IMR HOMEWORK 6

Bulichev Oleg, Koshechkin Alexey

Dichboi project

Overview

We did the full work together, but the biggest part of setting the experimental conditions, robot algorithm development and data collection was performed by Koshechkin Alexey, biggest part of map building, line extraction and data visualisation was performed by Bulichev Oleg.

We built an autonomous mobile robot using LEGO Mindstorms set

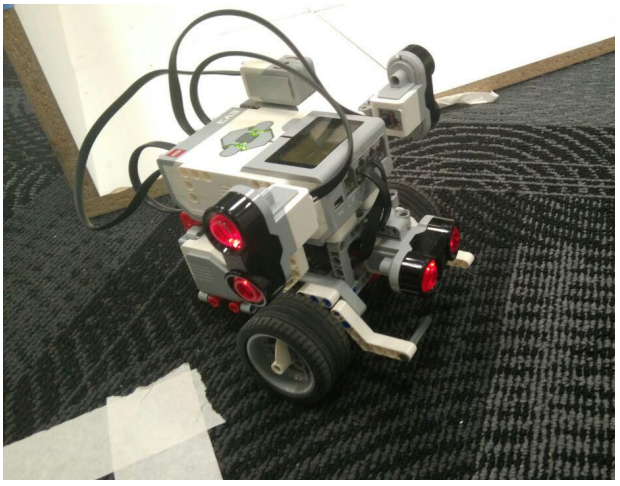

Fig. 1 Mobile robot set

Robot has 3 sonars attached, forward sonar for detecting obstacles and side sonars for mapping an environment. Also they were used for navigation in order to keep in the middle of the track and turn accordingly. Robot has gyroscope for calculating the angle and wheel encoders to obtain odometry data. The robot uses ev3dev environment and used a python script in order to navigate using the real time sensor data. The code is in the github repository

https://github.com/Lupasic/hw6_imr

2. The corridor

Oleg built a track for mapping. It contains 5 turns :p

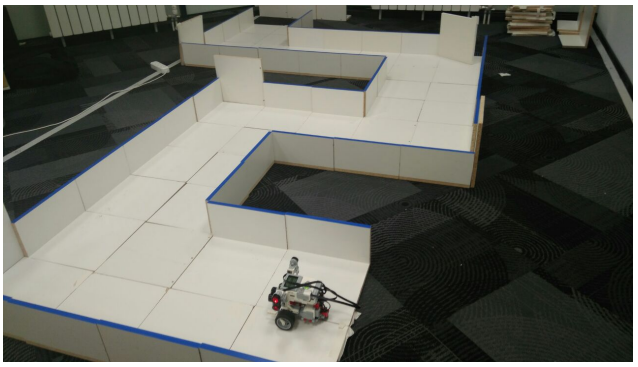

Fig. 2 Corridor for mapping

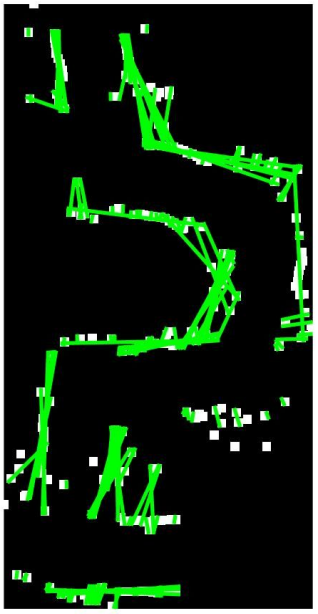

Fig. 3 Result map of the corridor built by sensor data

3. Driving

The robot went from the start to the exit by sensors in real time while collecting map data. The robot did not collide with the walls. Robot collected sonar data every 0.2 sec.

The link to the video is here:

https://youtu.be/HCY2skGZJvg

4. Map building

Obtained odometry data was processed, fused with gyroscope data by Extended Kalman Filter

Then the cloud points of walls was recreated from sonar data on every data record step. After that dilatation was applied and Hough line extraction method was used in order to approximate the data into lines

5. Software

Ev3dev was used to control the robot systems.

Python script was used to collect data and operate wheel speed.

Matlab script was used for position estimation, walls position estimation and visualisation.

Numerous experiments

Basically, the code performs a control loop every 0.2 seconds where movement parameters are set and all the sensor and odometry data is obtained and recorded to the external file. Control is conducted by setting speed of wheels on every step according to the state of the robot and sensor data.

Angle of turn is determined by the previous saved states, difference between side sonar data and the data from the front sonar. The algorithm of movement is the following.

Linear speed is set initially. If the robot determines that there is an obstacle in less than 12 cm in front of it it addes part of the angle turn to the side with the bigger distance to the side wall. Then it calculates the difference between the side sonar data and reaches to the middle of the corridor. The value of the turn is determined by PID controller. P part controls the speed of the turn, D part smoothes the movement and avoids the jerking moving and I part eliminates the static error.

Data analysis

It was decided to receive next data:

1) Distance from left, right and forward sonars in cm.
2) Angle from gyroscope in degrees.
3) Angle velocity from both motors in degree per sec.
4) Position from both motors in encoder pulses.
5) Number of tacho counts in one rotation of the motor

All this data is needed to create a map. The main idea is to find odometry and after this - find the position of wall points by sonar data.

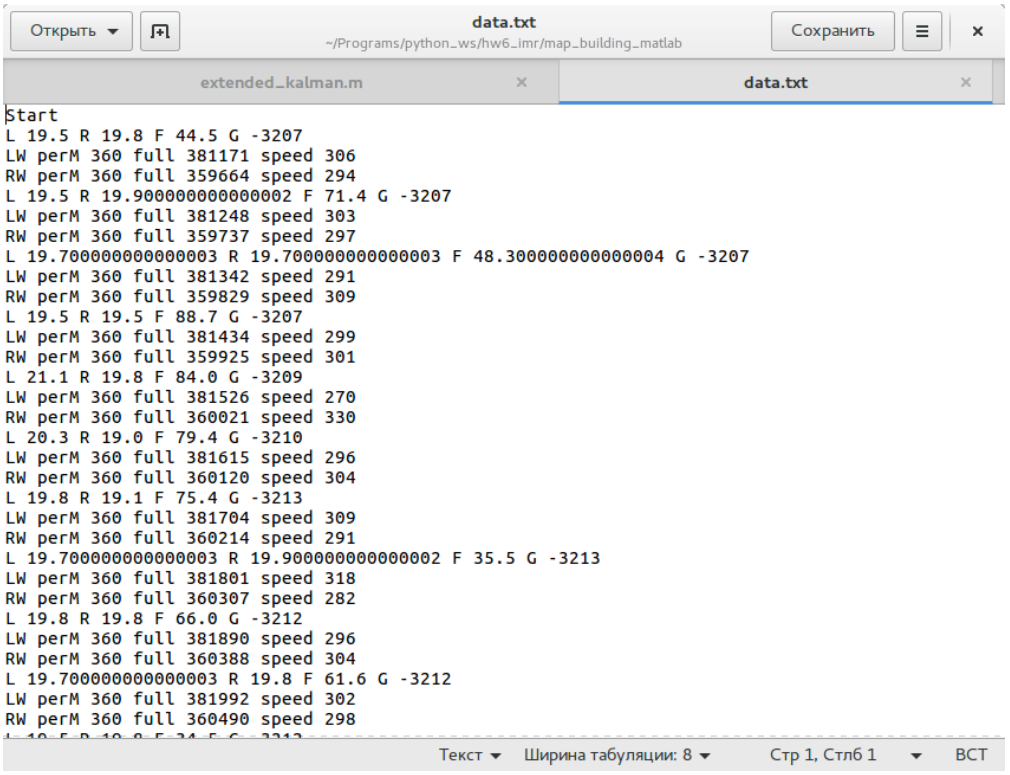

Fig. 4 Data received from robot

Odometry receiving

I decided to use Extended Kalman filter for doing this task. The main reason - encoder data is suitable only for a small distance and i need to use ground truth for eliminating error. Moreover i can fuse angle data from encoders and gyro. This approach give me more realistic results.

I code it using matlab.

The results is following:

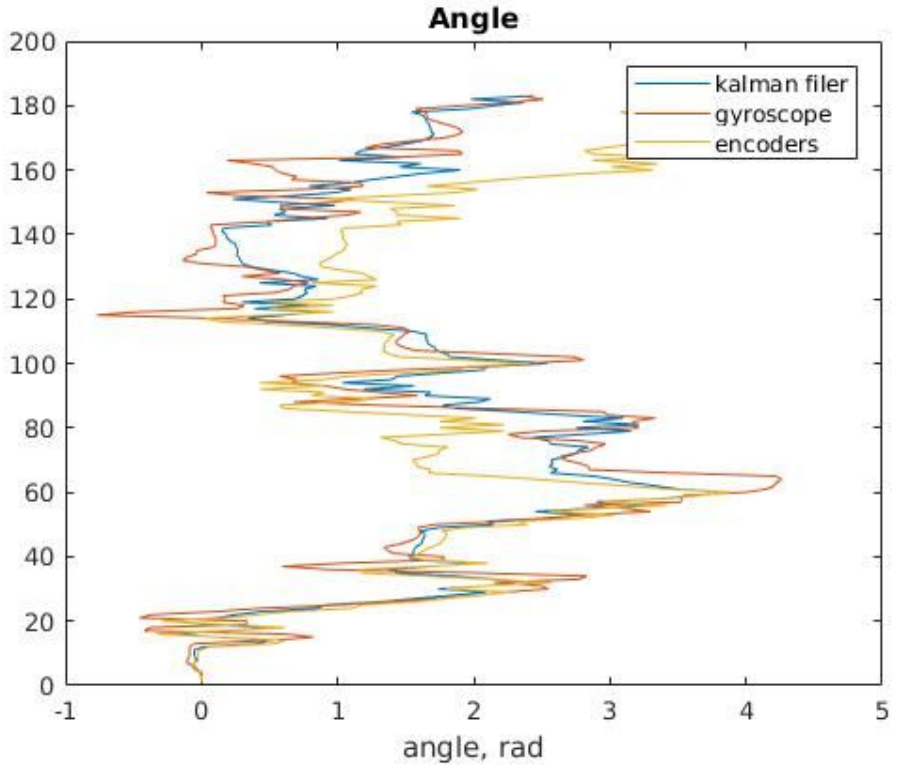

Fig. 5 Robot angle from gyroscope, encoders and kalman filter

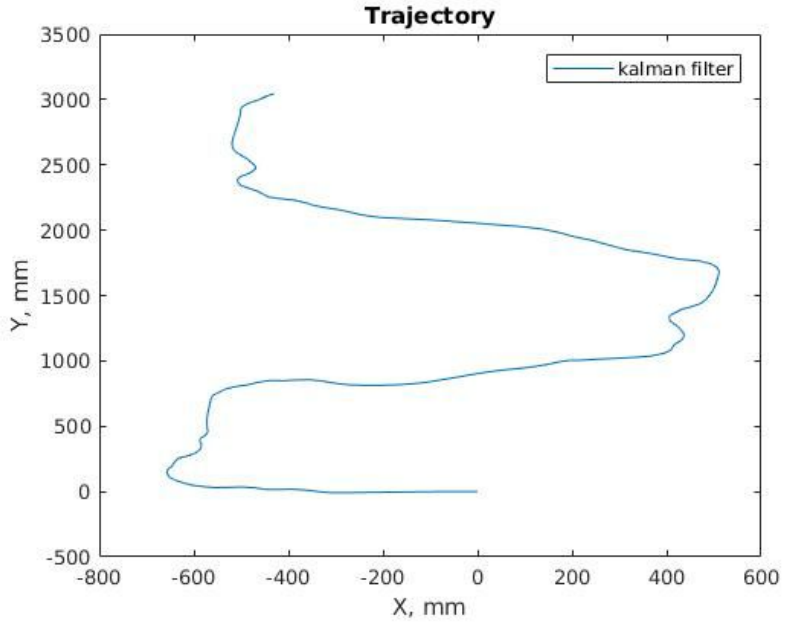

Fig. 6 Odometry from EKF

Map building

For map creation, firstly, i need to transform frame from sonar data to global frame. I use this formulas for it.

$$x_R = x_0 + dist_R \cdot \cos\left(\theta - \pi/2\right)$$
$$y_R = y_0 + dist_R \cdot \cos\left(\theta - \pi/2\right)$$
$$x_L = x_0 - dist_L \cdot \cos\left(\theta + \pi/2\right)$$
$$y_L = y_0 - dist_L \cdot \cos\left(\theta + \pi/2\right)$$

Fig. 7 Equations for wall points

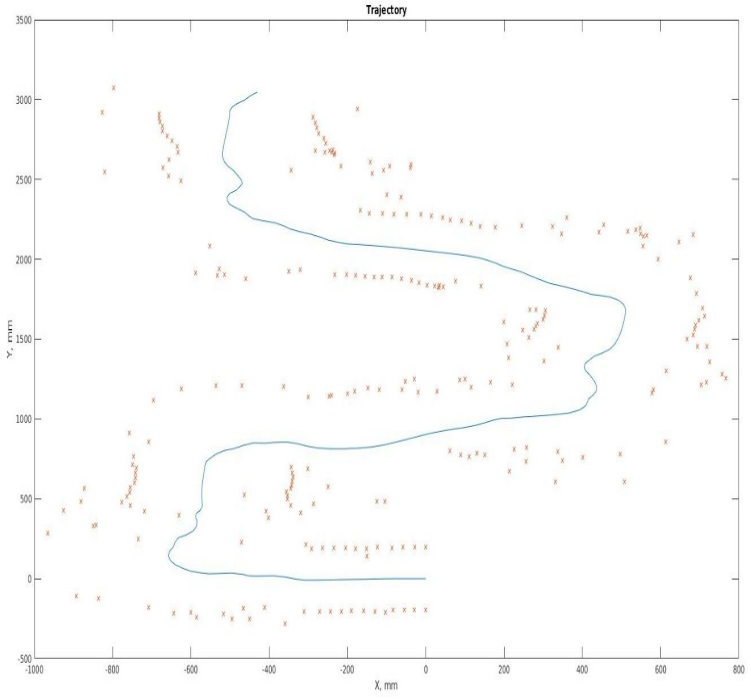

Fig. 8 Trajectory and wall points from sensors

But it isn't a map, it just point cloud. So we studied Hough transform and it was decided to use it.

Hough transform

But it isn’t a map, it just point cloud. So we studied Hough transform and it was decided to use it.

Firstly, i transform our point cloud in binary image (point cloud needs to be shifted in positive part).

For the best result of hough algorithm several params need to be tuned:

1) Amount of houghpeaks.
2) Minimum length between to points.
3) Distance between neighbor points.

At the result map was builded.

Fig. 9 Map with lines extracted by Hough transform method