A PROJECT REPORT ON

PSI BOT (Ψ)

Indoor Navigation Robot

Submitted to

"INNOVATION CELL"

IIT BOMBAY

By

Shubham Wagh(2nd yr EEE,VNIT,Nagpur)

Rohit Salem(2nd yr EEE,VNIT,Nagpur)

Tanvi Rastogi(2nd yr EEE,VNIT,Nagpur)

Varsha Reddy (1st yr ECE,IIT,Guwahati)

Acknowledgement

At this juncture we feel deeply honored in expressing our sincere thanks to Prof. Abhishek Gupta, Prof. S. N. Merchant, Prof. V. Karthik and Prof. Ramesh K. Singh for providing us with the magnificent opportunity to work on this project.

We are obliged to the Indian Institute of Technology, Bombay for extending tremendous support to such endeavors which take students to the high ridges of science.
We are highly indebted to the Sine Lab and the Innovation Cell, IIT Bombay for providing us with the prospect and the tools for the successful completion of this project.

We take this opportunity to express our profound gratitude to Pranav Maheshwari, our mentor, for his exemplary guidance throughout the course of this project. We would like to express our deepest regards to Ravi and Kriti for their stimulating suggestions for our project. Special thanks to all the members of the Innovation Cell for their assistance and constant encouragement throughout the course of this project. This experience has not only provided us with scientific understanding but has been enjoyable all along.

INDEX

1. Abstract
2. Introduction
3. Bot Description
4. Components
5. Design and implementation
   (i) System Architecture
   (ii) Tele-Operation of the robot
   (iii) Localization of the robot
   (iv) Mapping
   (v) Navigation and Path planning

6. Future insight

Abstract

In the large area of autonomous robot navigation, encompassing map creation, path planning and self-localization, we develop the idea of building Psi, a simple autonomous indoor navigation robot. The main objective of this robot is to autonomously travel in the premises and do basic tasks, avoiding any dynamic obstacle that comes in its path by relying on vision and wheel odometry information.

Accomplishing this includes the following steps –

- Being able to manually move the robot with the help of keyboard keys or a joystick using appropriate motors and a motor controller.
- Being able to perform localization. Kalman filters and particle filters are methods of localization that can be employed.
- Performing SLAM – Simultaneous Localization and Mapping.
- Path planning using algorithms such as Dijkstra's and A-Star.
- Efficiently traversing this path by successfully implementing the Proportional Integral Derivative control.

Introduction

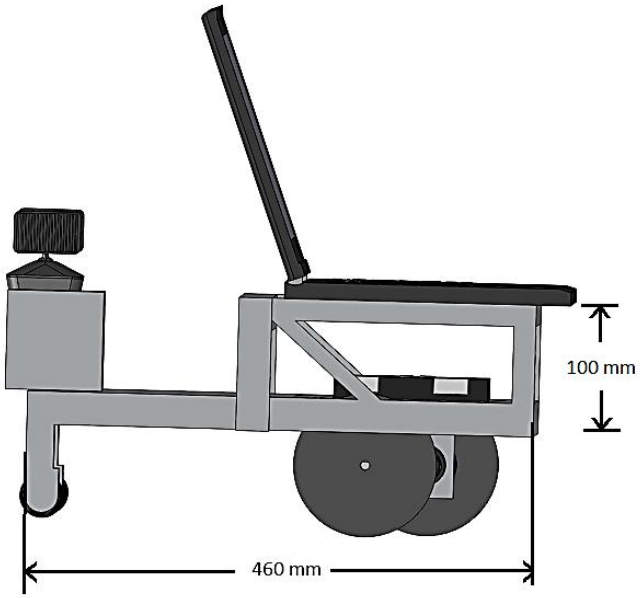
In today's world everything is being automated to save human effort and time. Prominent companies such as John Deere, Steelcase, and HP use autonomous indoor navigation bots to humanize remote conversations and to bridge the gap between formal and informal interaction. Recently, Raffi Krikorian, VP of Engineering at Twitter bought Double, one such bot, to act as his replacement when he can't attend a meeting and to interact with his colleagues, while he is someplace else, by making Double move the isles of his office.

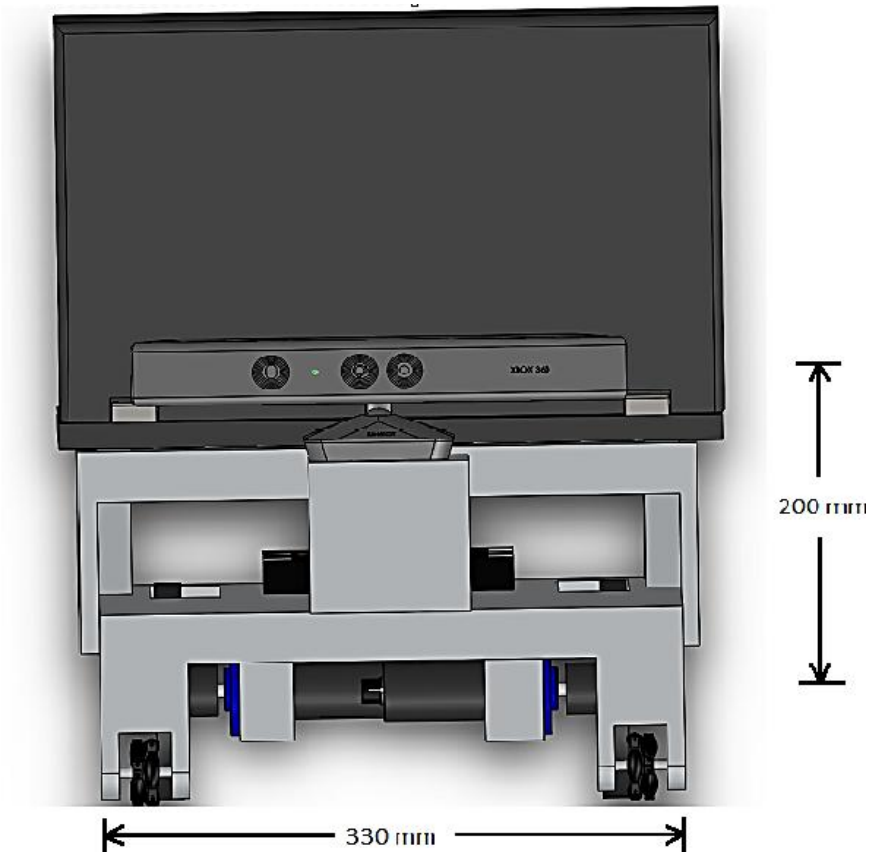
Getting necessary tasks done such as fetching files, documents or beverages from the comfort of your bed or office is becoming more viable to employers and more attractive to employees.

In this project, we develop PSI, an indoor navigating office bot. Given a prior knowledge of the environment and the goal position, the robot has the ability to safely move towards the goal using its knowledge and sensorial information.

Sensors like Kinect, LIDAR and ultrasound help in safe navigation. They provide information about the obstacles surrounding the robot, allowing the application of path-planning algorithms such as A* and Dijkstra's to achieve safe and robust navigation. They also help in mapping and localization. Encoders are a rich source of information for robot indoor localization. Motor controllers such as Roboteq can be used to move the robot.

Bot description





Components

1. Kinect XBOX 360



Kinect has a single foot for stable positioning with a motorized tilt. It consists of three major devices:

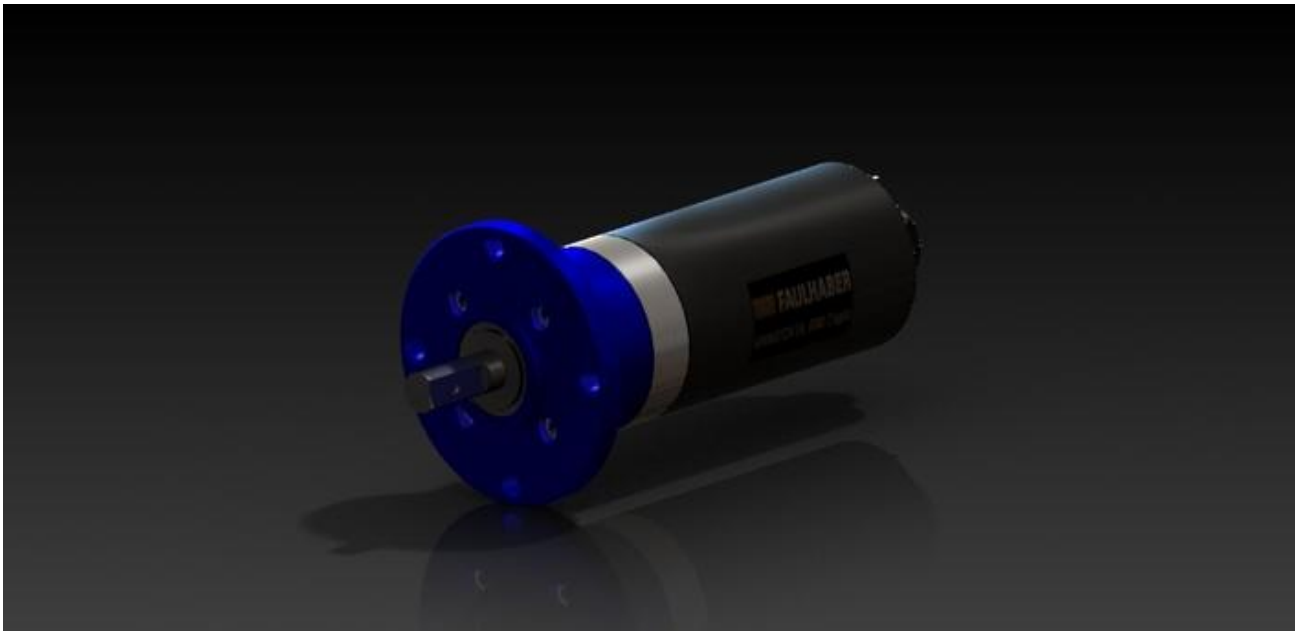
1. VGA Camera

2. Infrared projector and sensor

3.Microphone array

For this report, infrared projector and CMOS(complementary metal oxide semiconductor) sensor work together to get the 3-D image of the environment.

2.FAULHABER 3257G024CR



It is a brushed DC-micromotor.For more details about the encoder channels and specifications ,click here.

3.ROBOTEQ MDC2250



Roboteq's MDC2250 controller is designed to convert commands received from an RC radio, Analog Joystick, wireless modem, PC (via RS232 or USB) or microcomputer into high voltage and high current output for driving one or two DC motors.

The controller features a high-performance 32-bit microcomputer and quadrature encoder inputs to perform advanced motion control algorithms in Open Loop or Close Loop (Speed or Position) modes. The MDC2250 features several Analog, Pulse and Digital I/O‘s which can be remapped as command or feedback inputs, limit switches, or many other functions. The controller's two motor channels can either be operated independently or mixed to set the direction and rotation of a vehicle by coordinating the motion of each motor. Numerous safety features are incorporated into the controller to ensure reliable and safe operation.

Mechanical Components:

- Aluminum bars and links.
- Screws, rivets, and lock nuts.
- Aluminum wheels with synthetic rubber gripper.
- Omni wheels.

Miscellaneous Components:

- LiPo batteries
- PCB
- Wires
- Heat sink tubes
- Emergency switch

Design and implementation

System architecture

The Robot System's software is based on the Robot Operating System (ROS), a software framework for robot softwaredevelopment, which is installed on the laptop. The ROSprovides libraries and tools including drivers for the Kinect and robot base. ROS is preferred as it is an Open Source platform and it provides standard services such as hardware abstraction, low level device control, implementation of commonly used functionality, message passing between processes and package management.

Interfacing Roboteq with Roborun plus motor utility software .This software helps to check the motors for their encounter counts, rpm, PID (proportional integral derivative) tuning, etc.

The project implementation is done in four stages:

- **Tele-Operating the robot:**

In this, the robot is controlled using the keyboard's arrow keys and then later by gamer's joystick.

Firstly, we subscribed the "turtle1/cmd_vel" topic from the turtlesim's turtle_teleop_key node. This topic passes message type "geometry_msgs/Twist" .The API Linux documentation of Roboteq , contains the set of commands , for example setting roboteq to mixed mode/separate mode ,closed/open loop ,getting values like counts , angular velocities etc.

A subscriber node subscribes the topic mentioned, and sends the commands to the roboteq to drive the robot similar to that of the motion of turtle in turtlesim node.

- ## Localization of the robot :

Localization is the problem of estimating the place (and pose) of the robot relative to a map; in other words, the robot has to answer the **characteristic question**, *Where am I?* Typically, solutions comprise tracking, where the initial place of the robot is known, and global localization, in which no or just some *a priori* knowledge of the environmental characteristics of the starting position is given.

- ## Wheel Odometry

Odometry requires a method for accurately counting the number of rotations of the robot wheels. A standard method for doing this is to instrument the wheels with shaft encoders or IMU (Inertial Measurement Unit) or both (for efficient localization).In this report shaft encoders are used because they just efficient enough in an indoor environment where there is negligible slipping.

Mathematics :

For the differentially steered robot, the location is constantly updated using the following formula. The distance the robot has traveled since the last position calculation, and the current heading of the robot, are calculated first:

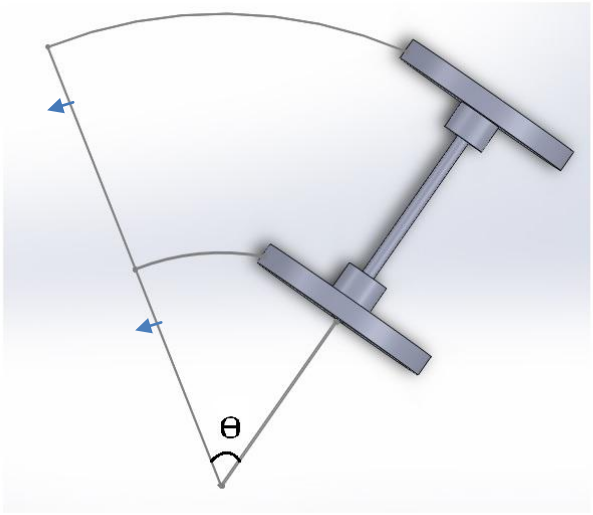
(1) distance = (left_encoder + right_encoder) / 2.0
(2) theta = (right_encoder - left_encoder) / WHEEL_BASE;

Where WHEEL_BASE is the distance between the two differential drive wheels.

Note that the system returns a 0 for straight ahead, a positive number for counter-clockwise rotations, and a negative number for clockwise rotations.

With these two quantities, a bit of trigonometry can give the robot's position in two dimensional Cartesian space as follows:

(3) X_position = distance * sin(theta);
(4) Y_position = distance * cos (theta);



- **3D Pointcloud to Laserscan**

There are many laser scanning devices like, for example LIDAR but LIDAR is very expensive, so we used Kinect for making the robot cost effective.

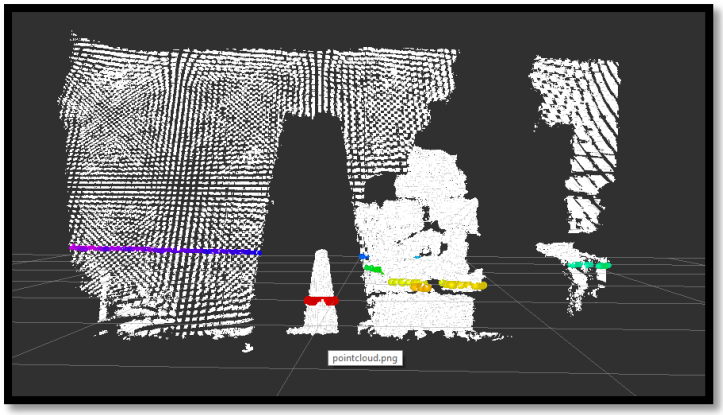
Kinect provides 3D depth data which can be used for mapping and navigation but requires a lot of processing .The same purpose can be served by 2D laser scanning, which requires less processing. So it is required to convert 3D depth data to 2D laser scan, which is done by “depthimage_to_laserscan” package .This package access the “pointcloud2” data from the Kinect and converts it to laser scan data.

The laser scan data is then published on the topic “scan” which is of type “sensor_msgs/LaserScan”. When this package is executed a nodelet is loaded

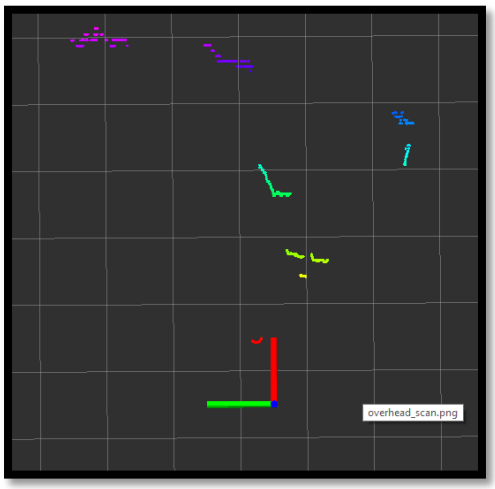
Example scene: Converting depth image to laser scan



The original scene



sensor_msgs/LaserScan projected on top of the sensor_msgs/PointCloud2



Top down view of the sensor_msgs/LaserScan

- **MAPPING**

Gmapping, an in-built package in ROS for SLAM (simultaneous localization and mapping), requires a source of odometry and a source of depth information. Depth information is provided by the laser scan data. Hector slam can also be used for the same purpose but gmapping is more efficient as it provides more parameters which can be set for fine tuning. We are using mainly the default values of the parameters but specifically chose a relatively high particle count to increase the accuracy which suffers from the fact that the Kinect sensor has a much smaller horizontal field of view than a real laser scanner.

Tf

As the robot has multiple coordinate frames like base_link, camera_link,odom, etc. It is necessary to keep track of all these frames over time.Tf takes into consideration how these frames are related to each other and links the other frames with respect to the fixed frame.

- ## Navigation and Path Planning:

This aspect involves taking in information from odometry, sensor streams and a goal pose to output safe velocity commands that are sent to a mobile base.

Path planning is accomplished through both global and local planners. Global planners work on a global costmap which is used for global planning, i.e., creating long-term plans over the entire environment. Local planners are used for local planning and obstacle avoidance.

Global planners such as navfn and carrot planner work on a given global map to provide a fast interpolated navigation function that can be used to create plans for a mobile base. The planner assumes a circular robot and operates on a costmap to find a minimum cost plan from a start point to an end point in a grid. In this project, we employed navfn whose navigation function is computed with Dijkstra's algorithm.

Given a plan to follow and a costmap, local planners such as base local planner and dwa local planner provide a controller that

produces velocity commands to send to a mobile base. Their output pertains to a local costmap. These packages can work with any robot whose footprint can be represented as a convex polygon or circle and can expose its configuration as ROS parameters that can be set in a launch file. We used base local planner which provides implementations of the Trajectory Rollout and Dynamic Window approaches to local robot navigation on a plane.

After configuring the local and global costmaps and path planners, the navigation stack must be launched.

BLOCK DIAGRAM

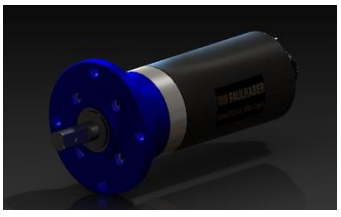






POWER SUPPLY





Future Insight

1. Incorporating speech recognition in the robot, thereby enabling it to take goal poses orally. This would be an important asset to the bot, increasing its ability to interact with human beings.

2. Incorporating a more efficient and robust path planning algorithm. A star algorithm, for instance, produces the same output as Dijkstra's algorithm but has a much lower time complexity.

3. Controlling the robot via simulation with Android phones, iPads, iPhones, PCs, etc.

4. Integrating IMUs with encoders for improved odometry.

5. With the increase in the prevalence of wireless LAN, “Wi-Fi” access points can be used for localization

For further details and codes regarding the report click here.

