1804336

Shinskiy MAXIM  CE315

Assignment 1

Table of Contents

[Introduction 2](#_Toc63880552)

[Underwater 2](#_Toc63880553)

[Land 2](#_Toc63880554)

[Air 3](#_Toc63880555)

[Task 1 3](#_Toc63880556)

[Task 2 4](#_Toc63880557)

[Task 3 5](#_Toc63880558)

[Appendix 6](#_Toc63880559)

[Links 6](#_Toc63880560)

[Task 1 6](#_Toc63880561)

[Task 2 6](#_Toc63880562)

[Task 3 7](#_Toc63880563)

# Introduction

Since the very first use of word “robot” back in 1920, the industry has been developing with exponential pace. Although, the foundation and definitions haven’t changed, scientists and researchers don’t stop looking for a new way of embedding computer systems to achieve industrial work optimization (e.g. Puma robotic arm[1]), land exploration (e.g. Curiosity Mars rover[2]) or just specific task based robotic system research (e.g. any DARPA Grand Challenge[3] contender). But no matter the complexity of the structure or architecture, all robots come from basics.

One of the fields of robotics is mobile robotics, which is intended to replace people from hostile environment and autonomously complete dynamic tasks using its adaptive to real-world problems behaviour. Mobile robots are built on the idea of travelling through specific median, that may differ from robot to robot. These medians are water, land, and air. Each one requires its own and unique way of moving and usually one robot may only travel through one terrain.

## Underwater

Let’s build the our upwards and therefore start from under the water. This environment is the least popular for traveling because of the difficulties that always come along. Mainly, the common feature between all robots is that most of them use radio waves to communicate but in this case, water blocks most of the signals making it very hard to control the system. However, there are many examples of robots that work autonomously and do not require constant control. These marine robots are usually used to study and monitor underwater living beings. A good example of such would be Imotus Hovering Autonomous Underwater Vehicle (HAUV) by Cellula[4], it uses SLAM algorithms to enable navigation, station keeping, route following etc. or uses dead-reckoning in open-water. It uses vectored thrusters for moving. Other field of use of AUV is for military purpose, like Pluto Plus AUV by Idrobotica[5] that is used for mine identification and deconstruction. Nevertheless, marine robotics’ are limited in terms of moving. There are either thrusters installed or the robot is based on an idea of fish-like motion, this narrows down the number of possible applications.

## Land

Land is the most common environment for human and most explored. Land robots are as different as they only could be. The one thing that differs them the most is the locomotion. Looking at ourselves and the nature we can state that most of the creatures are legged and this would be the obvious type of locomotion, however the most difficult. The problems come when we need to balance everything in the terms of mass and centre of gravity. The best example of legged robots is Atlas by Boston Dynamics[6]. It is the most advanced two-leg robot at this time, although even for Atlas simple motion task like walking up the stairs may be complicated depending on the terrain. When it comes to less complex robots, most motion tasks like moving objects around or running become impossible. One approach may be to increase the number of, so called legs, and in this case multi-legged robots’ (three or more) system becomes static and stable and therefore much easier to maintain the balance. Now, the centre of gravity lays in the triangle of three points of contact with ground. Spot is an example of a multi-legged robot from Boston Dynamics[7]. This design increases travelling potential but brings greater complexity and lower pace. Unfortunately, legged robots haven’t found its niche in the industry yet: they are complex, expensive, and mostly unreliable. The traditional wheeled robots show that they won’t be substituted at least in this decade.

Wheeled or tracked locomotion makes it easy to follow a precise route using well-known dead reckoning technique. To provide a robot with 6 degree of freedom it only requires 3 drive and steer wheels, making it cheap and easy to control. The perfect use of which is Roomba by iRobot[8]. Roomba is an autonomous vacuum cleaner that is able to navigate around the house and perform tasks without constant control. Tracked vehicles are used in the terrain where it requires high passability in forests, sands or marshlands. Generally, they come with a gripper on board to perform its task, the large area of contact provides enough sustainability to place an object on top in a dynamic and dangerous environment. Such tracked robot was used in Afghanistan and Iraq to defuse mines and bombs, its name is PackBot by FLIR Systems[9]. It also was the first robot to enter crippled Fukushima nuclear plant after Japan’s earthquake and tsunami in 2011.

## Air

Travelling via air is still relatively new to humanity and has many new researches and discoveries to this day. The most used design in air robots is a quadrotor. It has two pairs of counter-rotating rotors located at the vertex of a square frame. These drones are getting more widely used every day. The most suitable field for them is cinematography for today and quadrotor are the sole leaders in compare to other air robots in that. Matrice 600 by DJI[10] is widely used to film movies with a maximum take-off weight of 15.5kg, which is not that impressive but it provides camera stabilization and relatively long time of flight (38 minutes) making it suitable for aerial photography and some industrial applications. Drones can also be connected in a network to form a multi-drone system to use for disaster assistance, search and rescue and more. However, drones are not the only kind of Unmanned Aerial Vehicle (UAV) in mobile robotics. The more common to people way, planes, do have several advantages over drones, such as longer time of flight and less noise. They are rarely used in industry due to this kind of UAV being not able to carry a big payload, therefore the use of it would be area exploration, security and scouting.

# Task 1

Diagram

Description automatically generated The first task of the assignment was to complete a function for odometry calculation. Function takes two arguments which are velocities for left and right motors in cm/s­. The starting position of the robot is (30, 30) and pi/4 heading. Inside the function there is a for-loop to calculate the next coordinate of robot using current velocities and header of the robot.

# Task 2

The goal of task 2 in the assignment was to use a function from task 1 to plot the trajectory for different velocities. First set of velocities were 10 cm/s in left motor and 8 cm/s in right. This set is shown on the upper-left part of the graph as a larger circle. The robot started from 30, 30 and heading pi/4. It went in a clockwise direction in a circular path because the speed of left track was greater than right track. The other set has left motor speed of 5 cm/s and 7 cm/s for right motor. In the second trajectory, unlike in first, velocity of right track is higher and therefore robot went in counter-clockwise direction. Second trajectory is represented as a smaller circle in the lower-right part of a graph. Circles intersect in 30, 30 because the share the starting position that is left unchanged between each run.

# Task 3

Diagram

Description automatically generated

A plot of an environment map is graph of a series of points on one chart. To plot such chart, we need to generate these coordinates and write them to file, we use programming to make it efficient. In my program I first generate coordinates of pillars for no specific reason. For this, I used two array each containing x or y coordinates of a centre of each pillar in the same order. Then, I used a for loop to iterate through these arrays and another for loop inside to calculate the coordinates of the circumference of pillars and a charger with a step of 20 degrees and write it to file, therefore, it generated 18 points for each obstacle. To generate walls, I used two for loop that to write 4 straight lines into the same file.

# Appendix

## Links

[1]: <https://www.bondsimulation.com/index.php/example/134-puma-560>

[2]: <https://mars.nasa.gov/msl/home/>

[3]: <https://www.darpa.mil/about-us/timeline/-grand-challenge-for-autonomous-vehicles>

[4]: <https://www.cellula.com/imotus-1?gclid=Cj0KCQiAgomBBhDXARIsAFNyUqM2C_34XeKfpgSURxF05xezYO9XZSN1DaqBr-KY__zxMBKxw0O8wWwaAlVcEALw_wcB>

[5]: <https://military.wikia.org/wiki/Pluto_Plus>

[6]: <https://www.bostondynamics.com/atlas>

[7]: <https://www.bostondynamics.com/spot>

[8]: <https://www.irobot.com/roomba>

[9]: <https://robots.ieee.org/robots/packbot/>

[10]: <https://www.dji.com/uk/matrice600-pro?site=brandsite&from=nav>

## Task 1

#define SIZE 200

int wheelbase **=** 30**,** delta\_t **=** 1**;**

double rob\_x**[**SIZE**]** **=** **{**30**},** rob\_y**[**SIZE**]** **=** **{**30**},** rob\_theta**[**SIZE**]** **=** **{**PI**/**4**};**

//Method for calculating the trajectory

int robot\_kinematics**(**double left\_vel**,** double right\_vel**)**

**{**

//Loop to calculate a position(x,y) of a robot at every time unit(SIZE)

**for(**int i**=**0**;** i **<** SIZE**;** i**++)**

**{**

rob\_x**[**i**+**1**]** **=** rob\_x**[**i**]** **+** **(**left\_vel **+** right\_vel**) /** 2 **\*** cos**(**rob\_theta**[**i**])\***delta\_t**;**

rob\_y**[**i**+**1**]** **=** rob\_y**[**i**]** **+** **(**left\_vel **+** right\_vel**) /** 2 **\*** sin**(**rob\_theta**[**i**])\***delta\_t**;**

rob\_theta**[**i**+**1**]** **=** rob\_theta**[**i**]** **+** **(**left\_vel **-** right\_vel**) /** wheelbase**\***delta\_t**;**

**}**

//return 0 for SUCCESS

**return** 0**;**

**}**

## Task 2

#include<math.h>

#include<stdio.h>

//define constants

#define SIZE 200

#define PI 3.14159265

FILE **\***fp**;**

//Method for calculating the trajectory

int robot\_kinematics**(**double left\_vel**,** double right\_vel**)**

**{**

//initialize variables

int wheelbase **=** 30**,** delta\_t **=** 1**;**

double rob\_x**[**SIZE**]** **=** **{**30**},** rob\_y**[**SIZE**]** **=** **{**30**},** rob\_theta**[**SIZE**]** **=** **{**PI**/**4**};**

//Loop to calculate a position(x,y) of a robot at every time unit(SIZE)

**for(**int i**=**0**;** i **<** SIZE**;** i**++)**

**{**

rob\_x**[**i**+**1**]=**rob\_x**[**i**]+(**left\_vel**+**right\_vel**)/**2**\***cos**(**rob\_theta**[**i**])\***delta\_t**;**

rob\_y**[**i**+**1**]=**rob\_y**[**i**]+(**left\_vel**+**right\_vel**)/**2**\***sin**(**rob\_theta**[**i**])\***delta\_t**;**

rob\_theta**[**i**+**1**]=**rob\_theta**[**i**]+(**left\_vel**-**right\_vel**)/**wheelbase**\***delta\_t**;**

**}**

//write the results of the calculations to the file

fprintf**(**fp**,** "%d, %d \n"**,** 30**,** 30**);** //start at x = 30, y = 30

**for(**int i **=** 1**;** i **<** SIZE**;** i**++){**

//write to columns (x y)

fprintf**(**fp**,** "%lf, %lf \n"**,** rob\_x**[**i**],** rob\_y**[**i**]);**

**}**

//return 0 for SUCCESS

**return** 0**;**

**}**

//main method

int main**(){**

//open a file to write to

fp **=** fopen**(**"odometry\_data"**,** "w"**);**

//run the calculations for Vl = 10, Vr = 8

robot\_kinematics**(**10**,** 8**);**

//run the calculations for Vl = 5, Vr = 7

robot\_kinematics**(**5**,** 7**);**

//close stream

fclose**(**fp**);**

//return 0 for SUCCESS

**return** 0**;**

**}**

## Task 3

#include <stdio.h>

#include <math.h>

//define constant

#define PI 3.14159265

//initialize variables

FILE **\***map**;**

double x**[**5**]** **=** **{**0.8**,** 0.8**,** 1.7**,** 2.2**,** 2.25**};**

double y**[**5**]** **=** **{**1.5**,** 2.25**,** 1.0**,** 1.3**,** 0.25**};**

double r**=**0.125**,** wallLength **=** 2.5**;**

double RD **=** PI**/**180**;**

int build\_environment\_map**(){**

map **=** fopen**(**"evironment\_map.txt"**,** "w"**);**

//pillars and a charger

//repeat for each index in the x and y arrays

**for(**int i **=** 0**;** i **<** 5**;** i**++){**

//write into file to plot circles afterwards

**for(**int j **=** 0**;** j **<** 360**;** j **+=** 20**){**

fprintf**(**map**,** "%lf, %lf \n"**,** x**[**i**]+**r**\***cos**(**j**\***RD**),** y**[**i**]+**r**\***sin**(**j**\***RD**));**

**}**

**}**

//walls

**for(**double i **=** 0**;** i **<** wallLength**;**i **+=** 0.01**){**

fprintf**(**map**,** "%lf, %d \n"**,** i**,** 0**);**

fprintf**(**map**,** "%d, %lf \n"**,** 0**,** i**);**

**}**

**for(**double i **=** 0**;** i **<** wallLength**;**i **+=** 0.01**){**

fprintf**(**map**,** "%lf, %lf \n"**,** wallLength**,** i**);**

fprintf**(**map**,** "%lf, %lf \n"**,** i**,** wallLength**);**

**}**

//close stream

fclose**(**map**);**

//return 0 for SUCCESS

**return** 0**;**

**}**

int main**(){**

build\_environment\_map**();**

//return 0 for SUCCESS

**return** 0**;**

**}**