TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

4SC020 Embedded Motion Control – Final design

Group 4

Bosselaar, M.A. Grolleman, B. Setz, J. Sommer, R.L. Tibboel, M.

/department of Mechanical Engineering

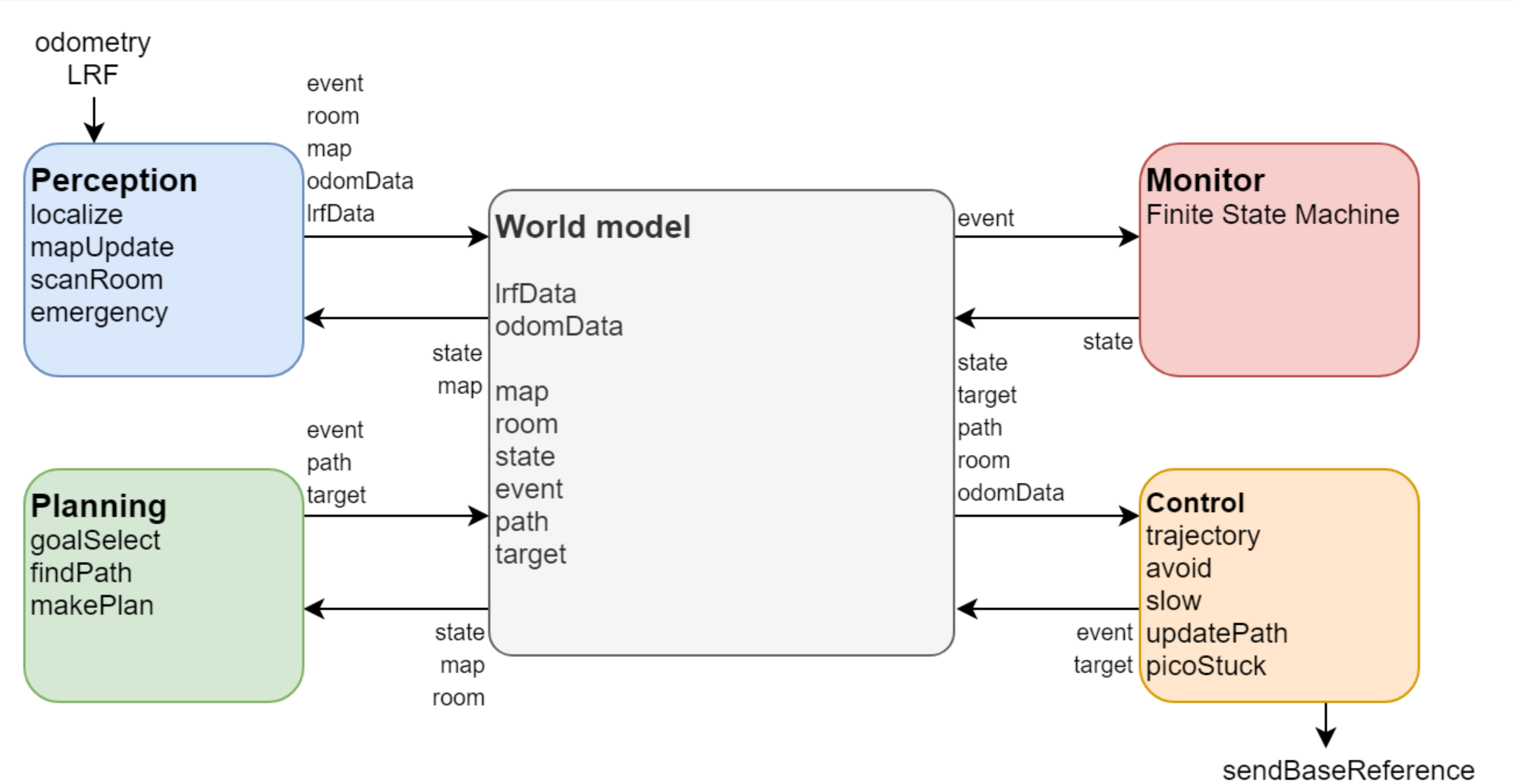
Contents

- Requirements
- Software Architecture
- Functions
  - World Model
  - State Machine
  - Localization
    - Assumptions
    - Alternative
  - Planning
  - Dynamic obstacle avoidance
  - Static obstacle avoidance
- Improvements

Requirements

- Software must be generic
  - Arbitrary number of cabinets
  - Arbitrary hospital map
  - Arbitrary number of cabinets to be visited
  - Arbitrary order of cabinets to be visited
  - Arbitrary start point

Software Architecture



odometry
LRF

Perception
localize
mapUpdate
scanRoom
emergency

event
room
map
odomData
lrfData

state
map

World model

lrfData
odomData

map
room
state
event
path
target

event

state

Monitor
Finite State Machine

state
target
path
room
odomData

Planning
goalSelect
findPath
makePlan

event
path
target

state
map
room

Control
trajectory
avoid
slow
updatePath
picoStuck

event
target

sendBaseReference

Functions – World Model

JSON Map

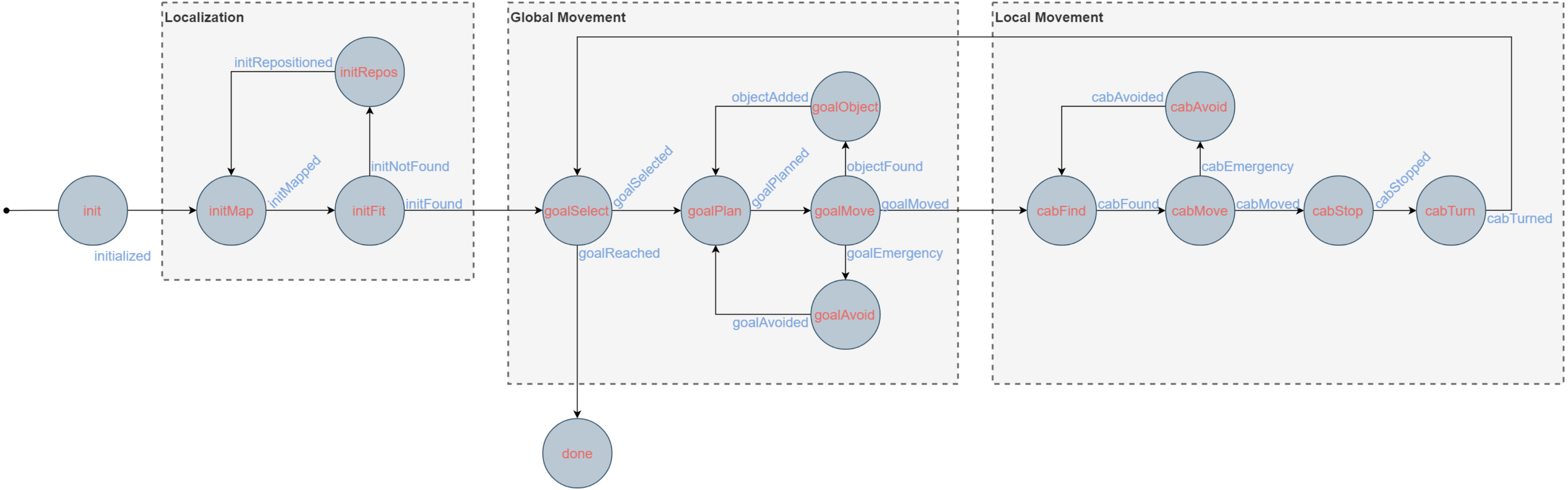
- Convert to mapping
  - Walls
    - Two points
  - Cabinets
    - Four walls
  - Corners
    - Two walls and a point
  - Obstacles
    - Walls

Functions – World Model

LRF Data

- Convert to room
  - Walls
    - Two points
  - Corners
    - Two walls
    - One point
    - Type
  - Doors
    - Two corners of certain type

Functions – State Machine



Functions – Localization

Fit map to LRF data

- Map preprocessing
- Orient PICO
- Match corners by range
- Move and rotate map

Functions – Localization

Assumptions

- No two rooms are the same
  - Multiple correct matches would be possible
- At least four map corners are visible
  - Four points required for non-ambiguous location

Functions – Localization

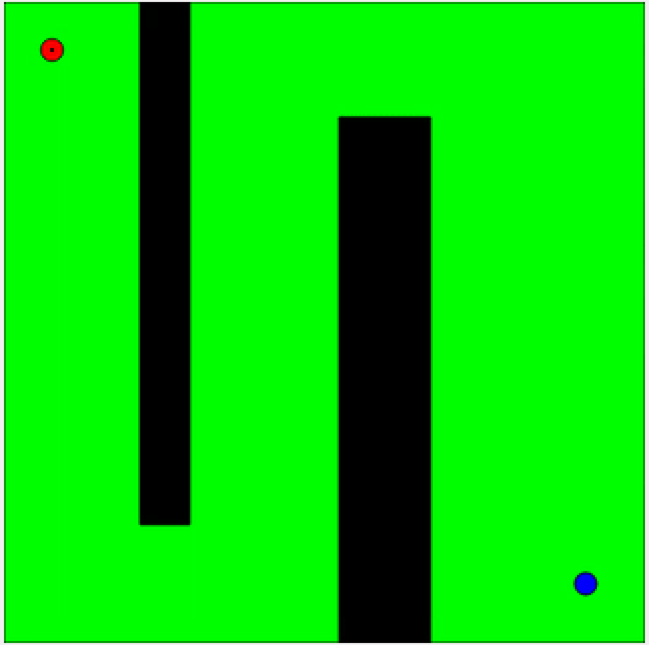
Alternative approach

- Filter potential map points by start area
- Use door-finding algorithm
  - To reposition
  - As localization

Functions – Planning

Rapidly expanding Random Tree

- Randomly generated path
- Avoids known obstacles



Functions – Planning

Improve random path

- Split path
- Avoids known obstacles
- Implement PICO rotation

Functions – Dynamic Obstacle Avoidance

Obstacle avoidance during movement

- Detect points that are too close
- Push PICO away

Functions – Static Obstacle Avoidance

What happens when an obstacle is encountered that can’t be avoided?

- Add unexpected obstacle to map
- Move to a safe spot
- Plan a new route

Improvements

- Recognition of moving objects
  - Avoidance with extreme prejudice
- Optimization of path planning
  - Find shortest path
- Optimization of localization
  - More robust
  - Faster
  - Quick updates
  - Improved preprocessing
    - Corner types

Fin.

Questions?