Group 6

4SC020 Embedded Motion Control

Design Document

Emre Deniz
0967631
e.deniz1@student.tue.nl

Joep Selten
0988169
j.stelten@student.tue.nl

Aris van Ieperen
0898423
a.i.v.ieperen@student.tue.nl

Stan van Boheemen
0958907
s.p.v.boheemen@student.tue.nl

Bram Schroeders
1389378
b.j.m.schroeders@student.tue.nl

Pim Scheers
0906764
p.scheers@student.tue.nl

Teacher:
dr.ir. M.J.G. (René) van de Molengraft

Eindhoven, May 4, 2020

1 Requirements and specifications

The requirements and specifications of the PICO robot for the escape room competition and the hospital challenge are listed below. Here, a distinction is made between customer-, border- and system requirements, where the customer requirements include the goals and preferences defined by the customer, and the border- and system requirements are more specific requirements to achieve those goals.

- Be able to autonomously exit the escape room as fast as possible from any (unknown) initial position through the designated exit within 5 minutes.
  Several cabinets must be visited in a random order, which is only known when the challenge is started.
  - Detection of the robot its surroundings and target objectives.
    - Within its field of view, the robot must create a 2D image (top view) with the current relative distances to any solids such as walls and objects with a 5mm accuracy.
    - Consequently, the detected objects should be categorised into 'avoid' or 'target'.
    - The robot must be able to know its position and direction.
    - The progress towards the target objective must be known.
  - Efficient navigation and movement towards the target objectives.
    - The robot must be able to scan and map the environment in an efficient way, preventing going over the same area twice.
    - Identification of the shortest distance path between current position and target position.
    - Swift execution of desired path.
  - Exiting the escape room counts when the rear wheels of the robot are further than 3m into the exit corridor.
  - A visit to a cabinet is defined as the robot begin positioned within the square region marked in front of the cabinets, while facing towards it, giving a clear sound signal. The cabinets are represented by a rectangular block with their width equal to the size of the target square.
- The escape room is roughly rectangular shaped, with unknown dimensions or perpendicularity. The exit is between 0.5 and 1.5 metres and perpendicular to the wall it starts from, with an open end. The hallways in the hospital are approximately 1.5 meters wide. Walls are approximately straight and 20cm thick. Corners in walls and hallways will be close to perpendicular. Doors are time-invariant openings in walls of 0.5 to 1m wide that may be closed or open. An unknown amount of dynamic and static objects will be placed throughout the hospital.
  - The detection must be robust to imperfections in the shape of the walls, exit and the random objects littered around the hospital.
    - No assumptions should be laid on the relation between walls and the location and orientation of objects, in the sense of parallelism, perpendicularity or straightness. All exceptions are specified in this document.
  - The behaviour of dynamic objects must be predicted.
    - The position and velocity of dynamic objects must be measured and extrapolated.
    - The path of the robot must be adjusted to efficiently prevent collisions, while keeping lost time to a minimum.
- The robot must be safe at all times.
  - The walls and other objects may not be bumped into.
    - A safe clearance of 50mm must be maintained to any stationary object and 200mm to any dynamic object.
    - The angle between the driving direction and the robot its x-axis is limited to ± 1 rad.
    - The robot must not only rely on its position measurement to avoid collisions. Any relative measurements of distances to objects must be used.
  - The maximum stopping distance must be less than 0.5m.
    - The velocity is limited to 1m/s.
    - The maximum response time between emergency detection and braking should be 100ms.
  - The robot should detect when a certain component fails and act accordingly.

    - In the case of a software crash, the robot must know which part of the software caused the crash.
    - When faced with implausible sensor inputs, the robot should omit their data until a plausibility check is passed.
    - If the actuator fails, the robot should detect that no movement is observed.
  - In the case of unforeseen disturbances, i.e. crashes or sensor failure, the robot must remain functional and safe.
    - Any crashed software part should be rebootable within 25s.
    - The robot may not drive when essential parts of the software are dysfunctional.
    - In the case of sensor or actuator failure, the robot should remain stationary until all is functional again.

- Working with the robot should be accessible to everyone.
  - The user interface must be easy to use, quick to use and foolproof.
    - The existence of a main executable which can be called, which then connects all underlying components.
    - The software must be able to be updated with a single command.
    - Sending an unknown command should prompt an error message with all possible commands.

PICO specifications

Dimensions	$(L \times W \times H) = 1005 \times 410 \times 350$ mm
	Weight = 8 kg
Omni-wheels: Holonomic base	Drive in forward and backward direction with a maximum velocity of 0.5 m/s.
	Rotate about its axis with an angular velocity of 1.2 rad/s.
Laser Range Finder	Detect objects in a field of view ranging from -2 to 2 rad with an increment of 0.004 rad.
	Determine the distance of the objects relative to its own position.
Wheel encoders	Determine travelled distances.
Audio-ouput	Give sound messages.
Computer operating Ubuntu 16.04	Execute the software.

2 Components and functions

In order to have good structured software, it is composed of several components. From which each component consists of functions.

- Perception: Retrieves LRF and odometry data and processes it. This processing consists of filtering unuseful data (e.g. high frequent noise), locating pieces of the environment (e.g. cabinets, walls and gaps in wall) and locating its current position. This processed data is then used to update the world model.
- World Model: Maps and stores the environment using current and new data, which is stored in a data structure. Other components can retrieve this information using functions (e.g. minimum distance to wall, distance around obstacle, distance to objective).
- Monitor: Maps the current situation in discrete states (e.g. at exit or not, near wall or not, at cabinet or not, exit found or not, cabinet found or not) using information of the environment. The monitor has a discrete perception of the system, which controls the state transitions.
- Strategy: Contains the finite state machine, which manages which action the robot should take using the discrete situation and its location. The state machine should also be capable of error handling. Its current state can be obtained using a `GetState()` function. Path planning and trajectory generation is incorporated within the corresponding state.
- Control: Actuates the robot to follow the desired actions received from the strategy. Localization information from the world model is received which acts as a feedback to prevent collisions with e.g. walls and obstacles.

3 Interfaces

The flow of information between the previously described components is presented in the block diagram below. It has to be noted that these are not the exact functions used, but they represent the desired functionality.

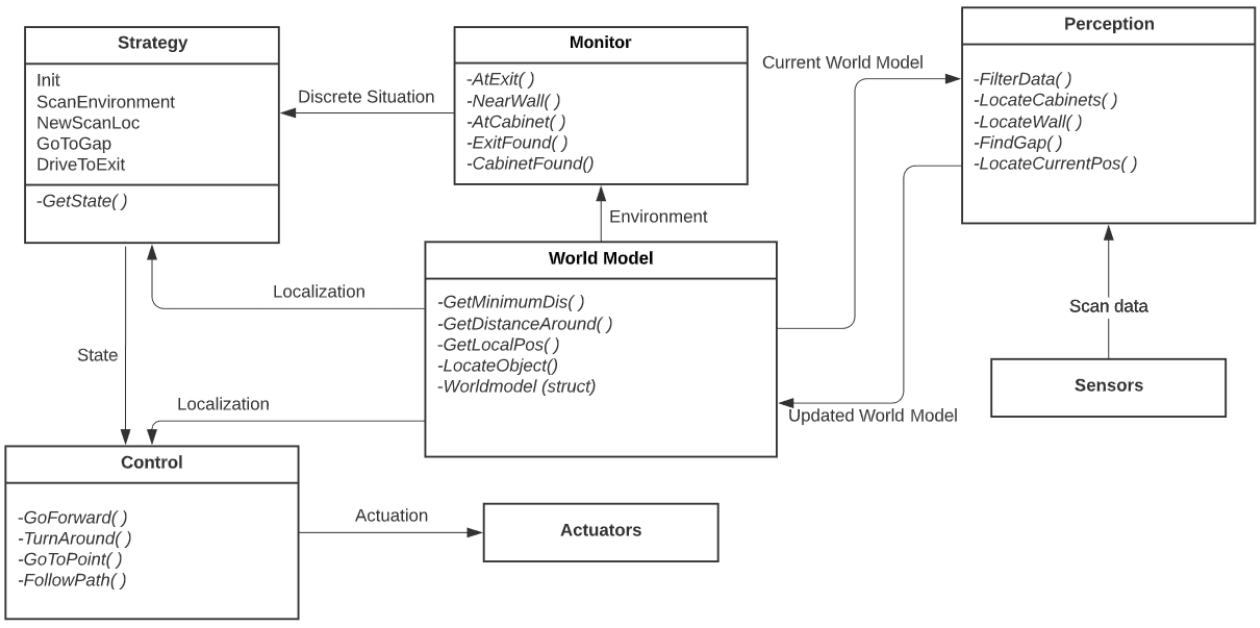

Figure 1: Information architecture