ms and the use of the CUDA language gave us a time of 160.43 ms, so we can say that there is a change in the execution time. The pose cell block has a time of 79.7 ms with the C/C++ language which is already small and we have a time of 78.0 ms using the CUDA language then we don't have a big difference between the time resulting from the two times of this block because the time in C/C++ is already minimized. For the experience map block we have an execution time equal to 54.3 ms using the C /C++ language. We find that the average execution time of the global code of an image in CPU: 82.742032 ms and using the CUDA language we find 69.23 ms. Figure 4 presents the evolution of times with the use of GPU GeForce 940Mx to evaluate the time of 40 images.

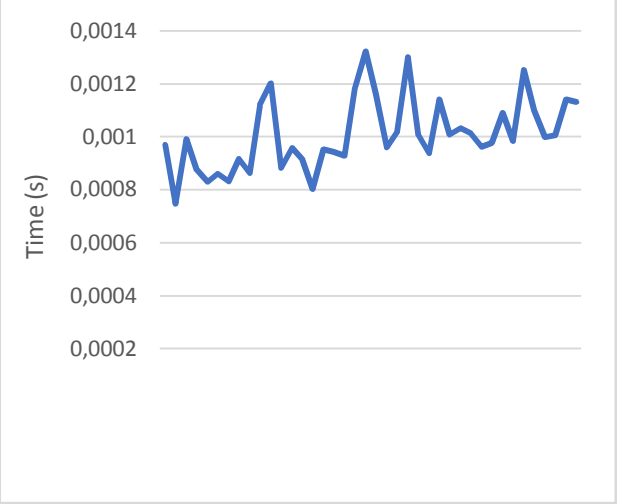

Fig. 4. The evaluation of 40 views of the function compare of the local view bloc with CUDA in GeForce 940Mx.

As it is shown in the figure 4, the execution time of the local view block compare function is minimized, which generates the reduction of the time of the block programmed in CUDA language in the Nvidia GeForce 940MX card of the laptop @ 1241 MHz, using also the Intel Core i7 CPU @ 2.70GHz The time variations are between 0.75 Ms and 1.32 Ms and which gives 1.01 ms in average.

4 CONCLUSION

Over the last decade, embedded systems have received particular attention because of their many applications. They can also be found in the field of robotics, so robots need a certain intelligence to work and make decisions, which is done with the help of embedded systems. The development engine for mobile robotics is the operation of embedded systems for the different implementation architectures made available. This work presents the implementation of the bio-inspired SLAM algorithm RatSLAM, on CPU-GPU architecture with using CUDA language. We obtained as a result a reduction of 10 ms, compared to the result with the homogeneous architecture CPU using (C/C++), which allows us to benefit from the technological progress, and allows us to respond more to the constraint of real-time. We aim the validation of the architecture in HIL (Hardware In the Loop), also we aim to use other heterogeneous embedded systems, so as to integrate its board in a robot in the future.

REFERENCES

1. R. A. Rebouças, Q. d. C. Eller, M. Habermann and E. H. Shiguemori, "Embedded System for Visual Odometry and Localization of Moving Objects in Images Acquired by Unmanned Aerial Vehicles," 2013 III Brazilian Symposium on Computing Systems Engineering, Niteroi, 2013, pp. 3540,
2. Huang, B.; Liu, J.; Sun, W.; Yang, F. A Robust Indoor Positioning Method based on Bluetooth Low Energy with Separate Channel Information. Sensors 2019, 19, 3487.
3. Jingbin Liu, Ruizhi Chen, Yuwei Chen, Ling Pei, and Liang Chen.iparking: An intelligent indoor location-based smartphone parking service. Sensors,2012.

4. Jingbin Liu, Ruizhi Chen, Ling Pei, Robert Guinness, and Heidi, Kuusniemi: A hybrid smartphone indoor positioning solution for mobile lbs. Sensors,2012.
5. Abby Yao, Teaching Robots Presence: What You Need to Know About SLAM,https://blog.cometlabs.io/teaching-robots-presence-what-you-need-to-know-about-slam-9bf0ca037553Gfgbgd , 2017 .
6. Rebecca Maxwell, Robotic Mapping: Simultaneous Localization and Mapping (SLAM), https://www.gislounge.com/robotic-mapping-simultaneous-localization-and-mapping-slam/ , 2013 .
7. Mohamed Abouzahir, AbdelhafidElouardi, Rachid Latif, Samir Bouaziz, and AbdelouahedTajer. Embedding slam algorithms: Has it come of age? Robotics and Autonomous Systems, 2017.
8. Ma, L., Falquez, J. M., McGuire, S., Sibley, G., 2016.Large scale dense visualinertial slam. In: Field and Service Robotics. Springer, pp. 141–155.
9. Dai-Duong Nguyen. A vision system based real-time SLAM applications. Hardware Architecture [cs.AR]. Université Paris-Saclay, 2018. English. NNT: 2018SACLS518ff. tel- 02398765.
10. Nguyen, D.-D., El Ouardi, A., Rodriguez, S., Bouaziz, S.: FPGA implementation of HOOFR bucketing extractor based real time embedded SLAM applications. Journal of Real-Time Image Processing ,2020, https://doi.org/10.1007/s11554-020-00986-9.
11. Smith. R, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics, Autonomous Robot Vehicles, pages 167–193. Springer Verlag, 1990.
12. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In: AAAI National Conference on Artificial Intelligence, Edmonton,Canada(2002).
13. M. Abouzahir, A. Elouardi, S. Bouaziz, R. Latif, A. Tajer. Large Scale Monocular FastSLAM2.0 Acceleration on an Embedded Heterogeneous Architecture EURASIP Jour- nal on Advances in Signal Processing, SpringerOpen, Juillet 2016.
14. Mohamed Abouzahir, Rachid Latif, AbdelouahedTajer, AbdelhafidElouardi, Samir Bouaziz, "Localization and Mapping algorithms implemented on a low-power embedded architecture: A case study", 5th International Conference on Multimedia Computing and Systems (ICMCS) Marrakech, 2016 IEEE Xplore digital library.
15. Mur-Artal, R., Montiel, J., Tardos, J. D., 2015. Orbslam: a versatile and accurate monocular slam system. Robotics, IEEE Transactions on 31 (5), 1147–1163. R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.
16. R. Latif and A. Saddik, "SLAM algorithms implementation in a UAV, based on a heterogeneous system: A survey," 2019 4th World Conference on Complex Systems (WCCS), Ouarzazate, Morocco, 2019, pp. 1-6, doi: 10.1109/ICoCS.2019.8930783.
17. S. Aldegheri, N. Bombieri, D. Daniele Bloisi and A. Farinelli, "Data Flow ORB-SLAM for Real-time Performance on Embedded GPU Boards", 2019 IEEE/RSJ International Confe- rence on Intelligent Robots and Systems (IROS), pp. 16, doi: 10.1109/IROS40897.2019.8967814.
18. Michael J Milford, Gordon F Wyeth et DF Rasser :Ratslam : a hippocampal model for simultaneous localization and mapping. In Robotics and Automation, 2004.Proceedings.ICRA'04. 2004 IEEE International Conference on, volume 1, pages 403–408. IEEE, 2004.
19. Michael J Milford and Gordon F Wyeth: Mapping a suburb with a single camera using a biologically inspired slam system. IEEE Transactions on Robotics, 24(5):1038–1053, 2008.
20. Arren J Glover, William P Maddern, Michael J Milford et Gordon F Wyeth: Fab- mapratslam : Appearance-based slam for multiple times of day. In Robotics and Automation (ICRA),2010 IEEE International Conference on, pages 3507–3512. IEEE, 2010.
21. Milford, Michael, Wyeth, Gordon, and Prasser, David (2004) RatSLAM: a hippocampal model for simultaneous localization and mapping. In Valavanis, K (Ed.) Proceedings of the 2004 IEEE International Conference on Robotics and Automation. IEEE, United States of America, pp. 403-408.
22. Dragonfly, How can visual SLAM be used and what are the applications? , https://dragonflycv.com/what-is-visual-slam/.
23. D. Scaradozzi1, S. Zingaretti1, A.Ferrari, Simultaneous localization and mapping (SLAM) robotics techniques: a possible application in surgery, http://shc.amegroups.com/article/view/4083/4890https://www.gislounge.com/robotic-mapping-simultaneous-localization-and-mapping-slam/ , 2018.
24. Mike Smith, Ian Baldwin, Winston Churchill, Rohan Paul, and Paul Newman. The new college vision and laser data set. The International Journal of Robotics Research, 28(5):595–599, 2009.
25. David Ball, Scott Heath, Janet Wiles, Gordon Wyeth, Peter Corke, Michael Milford: OpenRatSLAM: an open source brain based SLAM system, Autonomous Robots, 2013.
26. David BALL. Open RatSLAM from Internet: https://github.com/davidmball/ratslam.2019.
27. A. Ouadrhiri , Implémentation d'un SLAM Monoculaire pour un robot d'intérieure ,

https://www.ensta-bretagne.fr/jaulin/rapport_pfe_amine_ouadrhiri.pdf , 2018.

28. Nickolls, J., Dally, W.: The gpu computing era. Micro IEEE 30(2), 56–69 (2010)