Universidad Complutense de Madrid

X Modelling Week 2016

13th-17th June 2016

Real-time navigation of bioinspired cognitive robots in static and dynamic environments

Carlos Lamas Lieb

Luis Lorenzo Alvarez

Blanca Pablos Martín

Instructors: José Antonio Villacorta Atienza & Carlos Calvo Tapia

Contents

Introduction 3

- The problem 3
- Involved processes 3
- Some simplifications 3
- Goals 3

Visual process 4

- Zenithal camera 4
- Frontal camera 5
- Kinect 5

Cognitive process 7

- Rat brain behaviour 7
- Cognitive mapping with no obstacles 8
- Cognitive mapping with obstacles 9
- Approaches applied to locate obstacles 11

Navigation process 12

- Feedback 12
- Navigation simulation 12

Conclusions 14

Introduction

In this report we will show how we can apply neuro-robotic techniques inspired on biology models to robot navigation.

The problem

The best way to fix the scope of our work is imagining how a robot should interact in a risky environment taking its own decisions remotely supervised in some cases, in others running in autonomous mode. Challenges in such situations should be:

- Perceiving the environment and associated risks
- Mapping environment to data that could be handled by a computer
- Navigation taking into account constraints on the movement of the robot

Involved processes

In this report we will give a detailed explanation of the following processes:

- Visual process: problems introduced by different types of cameras
- Cognitive process: map visual perception on a numeric lattice adding making-decision information too
- Navigation process: translate robot decisions into real movements making feasible trajectories

Some simplifications

In our experiment we fix a reduced frame for our tests:

- Static environment
- No on-board camera but a zenithal
- Reduced space of movements (5 x 3 m)
- One obstacle

Goals

I. We will understand and develop a complete model accomplishing requirements for each ones of the previous processes (visual, cognitive, navigation)

II. More relevant objective is achieving robot navigation on the arena surrounding a fixed obstacle (no-collision).

Visual process

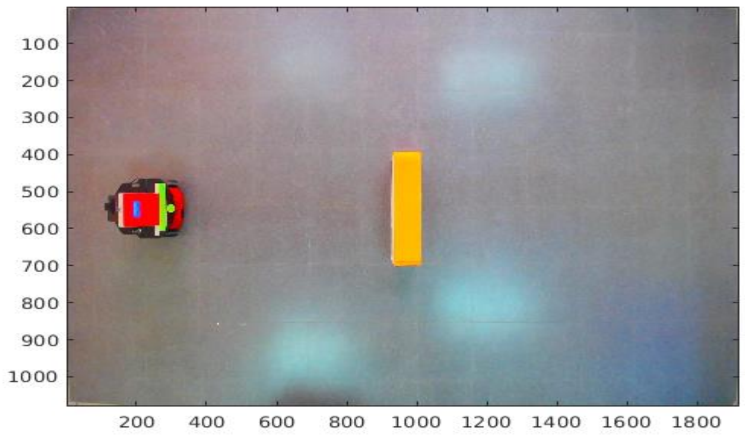
In order to move the robot around the arena, the first thing we need is visual information about the surrounding environment, for which we have considered three options:

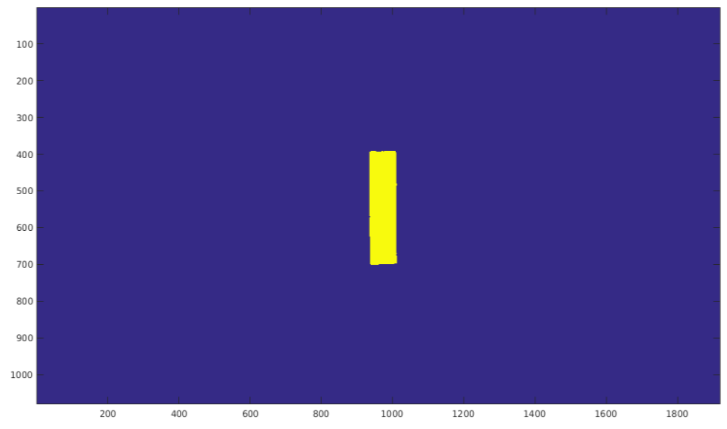
- Zenithal camera fixed on the ceiling
- Frontal camera on the robot
- Kinect (motion sensing device)

Experimentation with the three cameras has been performed in order to compare them and choose the best option for our purpose. We will also study the processing of the images that we will need to provide the robot with the visual information it needs to create a cognitive map

Zenithal camera

Our zenithal camera is fixed on the ceiling, approximately 3 meters above the arena (a zenithal camera should be high enough to avoid the margins of the obstacles appearing in the image). We took a picture of the arena with the robot and an obstacle, and we applied an RGB filter to recognize the position of the box.





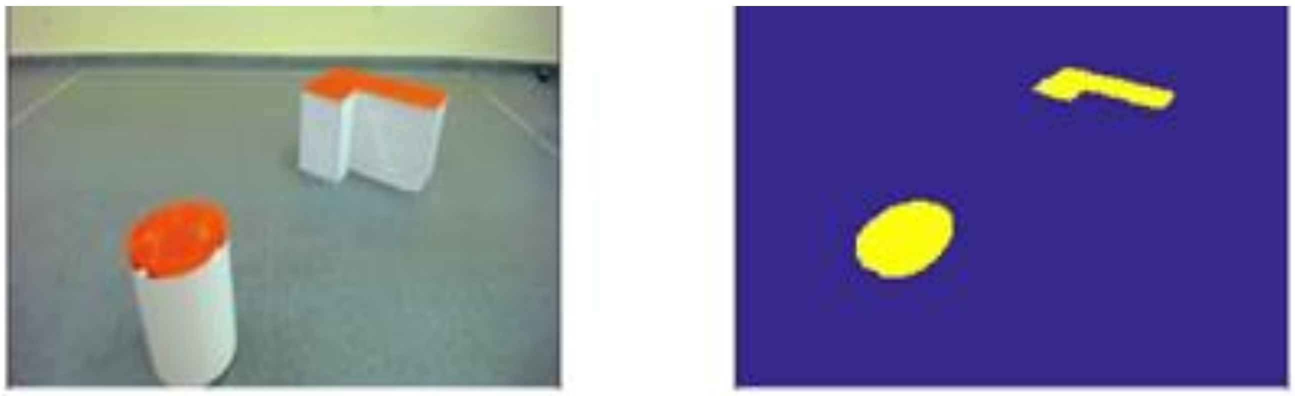
This way, we obtain a binary matrix representing a discretization of the arena: 1s represent the obstacle and 0s represent the empty space. The position of the robot is saved as a point so, in order to take in account his width and avoid it to collide with the box, we virtually extended the obstacles in the matrix with the width of the robot.

The main advantages and disadvantages we observed were:

+ We can get a view of the whole arena
+ We can easily get information about the relationships between the objects
+ The mathematical simplicity given by the planar image, which makes it specially easy to handle.
- It is external to the robot, which makes the robot loose autonomy.

Frontal camera

It consists on a camera placed on top of the robot directed to the arena. It gives a frontal view of the arena which we can filter as we did with the zenithal camera in order to get visual information of the obstacles.

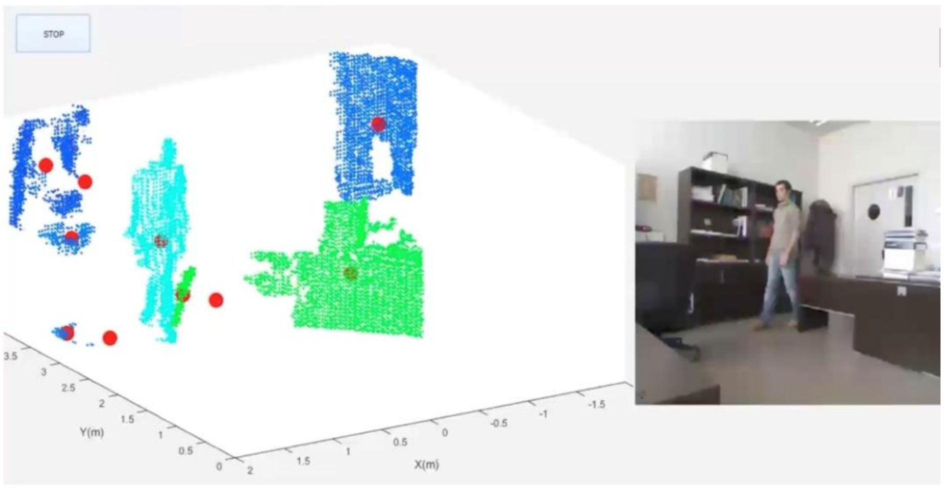


The main advantages and disadvantages we observed were:

+ It is on board, which is a huge advantage on itself due to the autonomy it gives to the robot.
- Monocularity: to be able to estimate the distance between the objects, we would need a point of reference (for example, all the obstacles having the same height)
- We lose information because of perspective: further obstacles will be represented with less pixels.

Kinect

A Kinect camera gives us a 3-dimensional view of the arena without the need of processing the image by using the provided software.



The main advantages and disadvantages we observed were:

+ With a 3-D model, we avoid a lot of the problems we stressed, such as the need of reference points or model simplifications.
- It comes with a hard implementation of the following processes.

Cognitive process

After we have obtained visual information of the arena, we are going to describe the cognitive process, in the sense of how we can tell the robot how to reach a point in the arena. For that, we find our isnpiration in a living being's cognition process. The first thing that we have to understand is the meaning of cognition. If we check the definition in the Oxford Dictionary of English (ODE), cognition is "*the mental action or process of acquiring knowledge and understanding through thought, experience, and the senses*". In our case, we are going to apply this concept to the process of understanding the area where the robot is in order to reach a target.

Rat brain behaviour

For our purpose, we are going to get our inspiration from a rat's brain; to be more precise, we are going to mimic the cognitive process of a rat. To understand how it works, we use as example an experiment in which a rat is located in a cylinder with some electrodes in its head. As we can see in Figure 1 the colours measure the activity on neurons in four different cells, where this process is done.

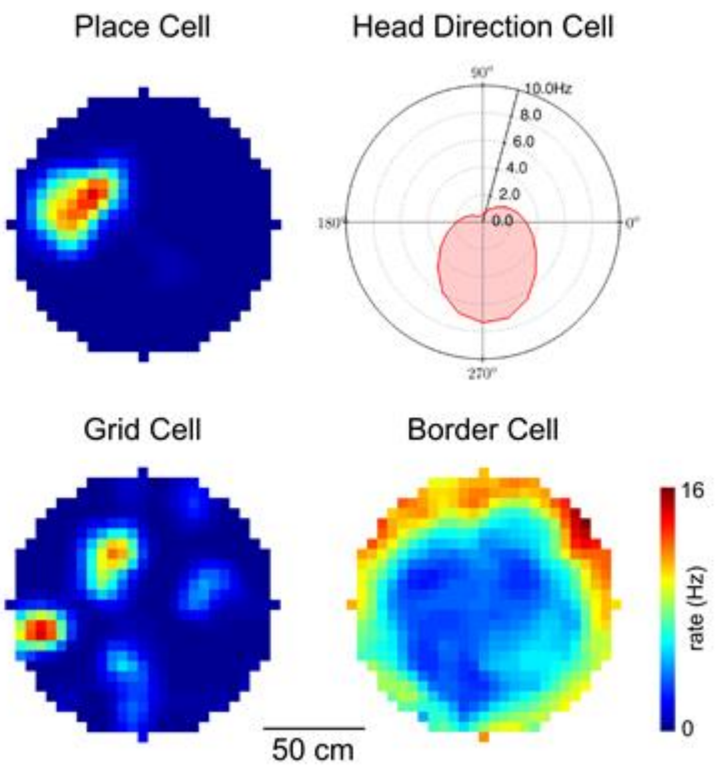


Figure 1: Estimulation of the different neurons of a rat during the cognitive process.

There are four types of cells that help the rat understand its surroundings:

- **Place cell**: it's responsible for giving information about where the rat is located in the arena, that is, the neurons activated, coloured in red, give the location of the rat inside the cylinder. If the rat of the experiment moves to another place, the part that is red at this moment will be blue and the new position of the rat in the cylinder is going to be red.

- **Head Direction Cell**: it gives the rat the orientation of where his head is facing, i.e. which direction is looking the rat in relation with the arena.
- **Grid cell**: it builds a grid of the area where the rat is and it gives also the distance between the points of the grid. In that way if the rat wants to focus a specific area in the arena, it can make a finer grid of this place and take more information.
- **Border cell:** provides the limits of the arena in which the rat is moving.

The question now is how we can reach any point of the arena using this information provided by the brain. We tried to apply these concepts to a really simple case, an empty and rectangular room.

Cognitive mapping with no obstacles

One idea is to follow a random-walk, which is used by the rat to explore the room, but there are two big disadvantages:

1. It's not efficient to make the trajectory between the point where the robot is and the target.
2. There is a low probability to reach all the points of the arena.

Because of this, we tried to find another way of exploration. The idea we followed is that the robot can reach all points of the stage in a straight line, like in Figure 2.

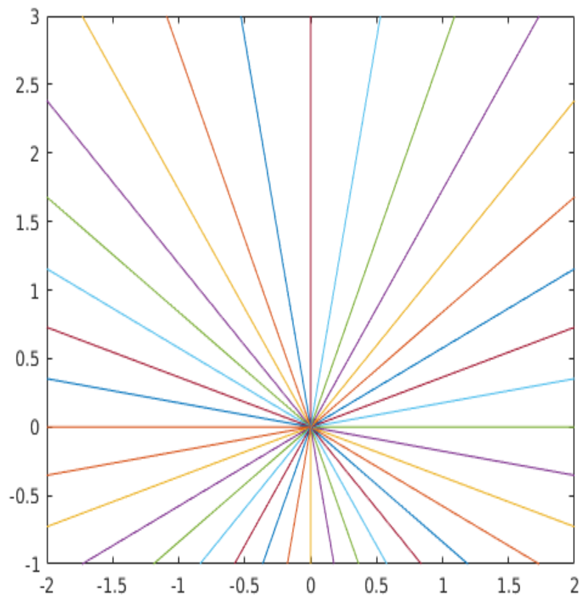


Figure 2: Trajectories to any point of the arena.

Hence, following this trajectories, we can create a map of the arena. Another idea is that any given point of the arena is reached by a circle with centre the original robot position, as we can see in Figure 3, so we can reach any point with propagating circles from the robot position as well, which gives us another cognitive representation of the space to explore. This is the key of virtual exploration: in order to understand and move around the arena using our cognitive map, we just have to follow the radial direction of the circles to reach any point of the arena.

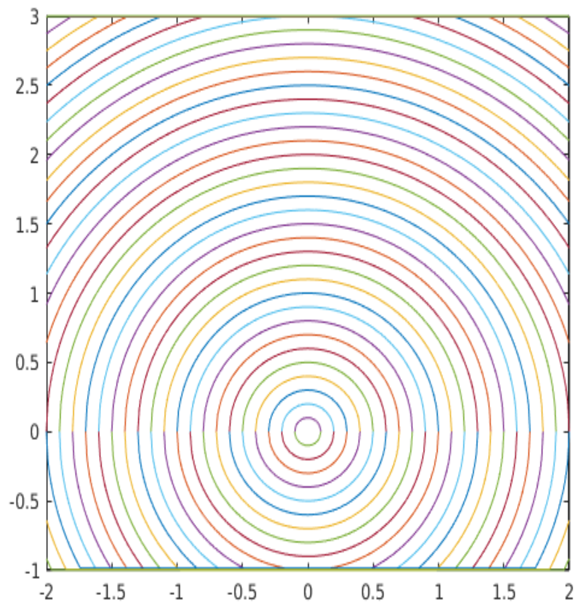


Figure 3: Virtual exploration of the arena using propagating circles.

To increase the difficulty of our problem, we are going to add an obstacle. We already have the visual information of the obstacle given in the previous section.

Cognitive mapping with obstacles

To extend the previous idea, we are going to introduce an example of how cognitive process works in this case. For that, imagine that we are in an empty room with a closet in the middle like in Figure 4, where the orange section is the closet, the black point is our position and the empty room is the rest.

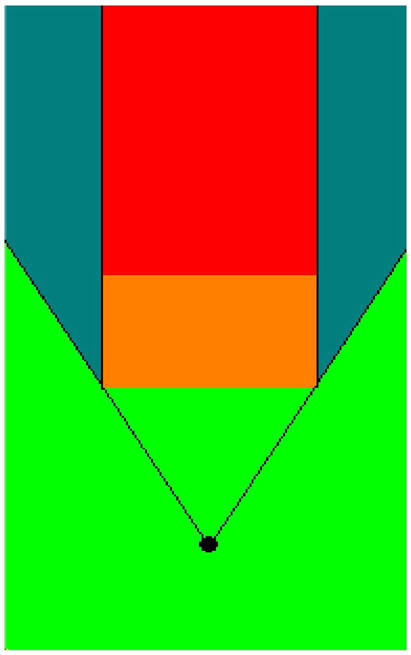


Figure 4: Empty room with an obstacle (orange region) and robot's position (black dot).

The first thing we should notice is that if we were located where the robot is, we wouldn't be able to see behind the closet, to be more specific we can't see what happens behind the visual lines that are naturally created from our position (black lines connecting the black dot and the corners of the obstacle), but in front of them and in front of the obstacle (the green region), we can follow the previous idea of straight trajectories. However, we need a step more to extend this idea iteratively to reach a point behind the visual lines in order to cover the entire arena.

As we would act in real life, we are going to move up to one corner of the obstacle to see what happens behind the visual lines. To avoid hitting the obstacle, we increase its size, as we explained in the previous section. In this new position we are going to encounter the same problem again, since there are still two new visual lines, but the idea will be the same. If we want to reach a point inside our visual perception (blue region), we can reach it with a straight trajectory; and if we want to go to the non-visible region (red part), we need to move to the corresponding corner and follow another straight line that is formed from the new position to the point we want to reach.

So with this point of view we can determine three regions that can be reached in one, two or three straight lines. Following the idea of propagation waves as before, we can compare the virtual exploration with the auto-waves that are propagated in the whole space and then disappear when collide, like the fire waves, as we can see in Figure 5.

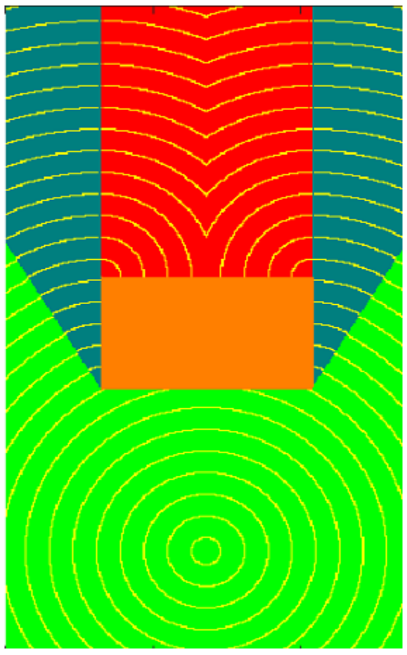


Figure 5: Virtual exploration with propagation waves.

In conclusion, to reach a point of the different regions, we just have to follow the radial lines to the propagating circles we have defined for the whole arena, as we can see in Figure 6 for the example with no obstacles. The map gives us all the information we need to reach any given point with no collision. Now, the question is how our robot is going to know where the obstacle is located from the image we obtained previously in order to create this cognitive map.

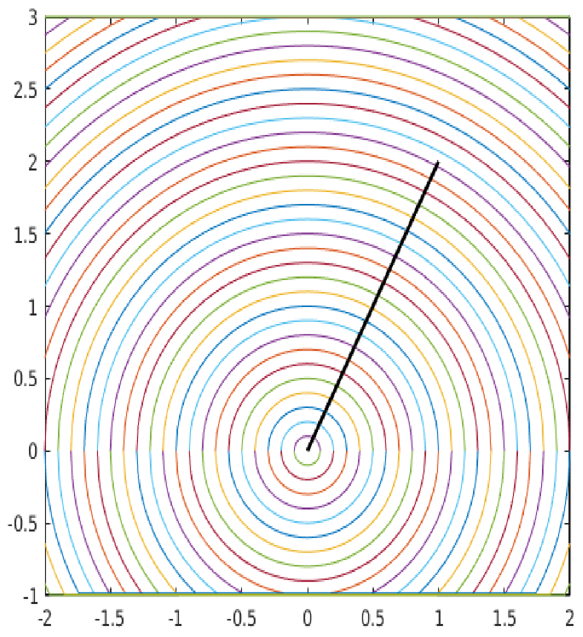


Figure 6: Reaching a point following radial lines.

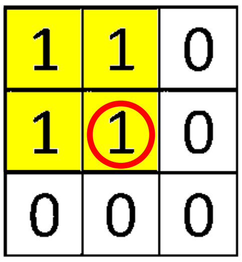
Approaches applied to locate obstacles

In order to implement the visual information we have of the obstacle into our mathematical model, we have considered two different approaches:

i. Scanning by straight visual lines until they collide with the obstacles.
ii. Detection with logical numeric matrix defined as:

$$z_{i,j} = \begin{cases} 1 \ \text{ if } \ (i,j) \in \Omega, \text{where } \Omega \text{ define the obstacle domain} \\ 0 \ \text{ otherwise} \end{cases}$$

taking into account that the sum of the numbers of the matrix corresponding to the neighbourhood of each corner is four:



1	1	0
1	1	0
0	0	0

Therefore, at this moment we are able to create our cognitive map using the iterative propagating circles approach that we previously explained.

Navigation process

Now that we have a cognitive map which tells us how to reach any given point of the arena, we have to actually take the robot there. In order to do this and to understand how we can tell the robot how to move, we have to understand how we can handle the robot directly from Matlab in real time. We do so via a set of simple commands that we can use to configure any given trajectory:

- arrobot_setdeltaheading(deg): it makes the robot turn on itself at initial position by giving it an initial angle.
- arrobot_setvel (mm/s): in order to make the robot move in a straight line, we can set a linear velocity with this command. To make it travel a certain distance, we can set the velocity and apply it during the corresponding time with Matlab command pause().
- arrobot_setrotvel (deg/s): in order to make the robot move in a curved line, we can set an angular velocity with this command in an analogous way to the straight movement we have described.
- arrobot_stop: it makes the robot stop its movement.

Feedback

A very important concept when moving the robot is feedback. When a living being is moving, it uses the visual information it gets in real time to correct its trajectory with the given feedback: otherwise, it would probably deviate from the planned trajectory and make mistakes. We can apply this concept to robot navigation without using any additional visual information: just by getting a reference of its location with the following commands:

- *arrobot_getx* and *arrobot_gety*: gives us the (x,y) coordinates of the robot in the zenithal image.
- *arrobot_getth*: gives us the orientation angle θ from initial angle.

Navigation simulation

Using all the previous results, we made a simulation of robot movement using the processed image from the zenithal camera, the cognitive map to determine our trajectory to reach a selected point and the navigation commands to actually take the robot there.

We chose a point behind the obstacle, we determine the trajectories given by the propagating circles as Figure 7 depicts, and we calculate the angle and distance of each part of the trajectory (taking into account the scale factor between image pixels and actual meters) in order to give the robot the navigation commands it needs to move along the trajectory.

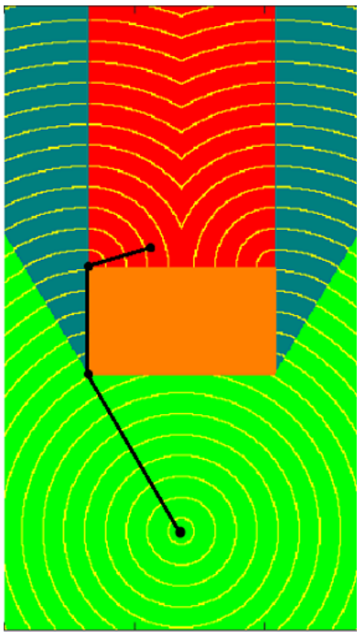
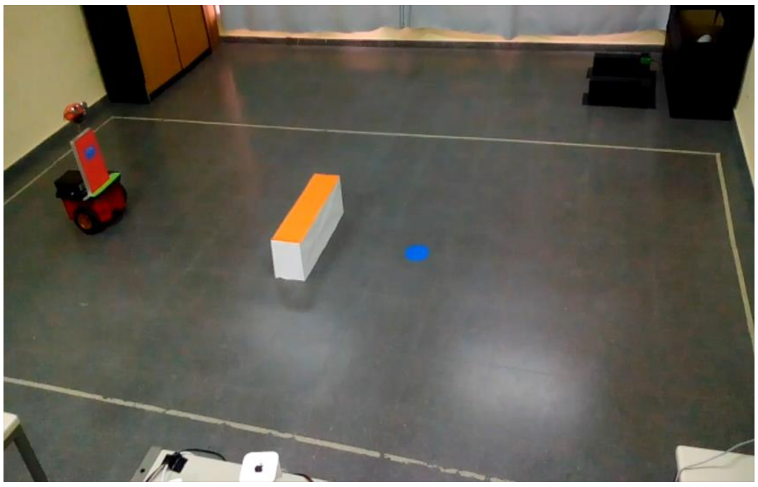


Figure 7: Calculated trajectory to reach a point behind the obstacle.

We simulated this trajectory successfully with the robot: no-collision was achieved due to the extended margins we considered in the visual processing step.



Conclusions

- We succeeded to translate a simple human cognitive behaviour to a robot where cognitive map change very much from the simplest scenario bases on a clean and empty arena to when we introduce a simple element which is a static square obstacle (i.e. Fig.3 vs Fig. 5). We show the way how a real environment can be mapped into helpful information on a lattice, given that only numeric data that can be handled by automata.
- Information processing even for the simplest scenario is quite complex: it requires hard implementation to define all possible trajectories to cover all the arena.
- Our model is a very first step previous to the much more complex dynamic scenario and even in our case we tackle with a lot of difficulties to map pictures taken by one camera to get useful and accurate information for our automata.