REST Journal on Emerging trends in Modelling and Manufacturing
Vol:3(2),2017
REST Publisher
ISSN: 2455-4537

Website: www.restpublisher.com/journals/jemm

A critical comparison between Fast and Hector SLAM algorithms

Mustafa Eliwa, Ahmed Adham, Islam Sami and Mahmoud Eldeeb
Department of Aerospace Engineering
University of science and technology, Zewail, Egypt
s-ahmed.sayed@zewailcity.edu.eg

Abstract
This paper compares between two Simultaneous Localization and Mapping (SLAM) algorithms. SLAM algorithms are used by a mobile robot to build a map for unknown environment and localizing itself in the map related to some landmarks using odometry. In this paper, Robot Operating System (ROS) is used for implementing and simulating Hector SLAM algorithm using a package called Hector mapping and a Fast SLAM algorithm using a package called Gmapping. The criteria for evaluating the performance are time-consumed, the size of the built map, error between the built map and the actual one and the performance variation while the map is increased. Changing the parameters for building the map is also studied, and its influence on the time consumed and the quality of the built map also. Finally, the results of the two algorithms are compared to each other in order to determine which application does Hector SLAM has the upper hand in, and in which applications Fast SLAM is recommended.

I- Introduction:

SLAM is constructing a map using mobile robot (mapping), then determining the position of this robot relative to this map. First, data about the entire environment is collected. Secondly, the data (observation) is assigned to the landmarks of the environment. These landmarks are either artificial or natural. The advantages of artificial landmarks are that they are designed to be detected optimally. In addition, the artificial method is more reliable than the natural one. On the other hand, the detection of natural landmarks is more difficult, but they are more efficient for large maps. There should be three or more landmarks to manage us pose the robot. The error of landmark positioning is bounded as it doesn't depend on previous readings. The collected data is used also for building maps for the surroundings. In this task, the data is collected from different sensors, each one is oriented in a different direction. The view obtained from each sensor is called the sensor view, and then these views from each sensor are integrated to compose the local map (a map for a single location at a time). After that, the robot moves to another location and repeats the previous steps to get another local map for a new location. Finally, the local maps are integrated to obtain the global view. There are different kinds of maps. First, the metric (grid) map. In this map the environment is divided into square cells of the same size. The obstacles are defined with black cells and a free space is defined with a blank cell. This map requires relatively huge memory size and extensive computational time to generate it. The second kind is the topological map (graph-based), it is relatively better than the metric map in terms of computational time consumed and memory consumption but on the other hand, it is more difficult in constructions; also, the position of the robot cannot be located specifically on it. Therefore, it is not useful for SLAM. The third map is metric topological map; it has the same problem of the topological map. Therefore, the most suitable map for SLAM is the metric map.

II. Problem Formulation and modelling

The research done by K. Kamarudin et al presents a performance analysis between Hector SLAM and G-mapping algorithm. They used Microsoft Kinect instead of laser scan sensor, in addition, they proposed a new system integration via Linux to run those open source algorithm. Two different environments were used, the first is the empty office corridor without obstacles, and the second isthe office corridor with desk and chairs. The results show that the system managed to perform real time SLAM, in addition to giving a reliable method to evaluate SLAM in windows platform relative to ROS-based SLAM. Moreover, the modifications of the laser scanners' parameters improved the performance. The Kinect sensor limited field of view and range caused bad performance specially in open environments [1]. The second paper by K. Kamarudin et al introduced a method for merging data from Kinect sensor and LIDAR to compensate for the defects of the Kinetic sensor and improve the performance. Based on this approach the two SLAM algorithms, G-mapping and Hector SLAM, were tested. The results showed that there is a significant performance improvement after implementing this approach. Finally, the performance of the two algorithms was compared based on the size of the built map the computational time [2]. The problem basically is to map and localize the robot simultaneously (SLAM). This problem is introduced using two algorithms (fast slam and hector slam), and use the data to compare both algorithms. In each experiment, the robot was placed at a different position and orientation in the same map and controlled to move to any position using navigation stack package. Then a comparison between the two algorithms is performed to detect strengths and weaknesses of each of them. Then some package parameters are changed to study their effect on the performance of the package.

Proposed SLAM Algorithms:
This paper proposes two solutions for the SLAM problem: Hector SLAM and Gmapping which is algorithm based on fast SLAM.
Odometry:
The odometry is the method that is used to find the new position of the mobile robot based on the robot geometry, the velocities of right and left wheel and the initial position. The position of K-step of moving from the initial position is given by:

$$x(k) = 0.5[d_L(k) + d_R(k)]\cos(\theta(k)) + x(k-1)$$

$$x(k) = 0.5[d_L(k) + d_R(k)]\sin(\theta(k)) + x(k-1)$$

$$\theta(k) = \frac{1}{D}[d_R(k) - d_L(k)] + \theta(k-1)$$

Where d_L and d_R is the distance travelled by the lift and the right wheel. In addition, █ is the angle of motion relative to the assumed horizontal. These parameters are measured using encoders on each wheel. Although this method is widely used, it has many symmetric and non-symmetric sources of error. The symmetric errors result from unequal wheel diameter, the average actual wheel diameter and wheel base are different from the nominal ones and the misalignment of wheels. In addition, the non-symmetric errors come from bumps or cracks of the floor, unexpected objects on the floor moreover the wheel-slippage.

Hector SLAM [1]:
Hector SLAM is an open source algorithm used for building a 2D grid map for the surrounding environment based on laser scan sensor (LIDAR). This algorithm locates the position of the robot based on scan matching and doesn't use the odometry of the wheel which is the common method in other SLAM algorithms. The LIDAR manages it to perform the scan matching task to locate the robot fast and accurately due to its high update rate. Hector algorithm uses Gaussian-Newton minimization method witch is considered as update for Newton method, it has the advantages that second derivatives needn't to be computed. This method is used to find the optimum end point of laser scan relative to the build map by getting the transformation

$\zeta = (p_x, p_y, \psi)^T$ where:

$$\zeta^* = argmin\sum([1 - M(S_i(\zeta))]^2$$

Where M is the map and S_i is the coordinates of the laser end point. After that, the error could be expressed as

$$\sum_{i=1}^{n}[1 - M(s_i(\zeta + \Delta\zeta))]\text{^}2 \rightarrow 0$$

Then, $\Delta\zeta$ goes to zero and M is factorized using Taylor series, therefore

$$\Delta\zeta = \mathrm{H}^{-1}\sum_{i=1}^{n}\left[\Delta M(s_i(\zeta))\frac{\partial s_i(\zeta)}{\partial\zeta}\right]^T [1 - M(s_i(\zeta))]$$

Where

$$H = \left[\Delta M(s_i(\zeta))\frac{\partial s_i(\zeta)}{\partial\zeta}\right]\left[\Delta M(s_i(\zeta))\frac{\partial s_i(\zeta)}{\partial(\zeta)}\right]$$

However, this algorithm doesn't have closed loop, it closes its loop in the real world. The advantages of it are that it requires low energy to avoid the changes in the dynamic environments.
Fast SLAM:
The FastSLAM is based on using known landmarks in the environment where the robot will make map for. The algorithm are explained below. First, the function of the posterior can be factorized as

$$p(s^t, \theta \mid z^t, u^t, n^t) = p(s^t \mid z^t, u^t, n^t)\prod_k p(\theta_k \mid s^t, z^t, u^t, n^t)$$

Where s^t the robot's pose at time t, θ is the location of each landmark, while z^t is the robot's sensor measurement for each landmark, in addition, u^t is the robot control at time t and n^t is the index of each landmark. After that, Kalman Filter of dimension 2 is used to filter the landmarks from the particles around it. Finally, the landmark location is estimated.

Gmapping:

In the field of robotics, Gmapping algorithm is the most widely used package in the world. Gmapping algorithm is based on Rao Blackwellized Particle Filter approach, introduced for the first time by Murphy and Doucet et al. [3,4]. The main usage of RaoBlackwillized Particle Filter is to solve for metric (grid based) maps used in SLAM, where the inputs are the observation of the sensor $z_{1:t}= z_1, \dots ,_t$, and the odometry readings $u_{1:t-1} = u_1, \dots , u_{t-1}$. Where the outputs are the trajectory estimation of the robot $_{1:t}= x_1, \dots , x_t$, and the map$M$. The joint posterior then will be $(x_{1:t}, M \mid z_{1:t}, u_{1:t-1})$, but in order to simplify the posterior calculations two steps are done. Firstly, the estimated trajectory is calculated from the odometry readings and the observations. Secondly, after getting the estimated trajectory we can simply find $(M \mid x, z_{1:t})$. Finally, we can find the joint posterior using factorization, the posterior simply becomes: $(x1{:}t, M \mid z1{:}t, u1{:}t{-}1) = (M \mid x, z1{:}t)$. $(x1{:}t \mid z1{:}t, u1{:}t{-}1)$ Individual maps for all particle are built. The output of the algorithm is the map, this map is selected from the most probable posterior of each potential trajectory using the particle filter. Grisetti et al. [5,6] development was a big milestone for the Gmapping technique. In this new development, the newest observation and the previous odometry data contribute to the approximation of the modified posterior: $(x_1{:}t \mid M_t{-}1, x_t{-}1, z_1{:}t, u_1{:}t{-}1)$. Resampling operations for the particles is an adaptive technique in Gmapping. This step is critical to remove bad samples and in the same time keep good samples. Therefore, a measure of dispersion (***N***) is introduced to differentiate between readings based on its reliability and how well the posterior trajectory is approximated. The measure of dispersion of the importance weights is formulated to be:

$$N = \frac{1}{\sum(\alpha')^\wedge 2}$$

Where α' is the normalized weight of a certain particle.

A resampling should be performed each time measure of dispersion drops critically. This implementation reduces the risk of particle depletion.

Robot Configuration:

The used robot is CATBOT robot made by M. Abdelazim. This robot is a differential robot composed of two wheels at the back, each is equipped with DC-motor and encoder and caster wheel in front. The main sensor is laser scan sensor (LIDAR) scanning 360 degree. The figures one, two and three show the geometry and the configuration of the robot.

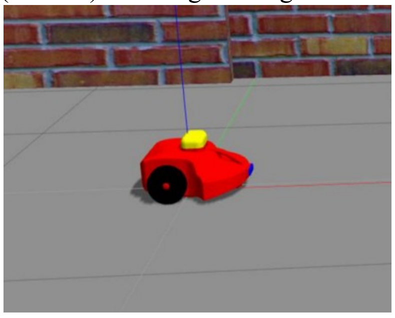


Figure 1

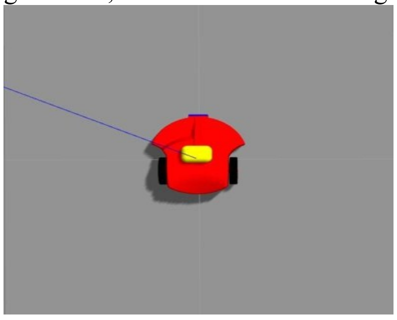


Figure 2

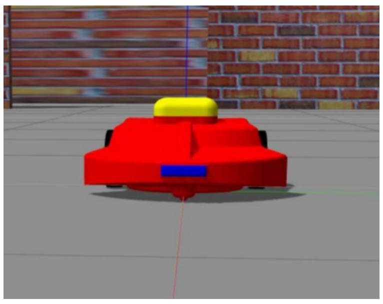
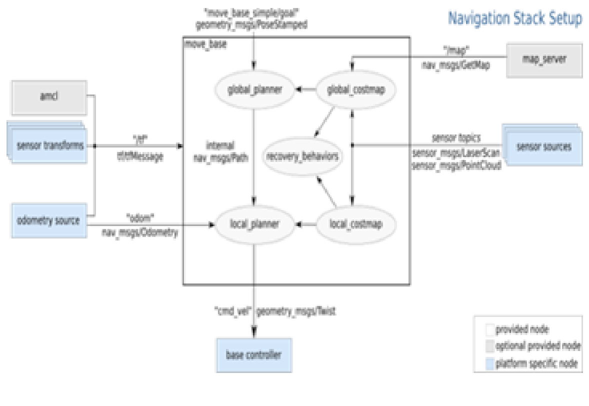


Figure 3

Navigation stack:

Navigation stack is a motion planning package. It is used to move the robot from a start position to a given goal position, without any collisions. The main use of this package in our project is to move the robot through the map easily. This package takes input from sensors, joint states, tf, and odometry, and generates the output as a Twist msg, which can be converted into velocity.



It moves the robot using a node called *move_base*.

V- Results:

Hector SLAM Results:

Hector-mappinghas been used to build a map. Firstly, the map has been built in gazebo and the robot (CatBot) has been placed in it. Then the robot is controlled in Rviz using a package called navigation stack. The robot takes the order to go to some place. It moves and during its movement, it builds the map that it is sensing with laser scanner. It also performs self-localization by odometry and determination of velocity of each wheel by an encoder. Through this paper two parameter are changed to test their influence on the generated map; map update distance threshold and map update angle threshold. Those two parameters indicate what distance or angle should the robot travel to update the map. The default values for those two parameters are 0.4 m and 0.9 rad respectively. In figure 4a and 4b the map built using the default parameters. And it also shows the map used through this paper.

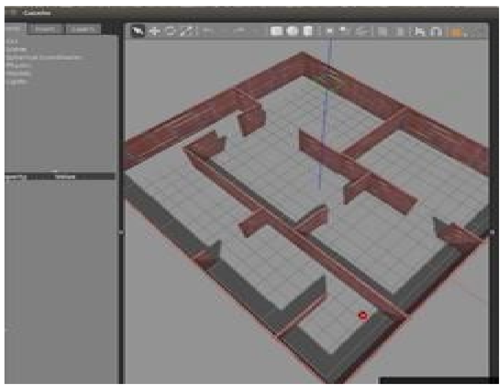


Figure 4a

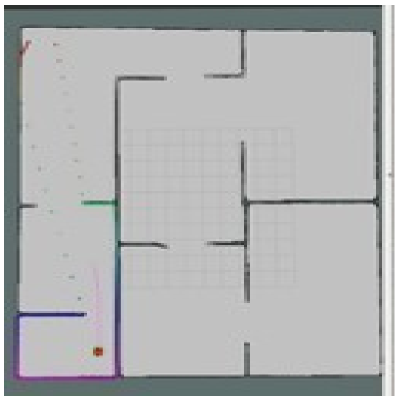


Figure 4b

Then the distance threshold parameter is changed to be 0.1 and the angle parameter also changed to be 0.1. Figure 5 shows the map generated using those parameters.

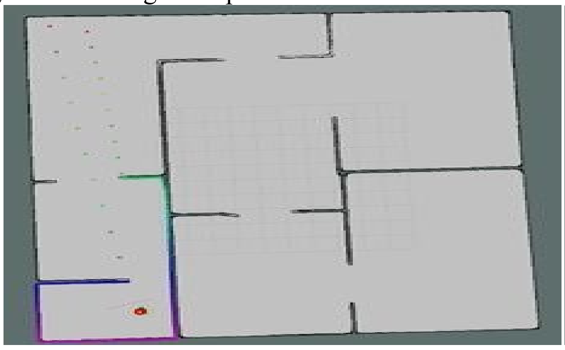


Figure 5

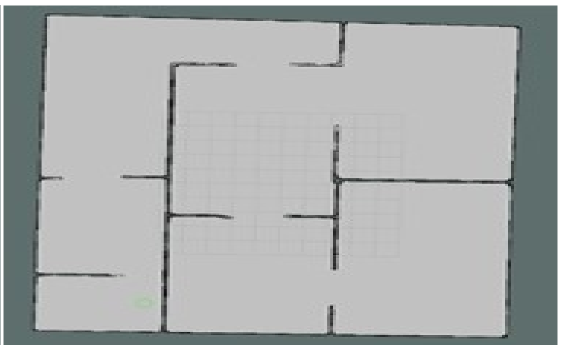
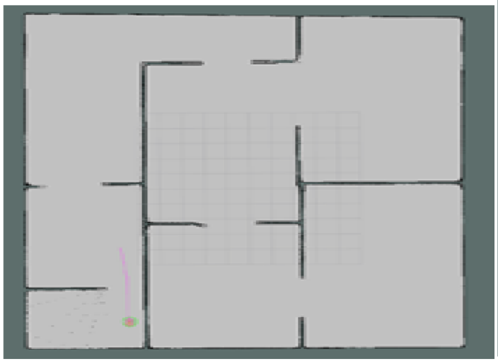


Figure 6

As could be noticed the accuracy of the map has increased but also the computational power. Then the distance threshold parameter maintained constant and the angle threshold parameter has been changed to be 0.8 rad. Figure 6 show the map generated using those parameters.

Then the distance threshold is updated to be 0.8 and the angle threshold is updated to be 0.1. The accuracy of the map has been lowered significantly. Also the time consumed to generate the map and generate a path based on this map has been increased significantly also. Figure 7 shows the map built using those parameters.



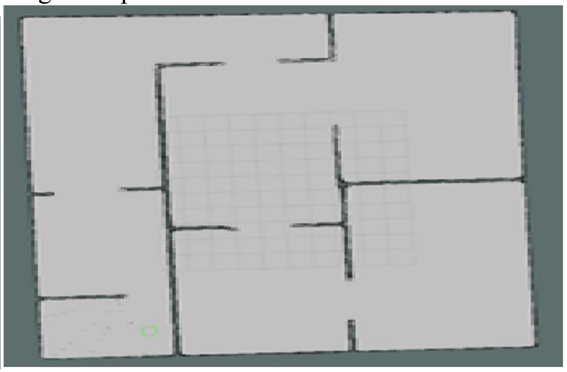


Figure 7 *Figure 8*

Then again the distance threshold is left unchanged and the angle parameter is updated to be 0.8. Figure 8 shows the map built using those parameters. The accuracy of the map again left almost unchanged but the time consumed while building the map again increased significantly.

G-mapping Results (FastSLAM):

For the fast Slam algorithm, two parameters were changed; linear Update and angular Update. The two parameters also indicates the distance and the angle should the robot travel to update the map similar to hector slam. And the default values was 1.0 for linear Update and 0.5 for angular Update. Figure 9 shows the built map using Gmapping package with the default parameters.

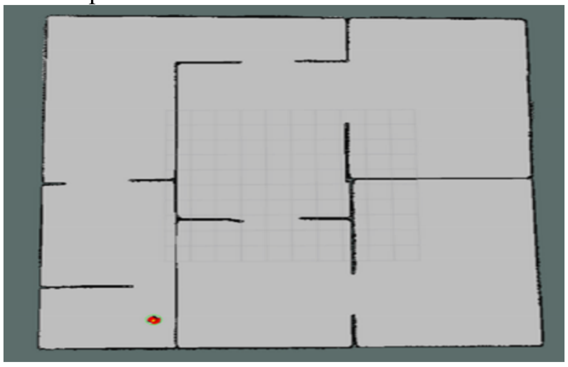


Figure 9

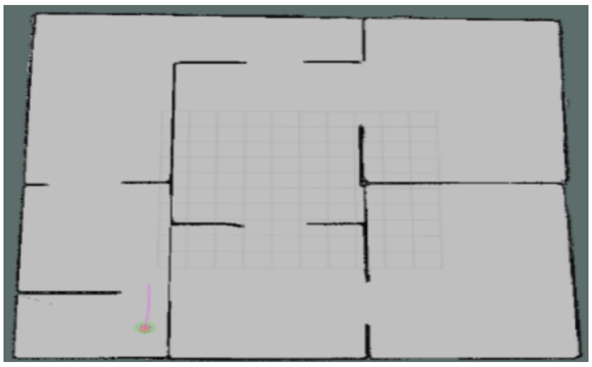


Figure 10

By changing the linear Update from 1.0 to 0.8 and keeping the angular as default, the map becomes as figure 10. For the case of changing the value of *angular Update* from the default value to 0.8 and keeping *linear Update* as default, the built map becomes as figure 11.

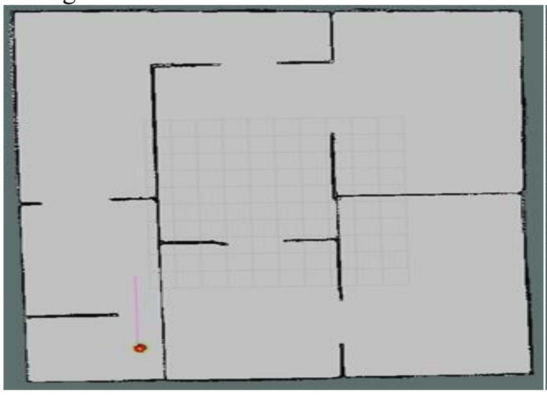


Figure 11

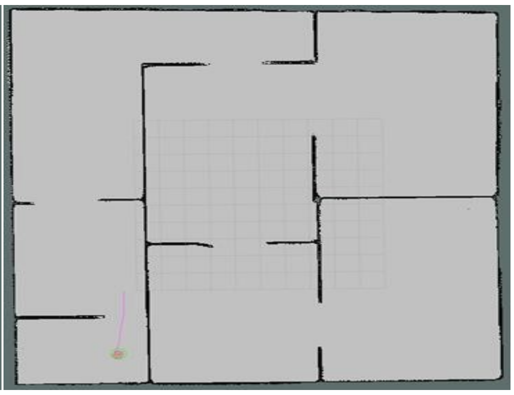


Figure 12

And by changing both the values of linear Update and angular Update into 0.8, the built map becomes as figure 12.

VI- Discussion and Conclusion:

The comparison between the two algorithms conducted on some parameters; time, accuracy, computational power, mape update rate and the effect of changing two parameters in each algorithm with respect to another similar parameters in the other algorithm. The time conducted using ubentu timer. And for the accuracy, it is calculated with regarding to the deviation between the actual map and the estimated map asan average residual error. Map smoothness represents how the map borders clear and smooth.

Comparisons	**Fast slam**	**Hector slam**
Time	4 seconds	2 seconds
Accuracy	0.7 cm error	1.2 cm error

In hector slam, the map update rate is slower than that in fast slam, as in hector slam it is updated in more than one cycle, but in fast slam the map is updated on one time. In hector slam algorithm, as the distance threshold parameter and the angle parameter were reduced, it was noticed that the accuracy of the map has been lowered significantly and the computational power of the map had increased. Also the time consumed to generate the map and generate a path based on this map has been increased significantly. When the distance threshold parameter maintained constant and the angle threshold parameter was decreased, the accuracy of the map left unaffected. For the fast slam algorithm, when the linear Update parameter and the angular Update parameter were reduced the accuracy of the map became higher and the computational power became higher. It also caused the time to increase. It was obvious that the two parameters in both algorithms have a similar effect on building the map. Mainly, the used time for building a map using Hector slam algorithm was much lower than that used in fast slam algorithm. The used time in fast slam algorithm to build a map was almost double the time used to build it using hector Slam algorithm. The resulted maps shows that G-mapping successfully could map with reasonable accuracy. Due to odometry errors, the raw map indicates some misalignments, but the G-mapping was able to correct it.

References:

[1] K. Kamarudin, S. Mamduh, A. Yeon, R. Visvanathan, A. Shakaff, A. Zakaria, L. Kamarudin and N. Rahim, "Improving performance of 2D SLAM methods by complementing Kinect with laser scanner," *IEEE,* vol. 102, 2015.

[2] Murphy, K.P. Bayesian Map Learning in Dynamic Environments. In Proceedings of the Neural Information Processing Systems (NIPS 1999), Denver, CO, USA, 30 November–2 December 1999; pp. 1015– 1021.

[3] Doucet, A.; Freitas, N.D.; Murphy, K.P.; Russell, S.J. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, Stanford, CA, USA, 30 June–3 July 2000; pp. 176–183

[4] Grisetti, G.; Stachniss, C.; Burgard, W. Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE Trans. Robot. 2007, 23, 34–46.

[5] Grisetti, G.; Stachniss, C.; Burgard, W. Improving Grid-based SLAM with RaoBlackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005), Barcelona, Spain, 18–22 April 2005; pp. 2432–2437 .

[6] K. Kamarudin, S. M. Mamduh, A. Y. M. Shakaff and A. Zakaria, "Performance Analysis of the Microsoft Kinect Sensor for 2D Simultaneous Localization and Mapping (SLAM) Techniques," *sensors ,* vol. 22, pp. 23365-23387, 2014.