A Framework for Real Time Indoor Robot Navigation Using Monte Carlo Localization and ORB Feature Detection

Lye Zhenjun, and Humaira Nisar
Universiti Tunku Abdul Rahman
Malaysia

Aamir S. Malik
Universiti Teknologi PETRONAS
Malaysia

***Abstract*—This paper has introduced a framework for indoor navigation implemented by using a computer, Android device and Lego Mindstorms NXT robot. The Lego Mindstorms NXT robot explores and navigates autonomously through a known environment, making its own decisions. An Android device is used for object recognition. The robot is able to localize itself based on the landmark observed using ORB (oriented fast rotated brief) feature detection and the sensory data from ultrasonic sensor using Monte Carlo Localization. The robot is able to plan its own path towards the goal using the A* shortest path. The navigation system is able to identify and recognize the landmarks and environment; and reacts accordingly to achieve the goal. Experimental results show that the robot navigation system is successfully designed and implemented with an accuracy of ±38 cm root mean squared error.**

Keywords—robot navigation; Monte Carlo localization; feature detection; A* algorithm;

I. INTRODUCTION

In this paper, a framework has been developed using the Lego Mindstorms NXT robot to perform indoor navigation. Robot navigation is based on three fundamental elements; localization, path planning and map interpretation. These three elements guide the robot to reach its target based on the desired path. For successful navigation, the robot must be aware of its current position (localization) in the environment, and be able to plan and allocate its path towards the destination. Vision based localization involves recognizing an object or landmark in the map to position the robot relative to the landmark [1]. Using the Monte Carlo Localization algorithm, the robot is able to estimate its position based on the feature matched by the vision system, ultrasonic sensor and motor encoder. Hence the Lego Mindstorm NXT robot will be capable of navigating an indoor environment based on the developed algorithm.

II. SYSTEM ARCHITECTURE

In this paper, the navigation system will be implemented using Lego Mindstorms NXT robot, computer and Android device. Fig. 1 shows the interface between computer, Android device and Lego Mindstorms NXT robot. The computer acts as the command centre to receive the data from Android device and NXT robot. leJOS library [5] is used to program the GUI and build the interface with the NXT robot. The computer will send instructions to the Android device and NXT robot after it processes the data obtained. Monte Carlo Localization [2]-[3] and shortest path A* algorithm [4] are implemented in the computer.

Android device is used as the robot vision for the navigation process. ORB feature detection [6] is implemented in the Android device. A predefined landmark will be available in the database of Android device. The predefined landmark will be used to perform feature matching on the scene in real time. Once the application is running, the landmark detection is initiated. Upon detection, a signal "landmark has been detected" will be sent to the computer through the socket server. The computer will use the signal to re-distribute the particles.

For NXT robot, the NXT intelligent brick is programmed in leJOS NXT [5]. The NXT robot receives direct commands from the computer and reacts accordingly. To prevent errors caused by the delay due to Bluetooth transmission, the motor is controlled by the NXT robot itself instead of being controlled by the computer. The values of the ultrasonic sensor and motor encoders are transmitted to the computer through Bluetooth. Thus, the NXT robot reacts passively based on the command received.

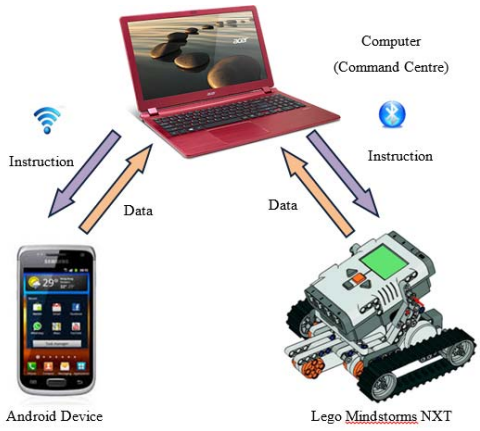

Fig. 1. Interface between computer, Android device and the robot.

III. METHODOLOGY

In this project, Lego Mindstorms NXT robot navigates through a known map and localizes itself by using Monte Carlo Localization together with landmark detection. Fig. 2 shows

the basic block diagram of the navigation process in known map situation. Particles are generated at the start at random locations and are redistributed when landmark is detected with ORB feature detection method as shown in Fig. 3 and 4. After the robot is successfully localized, it calculates its path towards its destination by using A* shortest path algorithm.

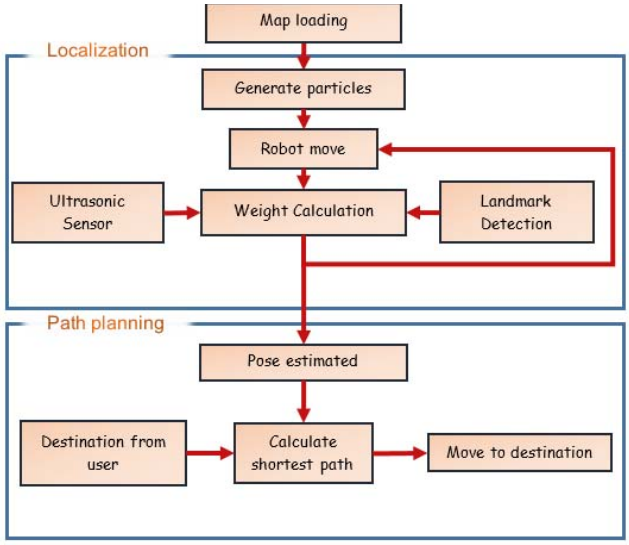

Fig. 2. Block diagram of the navigation process

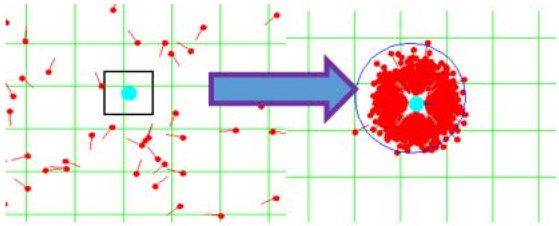

Fig. 3. Redistribution of particles around the landmark, after landmark is detected

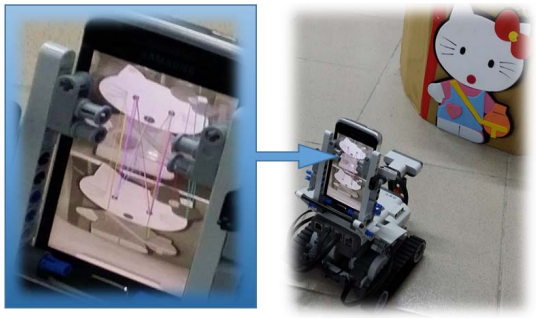

Fig. 4. Landmark detected using ORB feature detection

IV. Experimental Results

Table I shows the result of the navigation process. The experiment area is a laboratory with a size of 930cm×1200cm. The iteration shows the number of scans required by the particle filter to successfully localize the robot. Landmark detected; means the number of landmarks detected during the navigation process. Lost is the number of times when the robot failed to localize itself inside the map. When all particles hold a weight of 0, the robot is said to be "lost" and all the particles will regenerate inside the map, the navigation is reset and has to be restarted. The error is the measure of distance between the estimated robot position and the actual robot position. The number of iterations required in Test 3 and test 4 are lower. This is because the particles are redistributed around the landmark after it is successfully detected as a landmark (see Fig. 2 and Fig. 3).

TABLE I. Localization and Path Error

Test	Localization				Path planning
	Iteration	*Landmark Detected*	*Lost*	*Error (cm)*	*Error (cm)*
1	47	0	2	62	62
2	33	1	1	32	34
3	18	1	0	33	40
4	8	1	0	17	28
5	42	0	1	44	49
6	54	1	2	12	22
7	39	0	1	33	27
8	22	0	1	20	30
Average Iterations = 32		RMSE = 35.0268 cm			RMSE = 38.5649 cm

V. Conclusion

In this paper, an indoor navigation is successfully implemented with Lego Mindstorms NXT robot by using computer and an Android device. By using Monte Carlo Localization, ORB feature detection and Shortest path algorithm, the robot precision is optimized to RMSE of ±38.5649 cm which is insignificant in a map of 930cm×1200cm. Hence we may conclude that the robot is able to navigate successfully through its environment. The results can be further improved by using more sensors. Accelerometer may also be used to reduce the errors in odometry data and laser sensors may be used to replace ultrasonic sensor for better accuracy.

References

[1] Jonathan Dixon, Oliver Henlich (1997), "Mobile Robot Navigation," http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol4/jmd. Retrieved, June 8, 2013

[2] F. Dellaertt, D. Fox, W. Burgard, S. Thrun, "Monte Carlo localization for mobile robots," Proceeding of IEEE International Conference on Robotics and Automation. Detroit, USA, pp. 1322-1328. 1999.

[3] F. Dellaert, D. Fox, W. Burgard, S. Thrun, "Robust Monte Carlo localization for mobile robots," Artificial Intelligence. pp. 99-141, 2001.

[4] Xiang Liu, Daoxiong Gong, "A comparative study of A-star Algorithms for Search and rescue in Perfect Maze,". International Conference on Electric Information and Control Engg (ICEICE), pp. 24-27, 2011.

[5] Solorzano, J., Bagnall, B., Stuber, J., and Andrews, P. lejos, "Java for Lego Mindstorms",http://lejos.sourceforge.net, Retrieved June 30, 2013.

[6] Rublee, E, Rabaud, V, Konolige, K., Bradski, G., "ORB: An efficient alternative to SIFT or SURF," IEEE International Conference, on Computer Vision (ICCV), pp 2564-2571. 2011.