Course Project Report

# Centralized Navigation for Multiple Mobile Robots

Student Name: TANG Wenbo

Student ID: 1155098538

# 1. Introduction

Nowadays, multiple mobile robots for cargo transportation in warehouse are more and more popular. However, these robots would be a chaos if they are controlled and working independently, because they don't know where other robots are. Therefore, it is necessary to design a centralized navigation system for these robots, which uses a server to collect every robot's sensor data and localize them, such that making appropriate decisions based on the total distribution of robots.

In this project, every mobile robot is defined as a client, which has an IMU sensor measuring the velocity of the robot and range sensor which collects the distances and angles from artificial landmarks. The client directly sends all the sensor data to the server and receive a velocity control command to the motors from the server (linear and angular velocity).

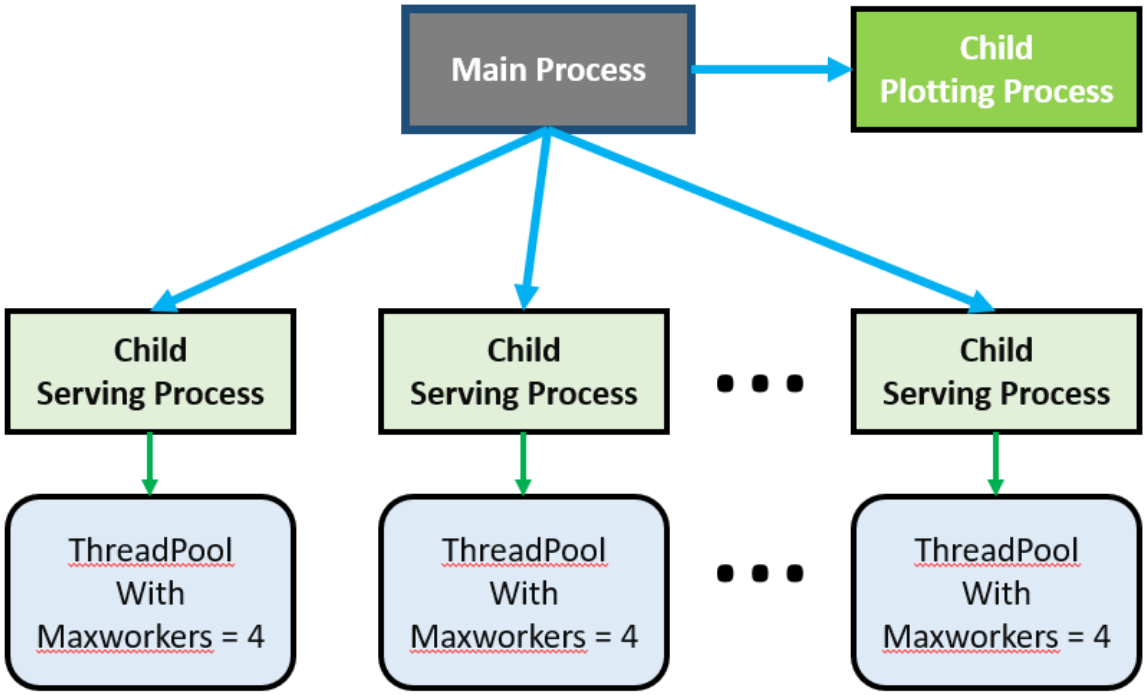
On the server side, it stores the most recent client location and received raw sensor data of every client. Subsequently, the clients will be processed using multiprocessing and multithreading technique. The first function of the server is to localize clients using Extended Kalman Filter (EKF). Given the task defined in the server, the clients will have corresponding velocity control command using the motion controller designed in the server. At last, the server sends the control command back to clients until the task have been completed.

Since time is limited for this project, the task defined in the server is simply tracking a pre-designed path until the end. Besides, for simplicity, the motion controller generates commands for every client independently rather than considering all the existing robots at the same time. However, all the existing clients will still be observed at the server side.

The features of this project are as follows: 1. Extremely small computation resources are needed for a client; 2. The server uses multiple processes and threads for concurrent computation; 3. All the clients can be observed, localized, planned and controlled using one server, which will be easy for extension of more sophisticated tasks.

## 2. Networking Technology

The server designed in this project is a pre-fork TCP server. The diagram below illustrates the server architecture:

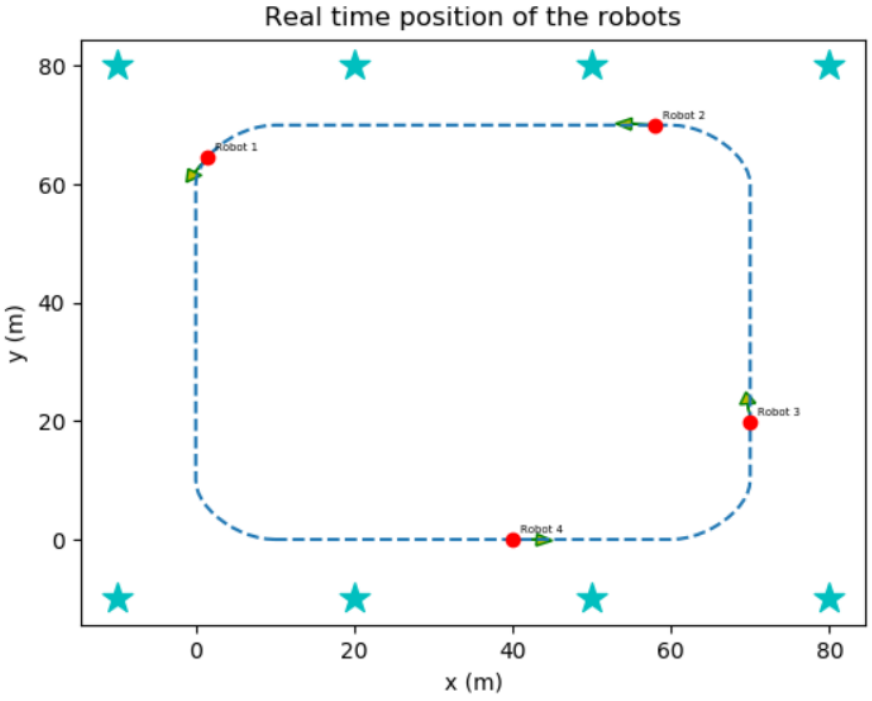



The Main Process is responsible for listening for client connections and passing client sockets and client addresses to Child Serving Process through a queue when new clients are connected. Each Child Serving Process has a ThreadPool with a maximum of 4 workers.

The client will send its sensor data to the server. The thread receiving the sensor data will perform the following:

- Build up a tracking trajectory for the client
- Localize the client using EKF
- Use a Motion Controller to generate velocity control command
- Send the localization and control command back to the client
- Repeat the data receiving and sending loop with a certain frequency (10Hz here)
- If the trajectory is finished, send a success flag and close the connection.

Besides the Child Serving Process, there is another Child Plotting Process used to visualize all the current mobile robots at the same time. All the processes are sharing a same dictionary object where each client registered in. When a new client is connected, the corresponding will continuously update a specific item and delete this item when the client is disconnected. The whole plot shows the artificial landmarks at the boundary of the plot used for range sensor measurements (distributed stars), a pre-defined tracking trajectory (dashed line) and all the robot poses (circle with an arrow around).



## 3. Libraries and Modules

**Used Liabraries:**

- **json**

  Used in TCP data transmission

- **matplotlib**

  Used for visualization

- **numpy**

  Used for matrix calculation

- **scipy**

  Used for path generation

- **multiprocessing**

  Used for creating multiple processes

- **threadpool**

  Used for creating threadpool with pre-forked workers

- **math**

  Used for normal mathematics

- **socket**

  Used for building up TCP/IP connection

- **sys**

  Used to accept command line parameters

- **random**

Used to simulate noisy sensor measurement

**Self-Created Modules:**

- **commontools**

  Some common mathematical functions

- **initialize** (2 classes)

  **class Params:**

  Initialization of simulation parameters including sensor noise, simulated time interval (controls the data transmission frequency) and controller parameters.

  **class RobotMap:**

  Build up all the map elements, including the position of artificial landmarks for range sensor measurements and tracking Bezier trajectories.

- **kalmanfilter** (1 class)

  **class EKF:**

  A python implementation of Extended Kalman Filter, including motion model prediction, sensor model correction and posterior extraction.

- **motioncontroller** (1 class)

  **class MotionController:**

  A python implementation of pure pursuit path tracking algorithm including local target identification, PID controller and command extraction.

- **robot** (1 class)

**class Robot:**

One class that considers all the robot parameters and functions, including robot state, velocity, index, laser sensing function, odometry measurement function and motion update function.

## 4. Potential Improvement

Since the time is limited for the project, some conceptual functions of the server are not implemented yet. If I have more time, I will improve the completeness of the server with the following functions:

- Independent Tracking Trajectory Generation

  For each client, the server will correspondingly design appropriate tracking trajectory rather than use the same trajectory for all clients.

- Multiple Robots Motion Control

  To avoid collision or achieve multiple robots' cooperation, the motion controller should consider all the robots at the same time. This will also need well-designed simulated tasks to better implement this.

In addition, I will add up more data transmission methods to facilitate different tasks, like producer and consumer model. In this case, tasks will be defined as a queue, the server will assign these tasks to available mobile robots.

Furthermore, the publisher and subscriber model may be a better method to uniformly control all the robots at the same time when it's necessary.