Autonomous Navigation of a Mobile Robot in Indoor Environments

Bimal Paneru, Niraj Basnet, Sagar Shrestha, Rabin Giri and Dinesh Baniya Kshatri

Department of Electronics and Computer Engineering
Pulchowk Campus, Institute of Engineering, Tribhuvan University

***Abstract*—This paper presents the use of probabilistic algorithms in a differential drive robot for totally autonomous navigation in indoor environment. The robot is capable of building map of its environment and global localization in the map thus built. The robot then performs path planning to compute an optimal path towards the goal location. We make use of LiDAR for mapping and localization, and a Microsoft Kinect is used in addition to detect obstacles not falling in the plane of LiDAR.**

***Index Terms*—LiDAR, Localization, Mapping, Navigation, Robot**

I. Introduction

Growing interest in navigation systems is due to the possibility of their use in a wide range of environments and conditions. Fully automatic indoor navigation system has been successfully implemented in robotic restaurant, cleaning robots, warehouse robots. Autonomuos navigation can be broken down into three major tasks - localization, mapping and path planning. The result of navigation is executed by motion control system that implements PID control [1]. The inputs to the position and orientation estimation are sensor data acquired from gyroscope, digital compass and encoders. Complementary filter [2] and Kalman filter [3] are two widely used techniques for sensor fusion. Since Kalman filter gives a better result [4], though computationally complex, we use Kalman filter over complementary.

Commonly used localization approaches include Monte Carlo Localization(MCL) [5] which is a particle filter based method, Extended Kalman Filter(EKF) [6], and Adaptive Monte Carlo Localization(AMCL) [7] which uses adaptive resampling technique to account for particle deprivation problem. Mapping is accomplished through occupancy grid mapping that uses LiDAR scan data to build a two dimensional map of the environment [8].The map thus built is a metric grid of binary random variables each of whose value corresponds to the presence or absence of obstacle at that point.

Given a known robot location, an optimal or suboptimal path can be computed towards a desired goal in the map provided that a possible path exists. Graph search techniques are more popular than other algorithms such as potential field [9] because of the simplicity in dealing with nodes and branches represented by cells and their location in the map. Such graph based techniques include A* [10], D* [11], Dijkstras search [12].

Correct execution of total path planned using these algorithms is an important issue. Moreover, global plan might not always be realizable given the dynamic constraints of the robot. To ensure correct and efficient execution of such plan, local planner is required. Local planner does not completely comply with global plan but finds a suitable compromise between global and local plan. Local planner, basically, takes a segment of the global plan and outputs required velocities for the motors. Dynamic window approach is one of such planners developed to support navigation at greater speed [13]. A complete solution for navigation is found out by the combination of appropriate algorithms to accomplish each task of navigation. Fig.1 shows our approach for navigation of a differential drive robot. IMU and encoder data are fused to obtain an estimate

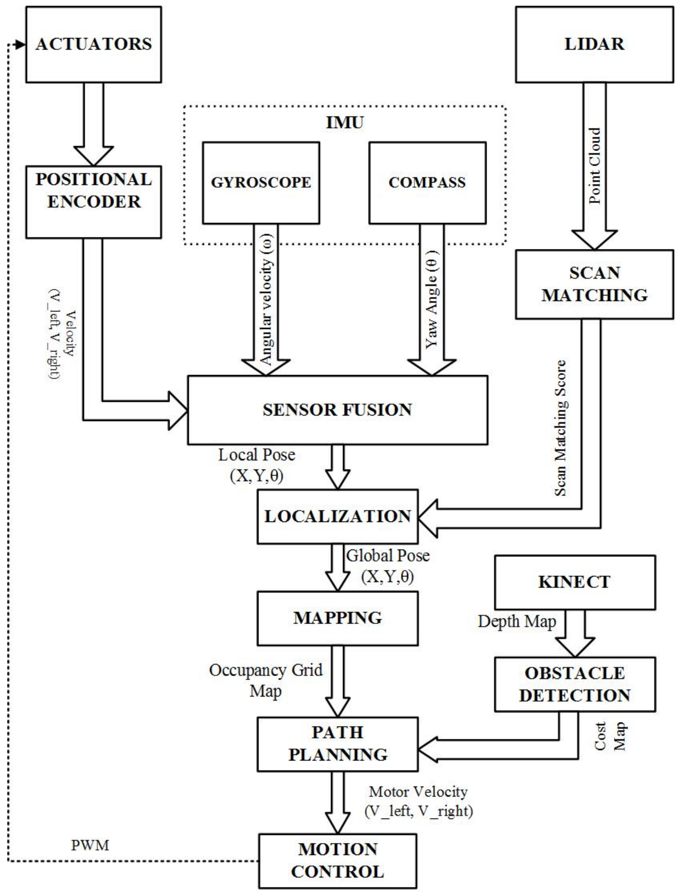

Fig. 1: System Block Diagram

of the local position. Scan matching of LiDAR data with the available map assigns a score to each local pose computed for particles in monte carlo localization. This results in an estimate of the global pose that is used both for mapping and subsequently for path planning. The result of path planning assigns motor velocities to motion control unit which runs PID control to give appropriate PWM signals to the motors.

II. Sensor Fusion Using Kalman Filter

Kalman filter is a Gaussian filter, meaning it assumes that its beliefs (which in our case is going to be the gyroscope measurement) can be represented by multivariate normal distribution. It is probably the best studied technique for Bayesian Filter implementation. The Kalman filtering process can be broken down into two major steps, the prediction stage and the measurement update.

A. The Prediction Stage

It calculates the next state prediction. The next state probability $p(x_t|u_t, x_{t-1})$ must be a linear function in its arguments with added Gaussian noise. The equation representing this function is

$$x_{t|t-1} = F_t x_{t-1|t-1} + B_t u_t \tag{1}$$

$$P_{t|t-1} = F_t P_{t-1|t1} P_{t-1|t-1} F_t^T + Q_t \tag{2}$$

where,
$x_{t|t-1}$ is the predicted state from previous state $x_{t-1|t-1}$.
F_t and B_t are the state transition matrix and input transformation matrix respectively.
$P_{t|t-1}$ is the variance associated with the prediction $x_{t|t-1}$ of an unknown true value of the state.
Q_t is a multivariate Gaussian to model the noise associated with the transition.

B. Measurement Update

The measurement probability must also be linear in its arguments, with added Gaussian noise. This stage is represented by a set of equations that are as follows:

$$x_{t|t} = x_{t|t-1} + K_t(z_t - H_t x_{t|t-1}) \tag{3}$$

$$P_{t|t} = P_{t|t-1} - K_t H_t P_{t|t-1} \tag{4}$$

$$K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t) - 1 \tag{5}$$

where,
z_t is the measurement and can be modeled as: $z_t = H_t x_t + R_t$
In this equation H_t is the transformation matrix to transform state space into measurement space. And R_t is the measurement noise covariance matrix.
The primary parameter from all this computation is the Kalman gain Kt which determines the amount of trust to be put into the measurement and prediction in order to calculate the posterior. The posterior at this state becomes the prior of the next state and so on.

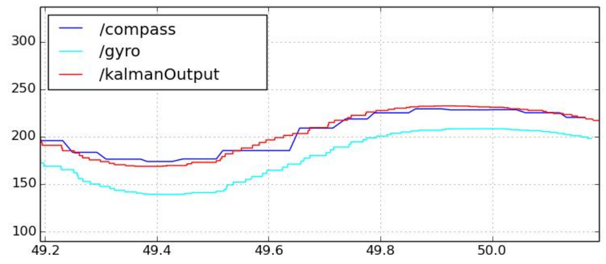

Fig. 2: Sensor Fusion Using Kalman Filter

Clearly in our case, the two inputs are gyroscope and digital compass. The gyroscope data replaces the input u_t. And the measurement z is given by compass data.

$$\begin{pmatrix} x_{angle} \\ x_{bias} \end{pmatrix}_{t|t-1} = \begin{pmatrix} 1 & -dt \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_{angle} \\ x_{bias} \end{pmatrix}_{t-1|t-1} + \begin{pmatrix} dt \\ 0 \end{pmatrix} u_t \tag{6}$$

III. Localization Using AMCL

Adaptive Monte Carlo localization (AMCL) is one of the most popular localization algorithms, which represents estimated pose by particles. It is a variant of implementation of particle filter and an improvised version of MCL, to localize the robot in given world space of map. The algorithm estimates the position and orientation of a robot in the given map as it moves and senses the environment. The algorithm can be used for global localization problem. It uses a particle filter to represent the distribution of likely states, with each particle representing a possible state, i.e. a hypothesis for robots position.
Initially, the algorithm starts with a uniform random distribution of particles over the configuration space, meaning the robot makes an assumption that it can be at any point in given space. So, the initial belief bel(x0) is obtained by randomly generating M such particles from the prior distribution p(x0), and assigning the uniform importance to each particle. For initial set of uniformly distributed particles, for each particle the motion model (odometry model) is computed to find new pose for particles. So actuation command is simulated for every particles no matter where they are and what direction they are pointing. The sensor measurement model is then applied to determine the importance weight of that particle. Then, it randomly draws new particles from the previous belief, with probability proportional to weight of particle. Particles which were consistent with sensor readings are more likely to be chosen and particles which are inconsistent with sensor readings are rarely picked. As such, particles converge towards a better estimate of the robots state. After some interval of time, the particle sets approximate the correct posterior and localization becomes successful. It is easy to implement, and tends to work well across a broad range of localization problems which MCL fails to address like robot kidnapping problem where particle deprivation is the most common issue.
To solve robot kidnapping problem and global localization problems of MCL, the original MCL is augmented with adaptive sampling stage which adapts the number of particles

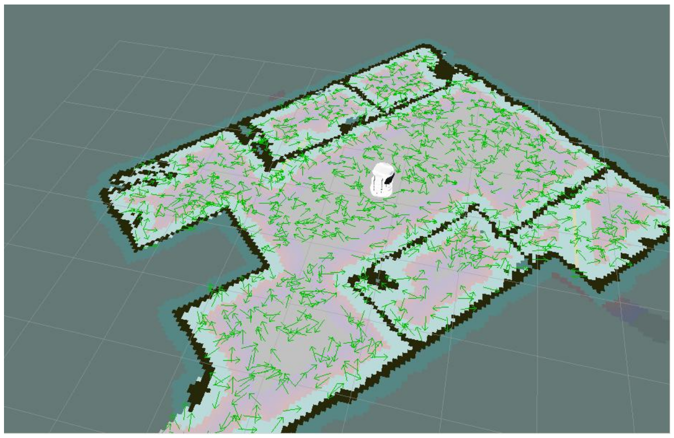

Fig. 3: Random Particle Injection Before Resampling

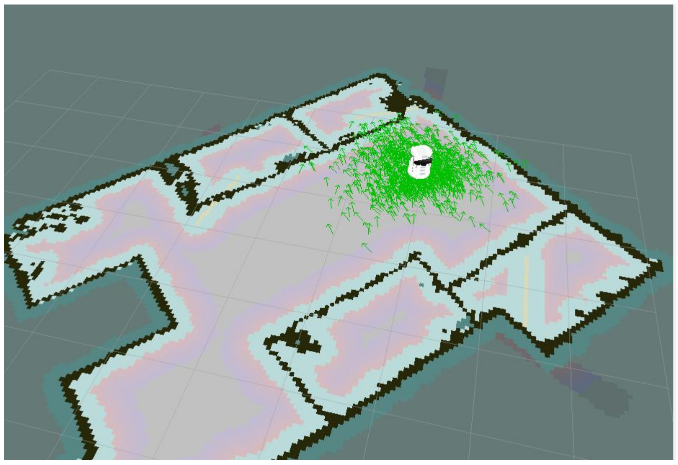

Fig. 4: Converged Particles After Resampling

according to the error in posterior estimate. It is accomplished by using the KullbackLeibler Divergence (KLD) method and since the improvisation is adaptive in nature, the modified MCL is known as Adaptive MCL. The notion of this method is to add random particles to the particle sets and this addition of random particles can be justified mathematically by assuming that the robot might get kidnapped with a small probability, thereby generating a fraction of random states in the motion model. This addition is determined by the degree of localization accuracy through short term and long term measurement likelihoods. The re-sampling condition used in this algorithm requires that $0 \leq \alpha_{slow} \ll \alpha_{fast}$, where parameters α_{slow}, and α_{fast}, are decay rates used in the exponential filters for estimating the long-term and short-term measurement likelihoods respectively. A random sample is dded based on the probability max(0 : 0; 1.0 - $\frac{\omega_{fast}}{\omega_{slow}}$). If the short-term likelihood approximates the long-term likelihood, re-sampling occurs normally without any extra injection of particles. But if the short-term likelihood is worse than the long-term one, then the probability of addition of random samples increases accordingly. Thus, a sudden decay in measurement likelihood results in an increased number of particles in resampling step and number of particles adapt to this change in measurement likelihood. Therefore, in practice, AMCL can consistently outperform and converge faster than classic MCL for localization of robot.

IV. Occupancy Grid Mapping

Occupancy grid mapping is a probabilistic technique to build a map that correctly represents the environment from sensor measurements done after successful localization. The fundamental idea of the algorithm is to represent the environment in two dimensional grid of evenly space cells weighted by binary random variables that determine the presence of obstacle. The algorithm computes approximate posterior estimates for these random variables to correctly represent robots operating environments.

The environment is represented metric grids of variables that correspond the occupancy of the environment. The occupancy of the grid is represented by three color values: gray for

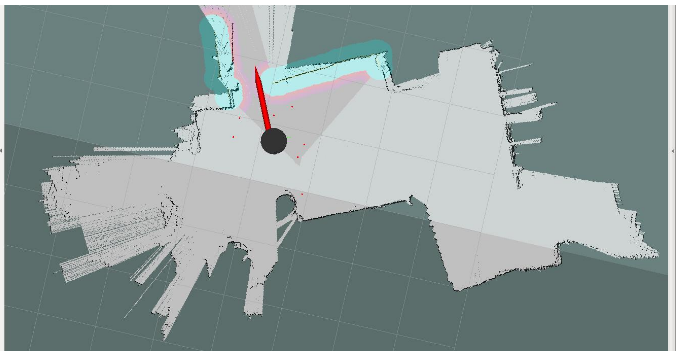

Fig. 5: Occupancy Grid Mapping Using LiDAR

unexplored space, white for free space and black for occupied space.A suitable threshold value is used to transition between these three states.It is particularly suitable for robot with range sensors like LiDAR whose value can be directly raytraced to determine occupancy of desired cell. Robot is allowed to travel only in free space denoted by white region and hence it provides reference for finding paths in unoccupied space preventing potential collision risks. The probability of occupancy of each grid cell m given the measurements $z_1 ... z_T$ is given as

$$p(m|z_1 ... z_t)$$

We calculate the log-odds as:

$$l_{x,y}^{t} = log\frac{p(m_{x,y}|z_t)}{1-p(m_{x,y}|z_t)} + log\frac{1-p(m_{x,y})}{p(m_{x,y})} + l_{x,y}^{t-1} \quad (7)$$

$$l_{x,y}^{0} = log\frac{p(m_{x,y})}{1-p(m_{x,y})} \quad (8)$$

V. Path Planning

The path planning stage is divided into two parts - global and local planner. Global planner plans the total path from the robot location to goal location. This path is further split into several smaller paths to be executed in the local planner. Local planner ensures correct navigation to each intermediate goal points.

A. Global Planner

Global Planner uses A* search algorithm. Each cell in the map is a node in the graph. A* reduces the number of nodes to be expanded from source to goal. In order to determine the likelihood of the node to fall on optimal path, it uses a heuristic function in addition to evaluation function. g(n) is the cost function that gives the distance of node n from the source. Heuristic function h(n) in our case gives the minimum distance of node n from the goal. So the evaluation function of a node is the sum of actual cost from node to source and optimistic estimate (minimum distance) of the cost from node to goal as in equation 9.

$$f(n) = g(n) + h(n) \tag{9}$$

B. Local Planner

The local planner implements dynamic window approach which has been proved to be very efficient for high speed obstacle avoidance. In dynamic window approach, the kinematics of the robot is taken into account by considering only those velocities that can be attained in the next time frame given the acceleration constraints and time period. Given the current robot velocity, a dynamic window is computed that contains all admissible tuples (v,w). This approach is implemented by the maximization of the objective function. The objective function incorporates the following:

$$O = a.target(v, w) + b.velocity(v, w) + c.clearance(v, w) \tag{10}$$

1) Target heading, the measure of progress towards the goal location supporting the direction directly towards the goal location

2) Clearance, the distance to the closest obstacle on the trajectory

3) Velocity, the forward velocity of the robot supporting fast movements

VI. Results

All the tasks of navigation were successfully accomplished with afore mentioned techniques. The robot was able to navigate to a position in a pre-built map by globally localizing itself at first, planning an optimal path and executing motion commands thus generated. Fig.6 shows the formation of occupancy grid map from Odometry and Laserscan data.The 3D point cloud obtained from Kinect also matches the result obtained from laserscan data.
The black pixels as a whole represent some form of obstacle like wall,door,desks and benches. As previously explained, white zone represents free space where robot can travel freely and grey zone represents unexplored region which the robot has still left to incorporate in its map. Fig.7 shows the complete 2D occupancy map generated of a classroom. We can visualize the general floor plan of this room from this map. Each pixel represents an area of 0.05 0.05cm2. The resolution of map can be further reduced to 0.25 cm, however it increases the runtime memory and computation overhead.

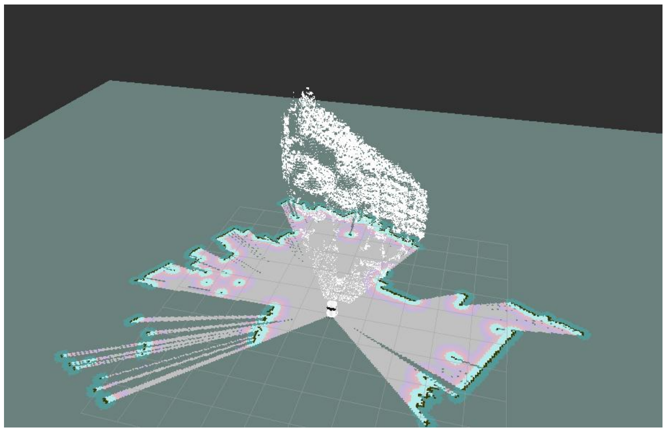
Fig. 6: Mapping From LiDAR Laserscan and Odometry

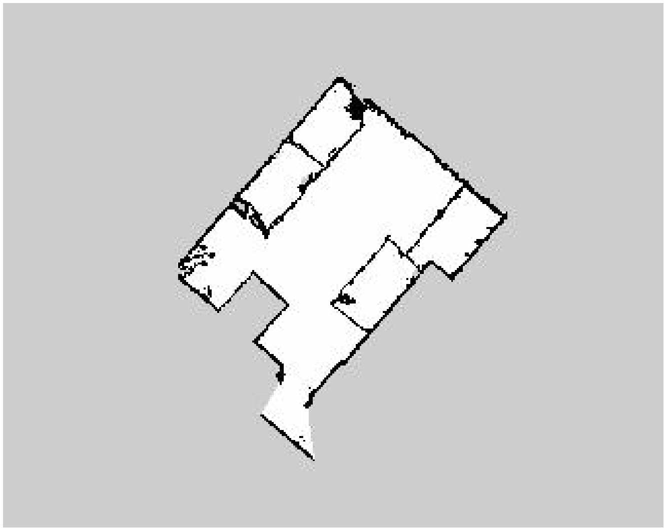
Fig. 7: Complete Map Built of a Classroom

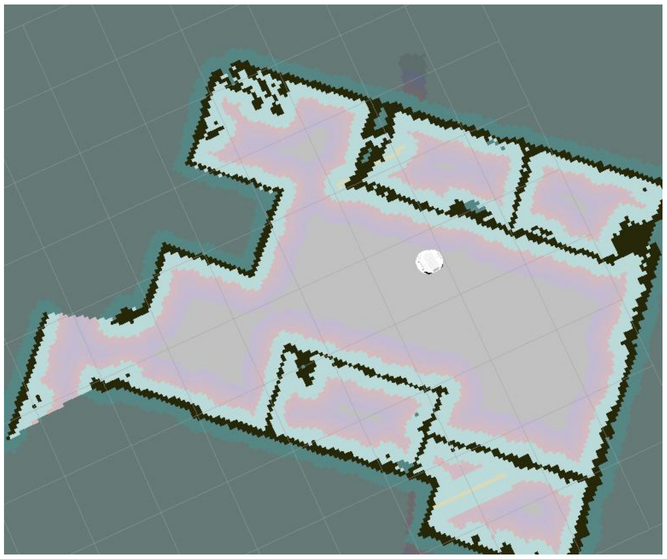
Fig. 8: Global Costmap With Obstacle Layer of 0.3m

Fig.8 shows the region in the whole map where the robot is fit to travel without any risk of collision. The cost of traveling to particular region around the obstacles is increased, hence constituting global costmap. The obstacle layer represented in the map by black pixel figures is further inflated to a certain distance (in our case 0.5m) depending upon the base surface area of robot, so that robot can plan a safe path around these potential collision prone regions.

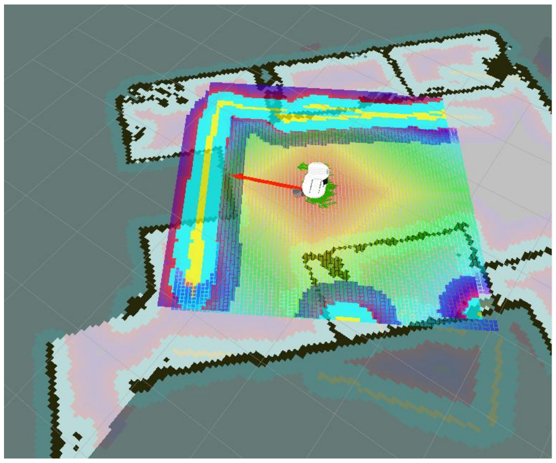
Fig. 9: Local Costmap With Obstacle Layer of 0.5m With Cost Decay 10

Fig.9 shows the local region around the robot in the global map, where robot continuously monitors static as well as dynamic obstacles to plan immediate course of action. The local costmap is a rectangular window of 3 by 3 meters. As seen in the figure, the nearby obstacles are inflated even beyond the global obstacle layer with certain cost decay i.e. cost decreases gradually with distance from any obstacle.

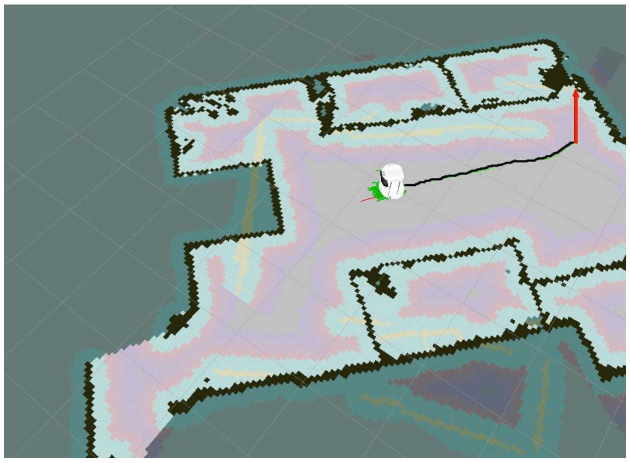
Fig. 10: Global Path Planning

Fig.10 is global path computed from the robots current location to the goal. The path is represented by a black curve joining the robots position to goal location. The global path is computed every second from robots current location to account for the error in navigation.
Fig.11 is the local plan for the robot. Short lines protruding from the robots location is the set of admissible velocities and

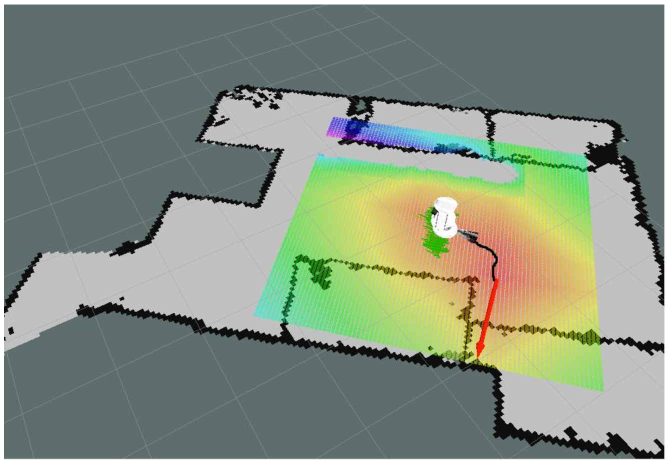
Fig. 11: Local Path Planning

the black cure is the global path. Objective function for each plan is computed and maximized in order to choose a suitable velocity.

VII. Conclusion

We were able to successfully navigate the robot in simulation and in real hardwares with some self built ROS nodes and some packages already existing in the ROS community. Range sensor produced better result during localization than depth camera. So, mapping and localization were done using LiDAR. But since LiDAR scans on a single plane and many objects are completely or partially missed on its plane, Kinect depth camera was used to gain appropriate estimate of the obstacles occupancy. Use of depth camera in addition to range sensor has enabled us to effectively locate obstacles and navigate clear off them. To sum up the results, sensor fusion of IMU data for orientation was accomplished with 2 degree tolerance. Localization with MCL was done with 500 particles and in the adaptive mode it could increase to as many as 2000 particles. Mapping was done with grid size 2048 * 2048 and scale: 1 pixel equivalent to 0.05 meters. And finally, path planning resulted in less than 0.2 meters deviation on final location. Research of algorithms and their comparisons were made thoroughly during the development of the robot.

References

[1] M. Araki, Pid control, Control systems, robotics and automation, vol. 2, pp. 1-23, 2002.

[2] P. C. Glasser, An introduction to the use of complementary filters for fusion of sensor data, Research Paper, 2013.

[3] R. E. Kalman, A new approach to linear filtering and prediction problems, Transactions of the ASME-Journal of Basic Engineering, vol. 82, no. Series D, pp. 35-45, 1960.

[4] W. T. Higgins, A comparison of complementary and kalman filtering, IEEE Transactions on Aerospace and Electronic Systems, vol. 11, no. 3, pp. 321-325, 1975.

[5] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, Monte carlo localization for mobile robots, in Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, vol. 2. IEEE, 1999, pp. 1322-1328.

[6] K. Fujii, Extended kalman filter, Refernce Manual, 2013.

[7] D. Fox, Kld-sampling: Adaptive particle filters, in Advances in neural information processing systems, 2001, pp. 713-720.

[8] S. Thrun, W. Burgard, and D. Fox, Probabilistic robotics, MIT press, pp. 221-241, 2005.

[9] Y. Koren and J. Borenstein, Potential field methods and their inherent limitations for mobile robot navigation, in Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on. IEEE, 1991, pp. 1398-1404.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, 1968.

[11] A. Stentz, Optimal and efficient path planning for partially-known environments, in Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on. IEEE, 1994, pp. 3310-3317.

[12] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische mathematik, vol. 1, no. 1, pp. 269-271, 1959.

[13] D. Bertsekas, Rollout algorithms for constrained dynamic programming."