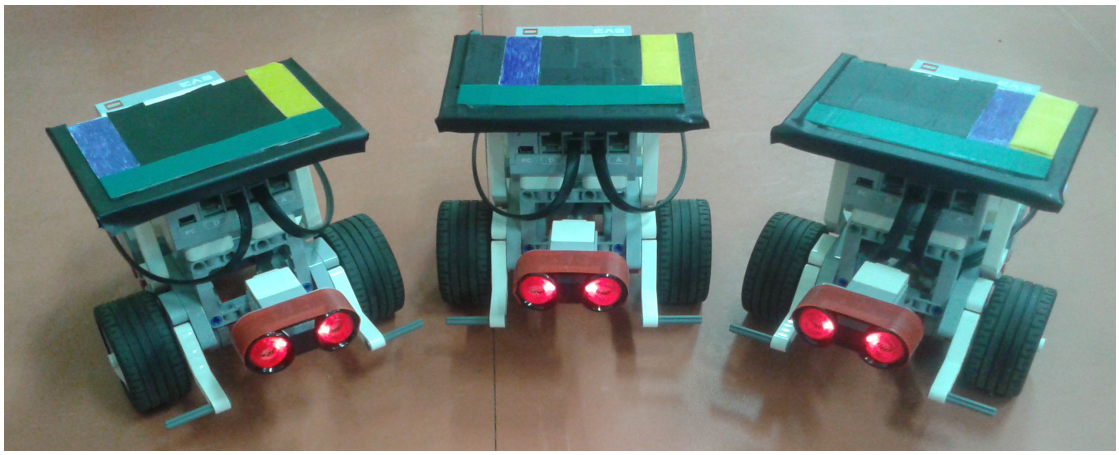
DEPARTMENT OF ELECTRICAL ENGINEERING

ID 4100 CREATIVE ENGINEERING PROJECT

A Testing platform using LEGO Mindstorms kits

S.R. Manikandasriram
srmanikandasriram@gmail.com
Guide: Dr. Bharath Bhikkaji
Jul-Nov 2014



1 INTRODUCTION

The objective of this project is to design and build a testing platform using LEGO Mindstorms kits that would help in validating algorithms in the fields of Artificial Intelligence, Control systems and other motion planning techniques. The platform consists of three LEGO robots and a ceiling mounted camera with a birds-eye view. Each LEGO robot has a differential drive system and an Ultrasonic Range finder for obstacle detection. This system can be easily extended to more number of robots. The localization module and low-level motion controllers

are made accessible to the user. The localization is performed using the birds-eye view camera and the low-level motion controllers ensure that the robot tracks a prescribed reference signal. These reference signals are typically generated using planning algorithms that need to be validated.

The entire source-code for the system is publicly available via GitHub through two repositories

1. ev3-ros
   This repository contains the the client software which would run in each of the LEGO robot.

2. LEGO Testing platform
   This repository contains the master/server software which would run in the central PC.

The system consists of three modules which would be explained in the subsequent sections

1. Low-level motion controllers

2. Vision based localization

3. Integration with Robot Operating System(ROS) framework

2 LOW-LEVEL MOTION CONTROLLERS

The low-level motion controllers are directly implemented on the LEGO EV3 hardware. For this project, a debian based OS, called ev3dev, developed specifically for EV3 has been used. The client software takes velocity commands from the master and calculates the required angular velocities of the individual motors. Then, a separate PID controller for each motor tracks the e desired angular velocities.

2.1 PID CONTROLLER

Each of the LEGO EV3 motor comes with in-built quadrature encoders which measure the rotation of the motors. Using this information as feedback, a standard PID controller is implemented. For this project, the default K_p, K_i, K_d values were found to be sufficient. But the **ev3dev** software provides enough freedom to tune these constants or design a different type of controller as well.

The desired angular velocities for the left and right wheels are calculated from the overall robot velocities using the following transformations

$$v_l = \frac{(v_x + \frac{L}{2} * \omega_t)}{R * \frac{\pi}{180}} \tag{2.1}$$

$$v_r = \frac{(v_x - \frac{L}{2} * \omega_t)}{R * \frac{\pi}{180}} \tag{2.2}$$

where,
v_l : angular velocity of left wheel in deg/s
v_r : angular velocity of right wheel in deg/s
v_x : required linear velocity of robot in m/s
w_t : required angular velocity of robot in rad/s

Since a differential drive robot cannot move perpendicular to its current orientation, the v_y component of the velocity command is ignored.

2.2 Encoder based Odometry

The in-built encoders produce 360 ticks per revolution which amounts to a 1° resolution. Using these encoder readings, a *dead-reckoning* based approach is adopted to localize the robot. This is achieved using the following transformations:

$$\Delta x = \frac{R * (\Delta l + \Delta r)}{2} \tag{2.3}$$

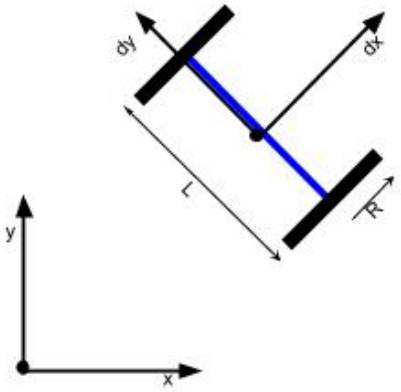
$$\Delta y = 0 \tag{2.4}$$

$$\Delta \theta = \frac{R * (\Delta l - \Delta r)}{L} \tag{2.5}$$

$$x \leftarrow x + \Delta x * cos(\theta) \tag{2.6}$$

$$y \leftarrow y + \Delta x * sin(\theta) \tag{2.7}$$

$$\theta \leftarrow \theta + \Delta \theta \tag{2.8}$$



A major limitation of the *dead-reckoning* approach with low resolution sensors is the exponential increase in error. Even a 1° error in the estimation of θ would lead to a large error when the robot moves forward by $1m$. In order to counter this, the vision based localization system is used to reset the positions periodically (say once in 2 seconds).

2.3 Sensors

The LEGO EV3 kit comes with a variety of sensors including IR beacon, Ultrasonic Rangefinder, Sound, Light and contact sensors. These sensor information can be broadcasted through the network using ROS. An Ultrasonic rangefinder sensor is available for obstacle avoidance. The sensor measurements are *published* as a *LaserScan* message using ROS. The program closely resembles the example program used in the ROS tutorial - Writing a Simple Publisher and Subscriber (C++) and can be easily extended to other sensors. The **ev3-ros** package

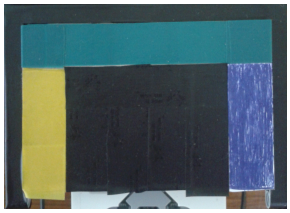
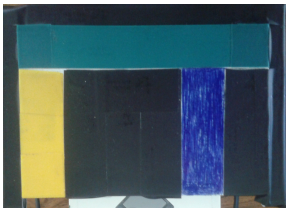
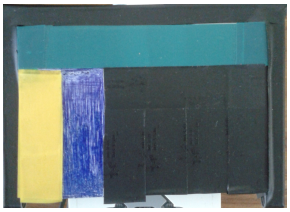
additionally has programs which enables the use of Color sensors and Contact sensors in order to increase the scope of usage.

3 Vision based Localization

Vision based localization of the different robots is achieved using coloured markers. The markers are designed to uniquely determine the identity of the robot and also determine its orientation.

3.1 Coloured Markers

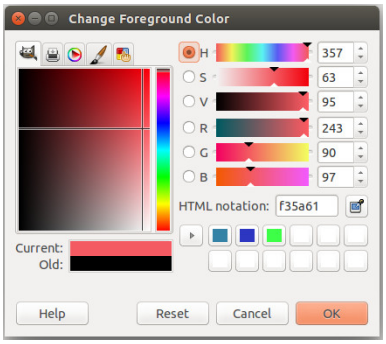
The markers are made using strips of coloured tapes on a black background. Each marker has three colours - Green, Blue and Yellow. The relative positions of these strips is used in uniquely identifying the robot and also to estimate the orientation of the robot.

3.2 HSV vs. RGB color space

The default color space used for representing a pixel is RGB (Red Green Blue). But this representation is difficult for color based segmentation. An alternative is the HSV (Hue Saturation Value) color space which is amenable for color based segmentation.

The 6 sliders in the figure represent the colors that would result in changing the corresponding values. As can be observed, the HSV color space provides a clear grouping of the color so that various shades of the same colour differ only in the Saturation or Value component of the pixel while the RGB color space lacks this nice property.

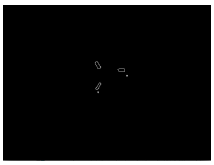


The color based segmentation is performed using OpenCV library.

Algorithm 1 Color based segmentation

1: **procedure** GETCONTOURS(frame)
2: cvtColor(frame,BGR2HSV)
3: inRange(frame,lower_limit,upper_limit) ▷ The limits depend on colour
4: apply MORPH_OPEN filter ▷ to remove pepper-and-salt noise
5: detect edges using Canny detector
6: find contours
7: filter contours based on area
8: calculate centroid of the contours
9: **return** centroids



The first image shows the captured frame. The next image shows the frame after Step 3 in the GetContours procedure. The third image is after Step 4 and the fourth image is after Step 7. Once the contours and their centroids are determined, the contours of different colours are grouped together based on distance. The three centroids in each group form a unique triangle using which the orientations of the robot is estimated.

Due to noisy data, all the blobs might not be detected in every frame. As can be seen in this frame, the yellow blob from one of the robot markers was not successfully detected and hence its overall position and orientation could not be determined.

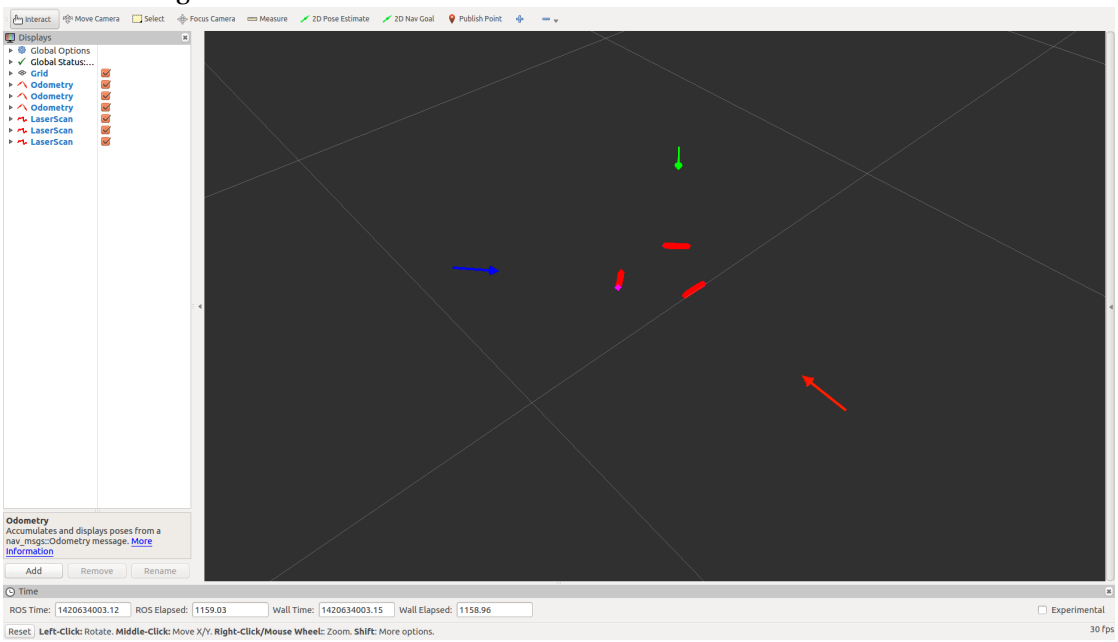
4 Integration with ROS framework

Robot Operating System (ROS) is a widely used library of softwares that are used by roboticists around the world. ROS provides a unified framework for developing and sharing Robotics software. All the major robot manufacturers provide drivers which are compatible with ROS. Thus, the ability to interface the LEGO EV3 robots with ROS greatly expands the scope of using LEGO EV3 kits in research and education. The ROS framework provides a robust framework for communication between the various nodes (robots and/or PC) in the ROS network. In this project, three robot nodes and one PC node (which is connected to the camera) are used. More robots and/or PCs can be easily added to the same ROS network but only one node can act as the master node.

4.1 Sensor data visualization

A powerful feature of ROS is the Visualization tool called **rviz** that allows graphical visualization of the sensor data and robot movements.

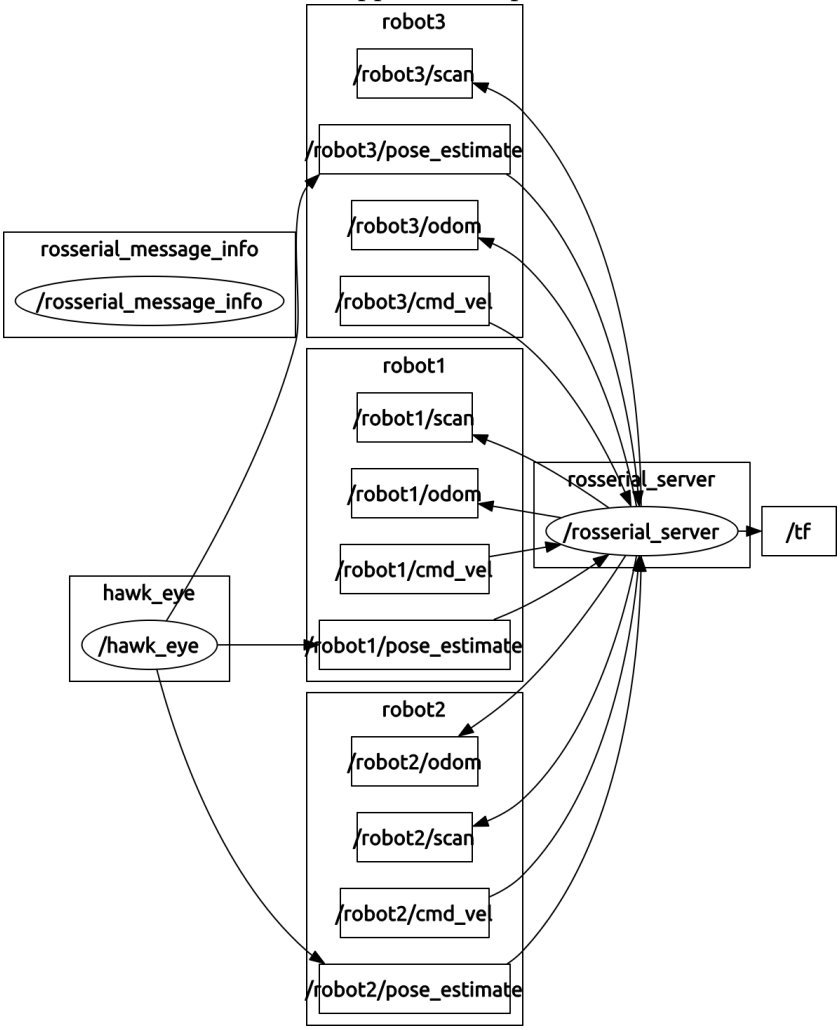
In the following picture the position of three robots and the respective ultrasonic range finder's readings are visualized. Refer ROS documentation on **rviz** for further details.



4.2 Interfacing with LEGO EV3 using ROSSerial

ROSSerial is a ROS package that is used to communicate with each of the robots. This package allows a ROS network to communicate with embedded linux devices, arduinos and other

embedded systems. You can refer the ROS documentation for further details. The following picture shows the ROS network graph for this system. The rosserial_server node behaves as a proxy between the PC node and the individual LEGO EV3 robot units. Hence to the main ROS network, all the information appears to be published and subscribed by rosserial_server node.



4.3 KEYBOARD/JOYSTICK TELEOPERATION

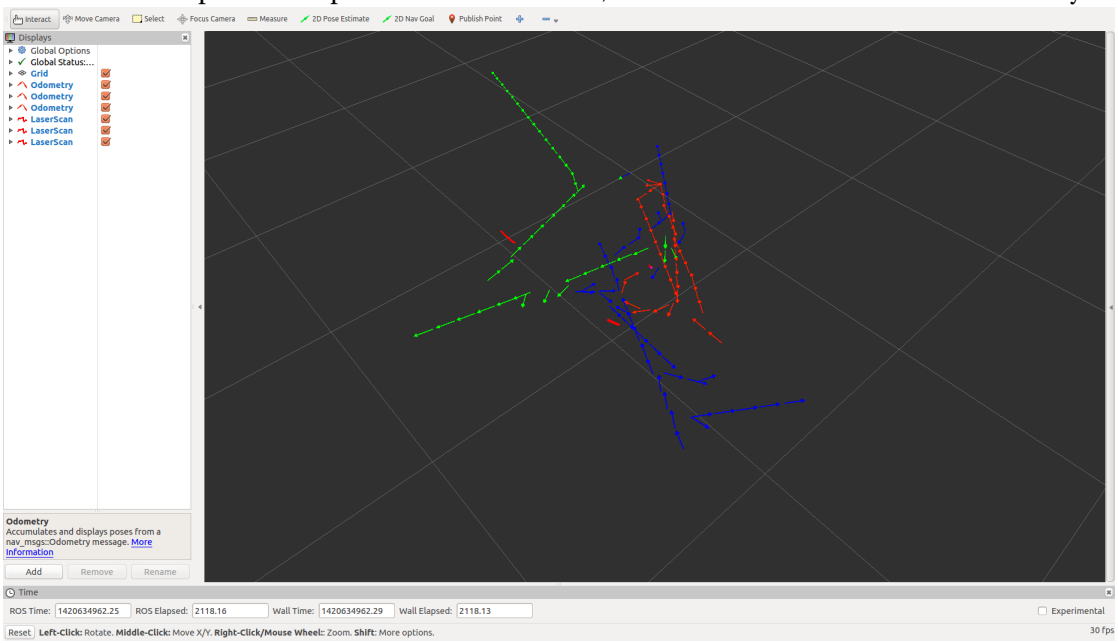
The teleop_twist_keyboard and teleop_twist_joystick packages available in ROS can be used to remotely control the individual lego robots via keyboard and joystick respectively. By default, these packages publish velocity commands via *cmd_vel* topic. But in order to individually control different robots in the same network, the velocity commands have to published in their respective namespace (i.e.) */robot1/cmd_vel*, */robot2/cmd_vel* and so on....

4.4 Random exploration with obstacle avoidance

Now that we are capable of controlling the three robots individually and also localizing the robots accurately as long as they remain within the field of view of the aerial camera, we can use this platform to test higher level algorithms. As a simple example, a *wanderer* script has been written. This script sends velocity commands to each of the robot based on a set of rules:

1. if an obstacle is present less than 20cm ahead, turn right
2. if another robot is nearby, move backward and turn right simultaneously
3. else move forward

The following figure displays the odometry information of the three robots as they ẃanderedb́ased on the same simple rules. Apart from visualization, this data can be stored for further analysis.



5 Future Scope

The integration with ROS framework in itself greatly extends the scope of this platform. The project can be extended in various dimensions some of which have been listed below:

1. Built-in behaviors (move in an arc, move forward by distance, turn by angle etc...) for the differential drive robots available as ROS services.
2. Implementing tracking to improve accuracy and speed of the vision system
3. Adding a gripper mechanism to the robots to allow for testing manipulation problems
4. Replacing the stationary birds-eye view camera with a camera mounted on Quadrotor for use in outdoor environments.

6 Conclusion

Through this project, we have developed a testing platform using LEGO Mindstorms kits. The system uses OpenCV for vision based localization and ROS for communication and control. These two software libraries are widely used and have a very active developer community thus greatly improving the scope of usage. Further, a behavioral algorithm, wanderer script, was tested using the platform and it was found to be very useful. Also, the validated algorithms can be directly used with a different set of robots or a mixture of different types of robots with different drive systems due to the universal nature of the ROS framework.