

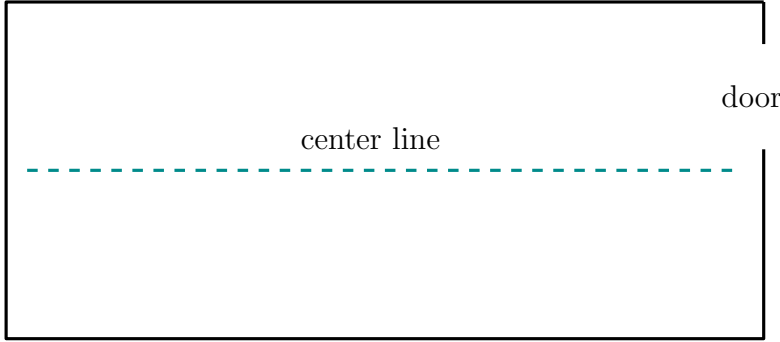
# Pseudo code for mobile robot

Lennart Großkreutz

January 11, 2023

## 1 Definitions

We use the term *center line* to describe the line depicted in the graphic below:



## 2 Algorithm

**Data:** none

**Result:** robot leaves the room through the open door without touching any obstacles

```
while door not yet passed do
    measurements  $\leftarrow$  scan();
    if the two long parallel walls detected in measurements then
        if robot not on the center line yet then
            rotate so that center line is in sight with no obstacles in between;
            move a fixed distance  $d_{searching}$  towards center line;
        else // robot already on center line
            if robot's gaze direction not aligned with center line then
                rotate the shorter way until robot's gaze is aligned with center line;
            if distance from robot to wall  $\leq$  fixed distance  $d_{wall}$  then
                //  $\alpha, \beta$  are constants indicating in which angle range to search the door
                (doorDetected, doorMiddleAngle) = detectDoorInAngleRange( $\alpha, \beta$ );
                if doorDetected then
                    // Hope to pass the door now
                    rotate so that robot is facing in the direction of doorMiddleAngle;
                    move forward a fixed distance  $d_{door}$ ;
                    return;
                else
                    // door wasn't found on this side of the center line, turn
                    rotate so that robot's gaze is aligned with center line in other direction;
            move forward a fixed distance  $d_{centerline}$ 
        else // no clue where robot is in the room
            while nearby obstacle in gaze direction do
                rotate by a random angle in a fixed range  $[\alpha, \beta]$ 
            end
            move forward a fixed distance  $d_{random}$ 
    end
```

**Algorithm 1:** Room escape

### 3 Sub-Routines

In this section, sub-routines are described semantically.

#### 3.1 Sub-Routine *scan()*

The robot gradually does a full rotation, meanwhile creating a list of measurements. It stores the current distance captured by the sensor, then it rotates for a fixed angle  $\phi$  (e.g.  $\phi = 1^\circ$ ) until the robot has completed a full rotation.  $\phi$  should be chosen as divider of  $2\pi = 360^\circ$  so that the robot returns to the original gaze direction after the scan.

#### 3.2 Sub-Routine *detectDoorInAngleRange*( $\alpha, \beta$ )

This sub-routine does a partial scan in the angle range determined by the function arguments  $\alpha$  and  $\beta$ . If a considerable and sudden increase in distance values followed by a considerable and sudden decrease is detected within this partial scan, we interpret this as the door and output the estimated angle corresponding to the vertical door middle as well.

The core idea is that this function is only called when we already have some information about the robot's position, namely that it's on the center line and that its gaze direction is aligned with the center line and it's close to a wall in sight. These information enable us to search for the port in a certain angle range. In a previous version of the algorithm, we searched for a door-like pattern in the distance array of an entire rotation. This led to problems because table legs also produce door-like patterns in the distance array, so the area between two table legs was often misinterpreted as the real door.

### 4 Technical remarks

- The robot does never know certainly whether it passed the door, so the outer loop is rather “while(true)”. The robot is stopped externally as soon as it passed the door. That's also why the *return;* statement doesn't appear in the real code.