Project Report

B.E. Project 2012

**Localization and Mapping with a Mobile Robot and Kinect sensor**

**Team**: 01

**College:** Fr. Conceicao Rodrigues College of Engineering, University of Mumbai

Ashish Belokar (BE Computer Engineering)

Kshitij Dholakia (BE Computer Engineering)

Swapnil Kate (BE Computer Engineering)

Table of Contents

**1. Introduction3**

**2. Problem Statement4**

**3. Requirements5**

3.1 Hardware5

3.2 Software implementation diagram6

**4. Implementation7**

4.1 Robot assembly7

4.2 Circuit diagram7

4.3 Software implementation diagram 8

**5. Testing strategy9**

**6. Discussion of system10**

**7. Future work11**

**8. Conclusion11**

**9. References12**

**1. Introduction**

A Robot capable of carrying out Simultaneous Localization and Mapping (SLAM) is essential in generating a view (example, a map) of an unknown (lacking a priori knowledge) environment.

The process would enable the robot to learn about its surroundings, in terms of path (the movable areas in the environment), Landmarks (any object that exists in the environment, for example, walls, equipment, furniture, etc.).

This would, in turn, make the robot ready to carry out tasks, in an environment, which it is now relatively familiar to.

The tasks could range from navigation and path planning to observation of the existing Landmarks to more complex feature exploration, semantic object recognition and detection, path planning, object fetching, etc. The complexity of application proportionately necessitates a capable robot.

**2. Problem Statement**

Localization is the process by which a robot estimates its pose in a given environment. The method by which it does this depends on the equipment it possesses (for example, it could use GPS to map its coordinates) and is unavailable in a vast majority of places (outer-space, inside buildings, mines, etc.) This necessitates self-localization of the robot with respect to a global reference point in the environment (usually its initial position.) However, odometry values are error prone and with time, the error assimilates. Therefore, it is necessary to correct the pose estimates with referential observations to Landmarks in the environment. Two successive observations of a Landmark from two different poses can be used to establish the more accurate pose estimate of the robot. The recording of Landmarks with which relative observations are made, amounts to the mapping of the environment as explained below.

According to the standard formulation of the SLAM problem, a robot executes controls and accumulates observations of its environment, both corrupted by noise. Each control or observation, coupled with an appropriate noise model, can be thought of as a probabilistic constraint. For example, each control probabilistically constrains two successive poses of the robot. Observations, on the other hand, constrain the relative positions of the robot and objects in its environment. As the network of constraints expands, new observations can be used to update not only the current map feature and robot pose, but also map features that were observed in the past.

**3. Requirements**

3.1 Hardware Requirements

|  |  |
| --- | --- |
| **Component** | **Quantity** |
| Arduino Uno | 1 |
| Xbox 360 Kinect | 1 |
| Platform I, II | 2 |
| 60 rpm DC Motors | 2 |
| Wheels | 2 |
| Castor Wheel (large) | 1 |
| Sensor (encoders) | 2 |
| Motor-driver (L298) | 1 |
| Digital compass | 1 |
| Battery Li-ion (12V) | 1 |
| Battery Pb-acid (12V) | 1 |
| Clamps | 2 |

3.2 Software Requirements

***Operating System***

Ubuntu 11.10

Robot Operating System

***Programming***

C++

Arduino IDE

***Drivers***

Arduino Uno (for Arduino microcontroller)

OpenNI (for Kinect)

***ROS Stacks, Libraries***

OpenNI with pointcloud\_to\_laserscan

rosserial

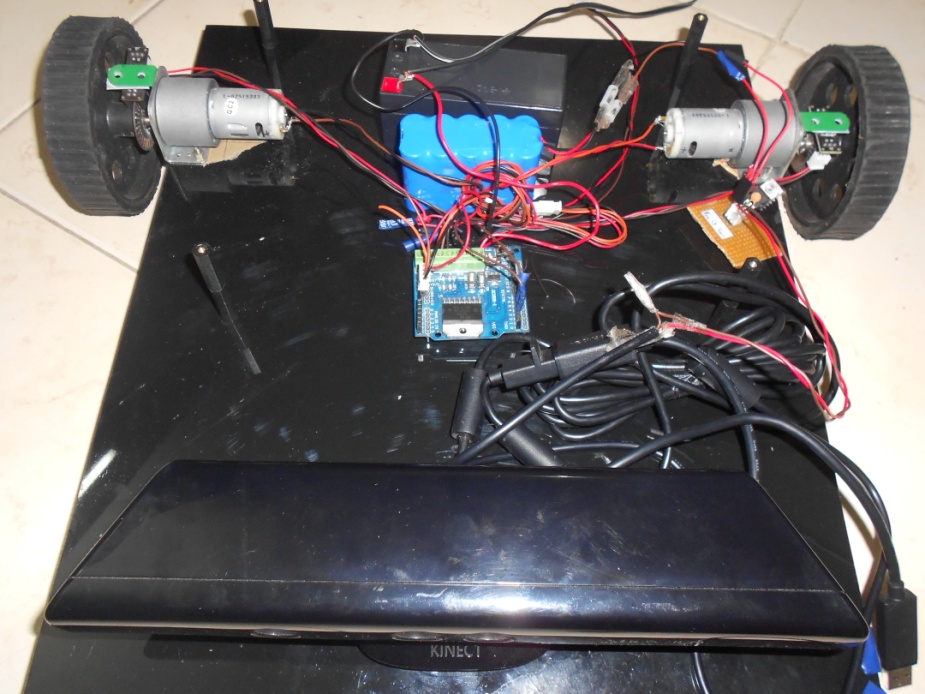
tf

gmapping

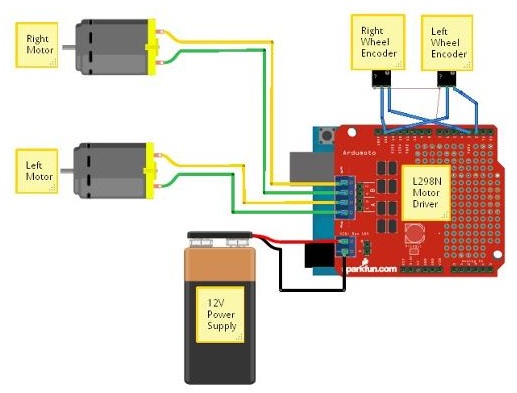
urdf

**4. Implementation**

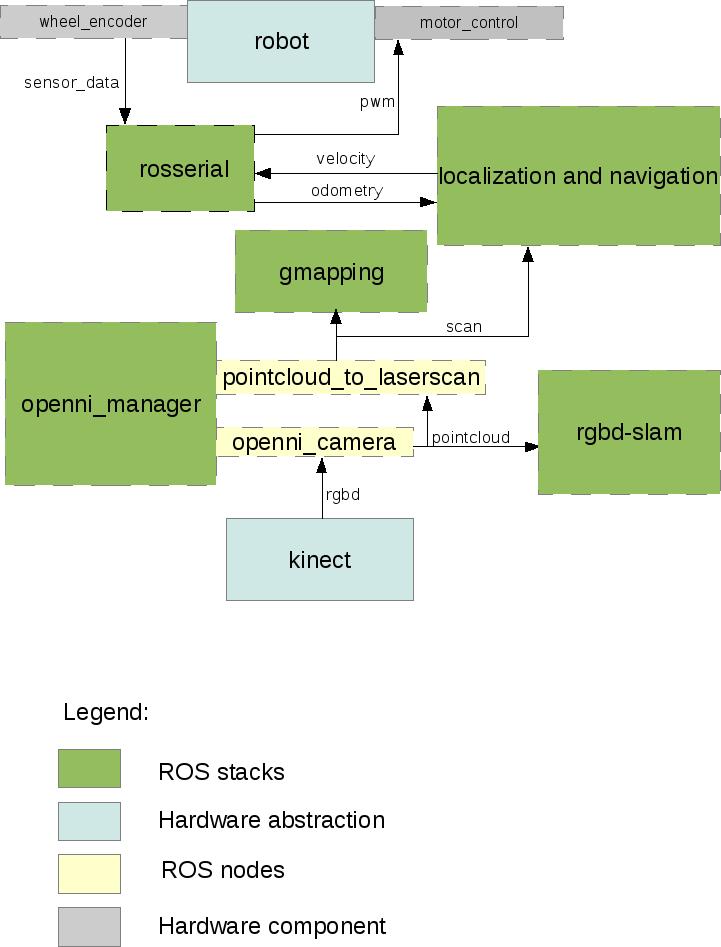
4.1 Robot Assembly



4.2 Circuit Diagram



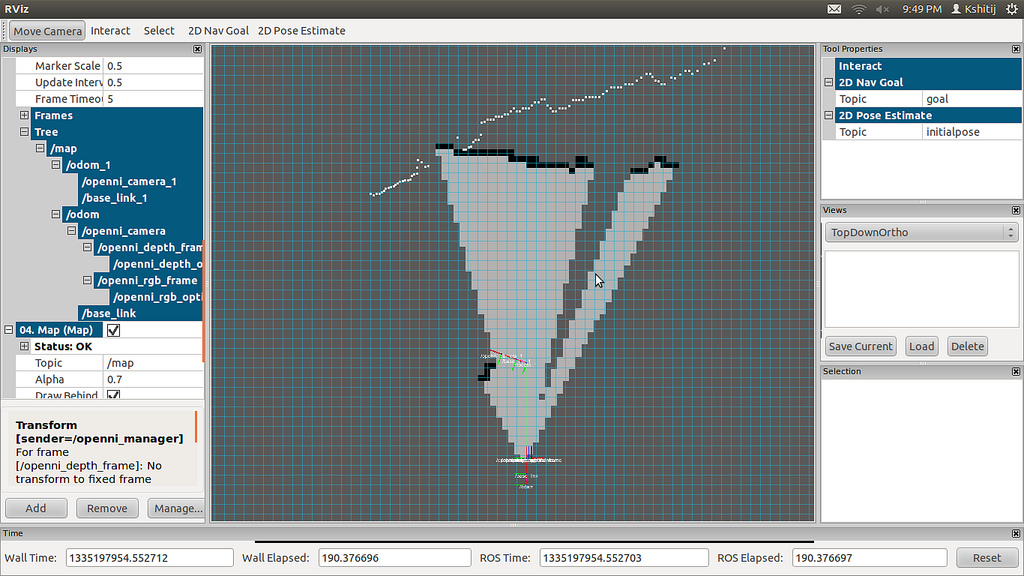
4.3 Software Implementation Diagram



**5. Testing Strategy**

The robot is given a predefined trajectory and allowed to map. This should be done in an indoor environment, as Kinect works only indoors. The map generated is then tested to see if it is consistent with the surroundings.

A circular trajectory is programmed into the robot. Therefore the map appears to be a radial one, recording any obstacle in the vicinity as a black cell in the grid. The open spaces in the grid are in blue.



**6. Discussion of System**

What worked as per plan?

* Development of a robot as per our specifications
* Interfacing Kinect and visualizing its data (rgbd snaps)
* Localization using dead reckoning
* Mapping

What we added more than discussed in SRS?

* We overlooked the fact that we had to work with multiple coordinate frames in our robot
* During our reading of ROS, we realized this and implemented 3 coordinate frames and the entire transformation process between them

Changes made in plan

* Used OpenNI instead of OpenKinect as Kinect driver because of ROS support
* Used Arduino Uno instead of Atmega because Arduino can run as ROS node
* Used slam\_gmapping as mapping tool instead of EKF

**7. Future Work**

The output of our project is the generation of a map which is a 2D occupancy grid. This map is defined in a ROS message nav\_msgs/OccupancyGrid. This message is readable by the robot, and we can implement path planning and navigation using this data structure. This would involve interacting with the robot by giving it goal coordinates – and it would plan a path to reach it in the most efficient way.

Further, we could use Kinect to register 3D point clouds and create 3D maps and eventually 3D occupancy grids, which then can be used to do path planning in a 3D workspace (like in robot manipulators.) This can be essential in pick and place type activities.

Kinect can also be used to implement various object identification and classification algorithms which can be used in autonomous pick and place robots.

**8. Conclusion**

We developed a mobile robot platform, suitable for indoor use. It’s equipped with wheel encoders for pose estimates as well as a Kinect sensor, a depth sensing device. This set up is well suited for carrying out mobile robotic experimentation.

Further, we studied the Robot Operating System, an implementation framework for robots such as the one we developed – a tool that provides hardware abstraction and message passing. Using this, we could easily publish the robot pose estimates, a prerequisite for robot localization.

We could interface the Kinect, a device capable of producing point clouds, with which we experimented with rgb-d slam, a package that uses visual odometry to create 3D maps of environments.

Next, we went on to study and implement coordinate transforms for our robot. We could successfully use the odometry data, and similarly the Kinect depth/scan data and transform it to the base link frame of reference, a representational point on the centre of the robot.

**9. References**

[1]. *SLAM for dummies - A Tutorial Approach to Simultaneous Localization and Mapping* by Søren Riisgaard and Morten Rufus Blas, MIT

[2]. *FastSLAM* – *A factored approach to SLAM by* Montemerlo, CMU

[3]. *Autonomous Robots: Modelling, Planning and Control by* Farbod Fahimi

[4]. *Robot Operating System* – *http:// www.ros.org, Willow Garage*

[5]. *A Quantitative analysis of Two Automated Registration Algorithms In a Real World Scenario using Point Clouds from the Kinect by* Jacob Kjaer

[6]. *Arduino Open Source (*http://arduino.cc*)*

[7]. *RGBD Slam* (http://www.ros.org/wiki/rgbdslam*)*