Embedded motion control initial design idea

Group 3

Outline

In this design outline the initial idea for the software design of the robot is discussed. First the rules of the maze challenge are presented. From this challenge the requirements for the robot are derived. These requirements are then translated to the functions, components, specifications and interfaces for the software design.

The maze challenge

The challenge is to design the software for a robot so it can find its way out of a maze. The dimensions and structure of the maze are unknown. The maze may also have doors that open automatically when to robot is standing in front of them after an unknown period of time. The software has to be applicable to multiple robots that all share the same basic functionalities, namely driving and using sensors such as its array of laser range finders. The objective is to have the robot find its way out of the maze as fast as possible. The low level controllers for the sensors and actuators are provided and do not have to be made by the student teams.

Requirements

Two requirements are devised based on the description of the maze challenge. The first one is to be able to complete the challenge by finding a way out of the maze. The second requirement is that the robot should avoid bumping into the walls or doors. While this second requirement is not necessary for the completion of the challenge, it is helpful for the longevity and operation of the robot.

Functions

The robot will know a number of basic functions. These functions can be divided into two categories: actions and skills. The actions are the most basic actions the robot will be able to do. The skills are specific sets of actions that accomplish a certain goal. The list of actions is as follows:

- Drive
- Turn
- Scan
- Wait

These actions are used for the skills. The list of skills is as follows:

- Drive to location

  Drive to a position in the maze based on the information in the map. This includes (multiple) drive and turn actions. Driving will also always include scanning to ensure there are no collisions.
- Check for doors

  Wait at the potential door location for a predetermined time period while scanning the distance to the potential door to check if it opens.

- Locate obstacles

  Measure the preset minimum safe distance from the walls or measure not moving as expected according to the driving action.
- Map the environment

  Store the relative positions of all discovered object and doors and incorporate them into a map.

These skills are then used in the higher order behaviors of the software. These are specified in the specifications section.

Specifications

The specifications are the objectives, that the software should be able to accomplish. They are the implementation of the requirements. The behaviors specified here are a set of the functions the robot will have.

The first specification results from the second requirement: Driving without bumping into objects. In order to do this the robot uses its sensors to scan the surroundings. It then adjusts its speed and direction to maintain a safe distance from the walls.

The way the robot will solve the maze comprises of a few principle things the robot will be able to do. Because of the addition of doors in the maze, the strategy of wall hugging is no longer effective. Hence a different approach is required. The second specification is that the robot will remember what it has seen of the maze and make a map accordingly. The robot should then use this map to decide which avenue it should explore.

The escape strategy of the robot is an algorithm. Because the doors in the maze might not be clearly distinguishable, they might be difficult to detect. The only way to know for sure if a door is present at a certain location is to stand still in front of the door until it opens. Standing still in front of every piece of wall in order to check for the presence of doors takes a long time and is therefore not desirable for escaping the maze as fast as possible. Therefore the robot first assumes that there are no doors in the way to the exit. It then explores by following the wall and taking every left turn. Whenever a dead end is hit, the robot goes back to the last intersection and chooses the next left path. Because the robot maps the maze it knows whether it has explored an area and when it moves in a loop. If no exit is found under the assumption that there are no doors, the robot starts checking for doors. At first it assumes that there can only be doors at dead ends (1). If still no solution is found the robot also checks for doors at corners (2), followed by intersections (3) and finally on every outside wall of the currently mapped maze. In order to detect these doors the robot stands in front of the potential door for a certain time and checks with its sensors whether the distance to the nearest wall changes.

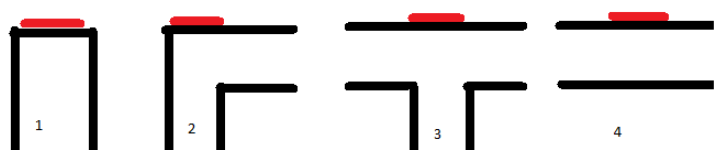

Figure 1: assumed door locations (red) for possible wall structures

Components and interfaces

The interaction between the different software components is very similar to the care robot example (figure 2). In this the Task control is the behaviors described in the specifications section, the skills and robot context are described in the functions section.

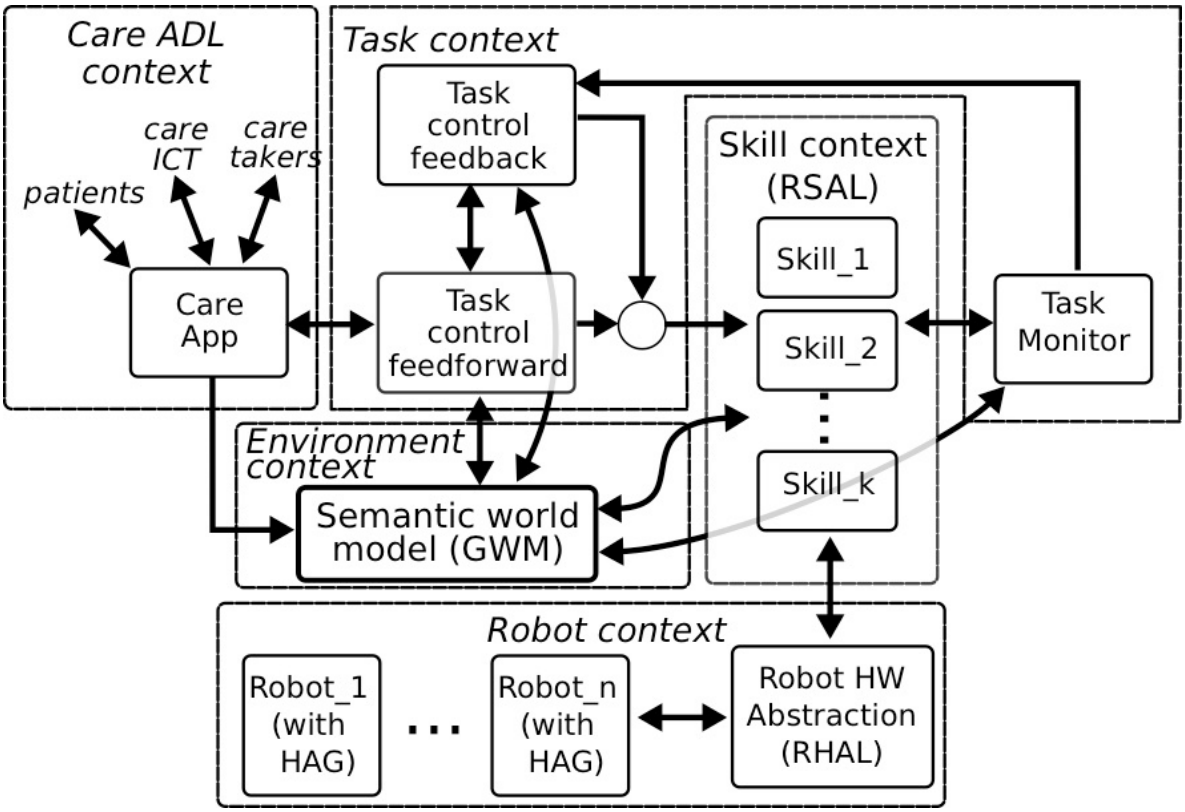

Figure 2 Interface for the care robot example

Uncertainties

Certain aspects of the design are not yet clear due to uncertainties in the specifications of the robot and/ or the maze challenge.

- Depending on the difference between the end of the maze and the inside of the maze, the robot may be enabled to detect it has completed the challenge. If however the outside of the maze is simply an open space similar to a place inside the maze, the robot might not be able to distinguish the difference. In this case the robot would have to be stopped manually.
- The exact specifications of the robot are still unknown and without testing the precise accuracy and range of the sensors, the resolution of the map and the safe wall distance are unknown.
- In order to make the robot complete the challenge faster control over the speed of the robot could be used. This way it could move faster in area's it has already mapped. This is only possible if the robot has the capability of moving at different speeds.