EINDHOVEN
UNIVERSITY OF
TECHNOLOGY

Design Document

4SC020 - Embedded Motion Control 2019

Group 5

Group Members:

Muliang Du	1279874
Shubham Ghatge	1316982
Winston Mendonca	1369237
Yi Qin	1328441
Robert Rompelberg	0905720

Group Tutor:
Hao Liang Chen

Lecturer:
Prof. Dr. René van de Molengraft

Dept. of Mechanical Engineering
Eindhoven University of Technology

May 20, 2019

1 Introduction

The aim of the project is to program a PICO robot in such a way, that it is able to pick-up medicine and deliver this to patients without hitting walls, people or other obstacles. Since the PICO has no arms, the picking-up will be standing in front of a cabinet and speaking. Providing the medicine will be done the same way.

2 Requirements

The following performance requirements are expected to be met to complete for the Hospital challenge:

1. PICO should visit the cabinets in a provided order.
2. PICO should give a sound indicating that it has arrived at a certain cabinet.
3. PICO should not bump into walls or hit moving obstacles.

To fulfill the above requirements, several functional requirements have to be met:

- PICO has to be able to determine its starting location on the provided world map
- PICO has to be able to detect walls
- PICO has to be able to find the exit and hence entry of a room
- PICO has to be able to plan a trajectory from one cabinet to the next one
- PICO has to be able to update its world map, to anticipate on open or closed doors
- PICO has to be able to determine its position
- PICO has to be able to face the cabinet and tells it number
- PICO has to be able to emergency brake if a by passer walks by

3 Functions

The tasks that PICO has to perform can be modularised into the following functions:

1. Configuration: Before PICO can begin navigating its environment, its sensors and actuators need to be appropriately initialized and ready to receive instructions or transmit valid sensor data. Any additional variable/data object memory allocations will also take place during the configuration phase. This phase would typically run only when PICO is first powered on.

2. Sensing: PICO's sensors are constantly analysing its environment in terms of its distance to nearby objects, as well as the states of its actuators in terms of the amount translational and rotational motion, control effort for actuation, etc. This data is saved into a "World Model" so that subsequent functions can access these sensor readings for further processing.

3. Environment Mapping: PICO needs to be capable of recognizing and remembering its environment as it progresses through its tasks. This will give it the ability to "know" its position within its operating environment, by recognizing and distinguishing between walls, corners, exit/entry points, etc., and on a higher level, to even identify how to return to a room that it has already visited. In situations where no map is provided before hand, the Environment Mapping function will create an environment map of PICO's surroundings in the World Model using the data previously recorded in the World Model set, and this map could then be used by any path planning or motion output functions.

4. Trajectory Planning: With the generated map and sensor data, PICO can evaluate the control action it needs to take to achieve its current objective. For example, in the Escape Room challenge, this could be the trajectory it would have to traverse to reach the exit of the room. Trajectory planning also ensures that the planned traversal path keeps PICO from bumping into any walls or obstacles.

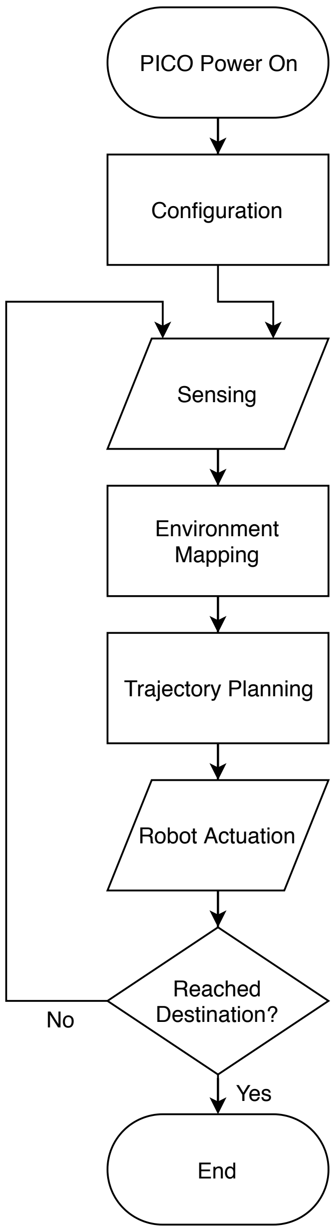

Figure 1: Function Flowchart

5. Robot Actuation: This function will handle all tasks pertaining to PICO's motion. Using the holonomic base, this function will determine the type of motion (translational, rotational or a combination of both) PICO has to undertake to follow the provided path from the Trajectory Planning function. A higher implementation of this function would also include actuating PICOs motors with smooth acceleration profiles to avoid uncontrolled jerks in motion.

3.1 Order

First of all the PICO has to configure itself. Next PICO should determine its starting position. However, since the starting position of the PICO is roughly given on forehand, not too much effort should be put into this. Once the PICO is configured, it should calculate its trajectory based on the provided world map. This trajectory is send to the actuators, which ensures the robot starts to move. Once PICO start to move, the number of wheel rotations will be stored. During the movement continuous sensing data is acquired. This data is thrown away generally. Once the rangefinder finds a distance lower than a given threshold, an obstacle is observed and the PICO stops, updating its world map. This updating is done by using the wheel rotation data. **This should of course be properly tested, to see the effect of wheel spinning!**Once its position is updated, the scanned obstacle will be added to the world map, if and only if the obstacle remains on the same spot for a least 1 second. Otherwise it is a dynamic obstacle. If the distance to the obstacle decreases while PICO stands still, it will move away perpendicular by a certain value. After updating the world map a new trajectory is calculated and send to the actuators.
Once PICO is in the area which indicates a cabinet, its distance to the cabinet should be lower than a certain value. If this is not the case, PICO has to reoriented itself, determine its position and update the world map. This is the hardest part of the code. After the update PICO can again calculate a trajectory and continue its journey. Once it actual reaches the cabinet, it should allign to the cabinet. This is done by picking several data points in front of the PICO, so for example from 10 degrees to minus 10 degrees. If the distance is the same, with respect to a certain threshold, for this range, the PICO is aligned facing the cabinet. Next PICO has to say it has arrived to this cabinet and continue its journey afterwards.

4 Components

PICO Telepresence Robot by Aldebaran:

1. Sensors:
   (a) Proximity measurement with Laser Range Finder (LRF)
   (b) Motion measurement with Wheel encoders (Odometer)
   (c) Control Effort Sensor
2. Actuators:
   (a) Holonomic Base - Omni wheels that facilitate 2D translation and rotation
3. Computer:
   (a) Intel i7 Processor
   (b) OS: Ubuntu 16.04

5 Specifications

1. Laser Range Finder (LRF)

   (a) Linear Range: 0.1 to 10 meters
   (b) Panoramic range: -2 to 2 radians
   (c) Panoramic Resolution: 0.004004 radians

2. Holonomic Base:
   (a) Maximum Linear speed: 0.5 m/s
   (b) Maximum Rotational speed: 1.2 rad/s

6 Interfaces

1. Communication with PICO: All software uploads to the PICO robot will take place through GitLab.
2. Communication within software functions: As described in the Functions section, it is necessary to exchange information from different functions. The main interface between the functions stated above will be the World Model which will contain the raw data recorded by the Sensing function, the processed information from the Environment Mapping and Trajectory Planning functions and the planned trajectories for the Locomotion function.