Cooperative Robotics

Final Assignment
Aziza Zhanabatyrova

Video demonstration available at:

https://www.youtube.com/watch?v=iEytQ7Vdhvg
https://www.youtube.com/watch?v=_yx_y9MB1sI

CODE EXECUTION

The code can be executed by running

python real_multi_player.py robot1 (for white computer robot)
python real_multi_player.py robot2 (for red computer robot)

Previously, for the first assignment with the physical robots, the code was being executed from my personal computer over a wireless connection. While the wireless connection provided the convenience of dealing with the code in my own computer, it affected the whole operation in different ways:

- The latency in communication associated with the WiFi connection influenced negatively the performance of the controllers present in the code.

- Since the detection of the ball depends on the image that is retrieved from the camera, the lower speed of communication of the WiFi connection impacted the performance of this feature.

- Packet loss can be significantly high over a wireless connection, especially if the network is suffering from congestion, and influence the whole program in an undesirable way.

The aforementioned issues were noted to have higher negative influence in the assignment with two physical robots. For this reason, it was opted to upload the program on board of the robots.

HOW THE PROGRAM WORKS

The program is composed of two parts:

1. Recognition of the environment

In this part, each robot is required to obtain their orientation and distance relative to the ball, to the goal and to the partner robot.

First, each robot starts by rotating anticlockwise to find the ball and the goal. Once an object (either ball or marker) has been found, the robot should rotate until the center of the object (ball and marker) is located directly in front of the it. This process of rotating to align with the center of the object is denominated as alignment.

1.1. Alignment with the ball

Once the ball has been seen, the robot calculates the number of pixels from the center of the ball to the center of the image. Let's call this number *misalignment of pixels*. With this value, the robot then estimates the angle by which it should rotate in order to see the ball exactly in front of it. Once the robot has rotated by the angle estimated, it calculates again the misalignment of pixels. Again,

with such a value, the robot estimates the angle by which it should rotate to see the ball in exactly front of it. This process finishes once the misalignment of pixels is less than a tolerance (set as 12).

To rotate by a certain angle, the robot uses the information from the odometry (angle *theta*) and a proportional controller. The gain for the controller was defined experimentally by simple trial and error.

To estimate the angle by which the robot has to rotate, a linear function is used:

$$\zeta = 0.00001917 * e$$

where ζ is the angle in radians by which it has to rotate and e is the misalignment of pixels.

Such function was obtained from a linear regression done with several pairs of points (angle and misalignment of pixels) collected experimentally.

Notice that, according to the function, the angle by which the robot has to rotate only depends on the misalignment of pixels. Indeed, it also depends on the distance to the ball, but such dependency has been noticed to be very small and thus negligible.

Also notice that with this method of alignment with the ball, the image processing is done much less frequently. Previously, in the last assignment, the image was being processed at every instant.

It has been noted that the ball, no matter at which distance it will be from the robot, is always on the half lower part of the image. Therefore, the image obtained from the camera is always cropped before being processed. This reduces significantly the time required to process the image.

Another improvement is related to the sampling period of the camera. To avoid that the same image is processed more than once, I have developed a simple solution. Once a new image arrives, an integer variable is set to one and it means that the image is allowed to be processed, once it has been processed, such variable is set to zero. The image can only be processed when such variable is set to one.

1.2 Alignment with the marker

Once the marker has been found, the robot rotates until its center is directly in front of the robot. For this, a proportional controller has been used and its gain was defined by trial and error.

1.3 Obtaining orientations and distances

Once the robot has aligned with the object (either the ball or the marker) it saves the relative distance to the object by using the information of the depth camera and its orientation using the data provided by the odometry.

When both robots have obtained the relative distance and orientation to the ball and marker, they exchange this information with each other to calculate the distance between themselves (H in Figure 1) and to calculate the angle (φ) that it needs to rotate, given that it is looking to the ball, in

order to see the partner robot. Also, they compare their distances to the ball. The robot that is closer to the ball will be the one to pass the ball to the partner robot.

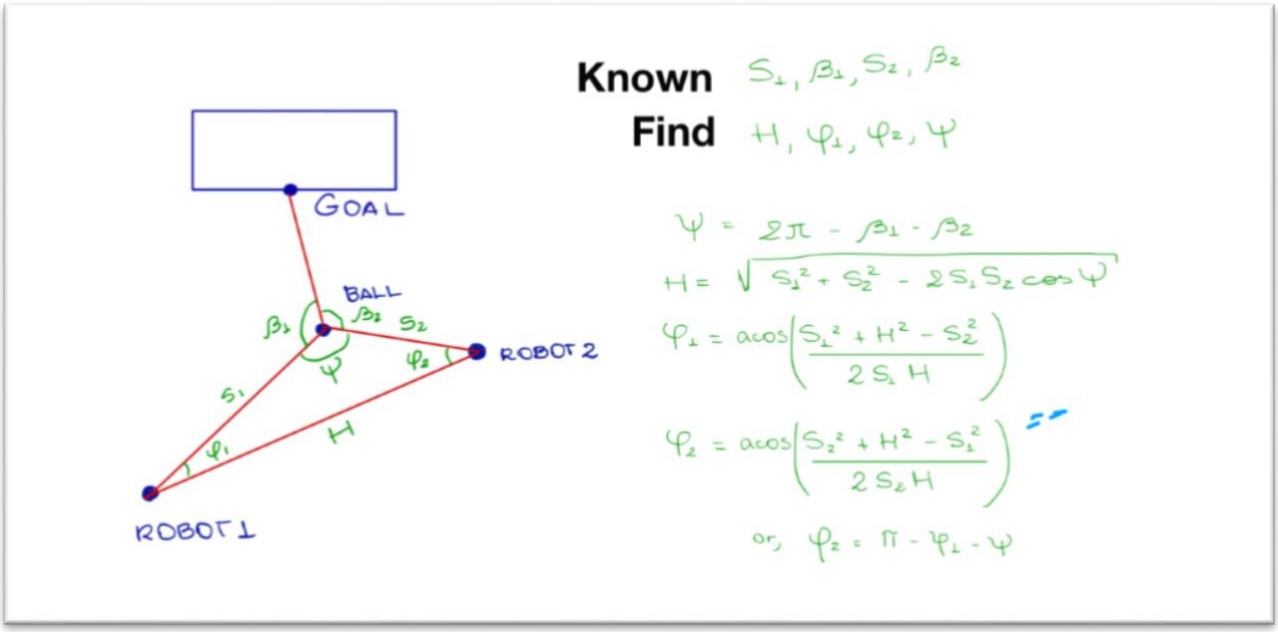

Figure 1. Finding the partner robot

2. Action on the environment

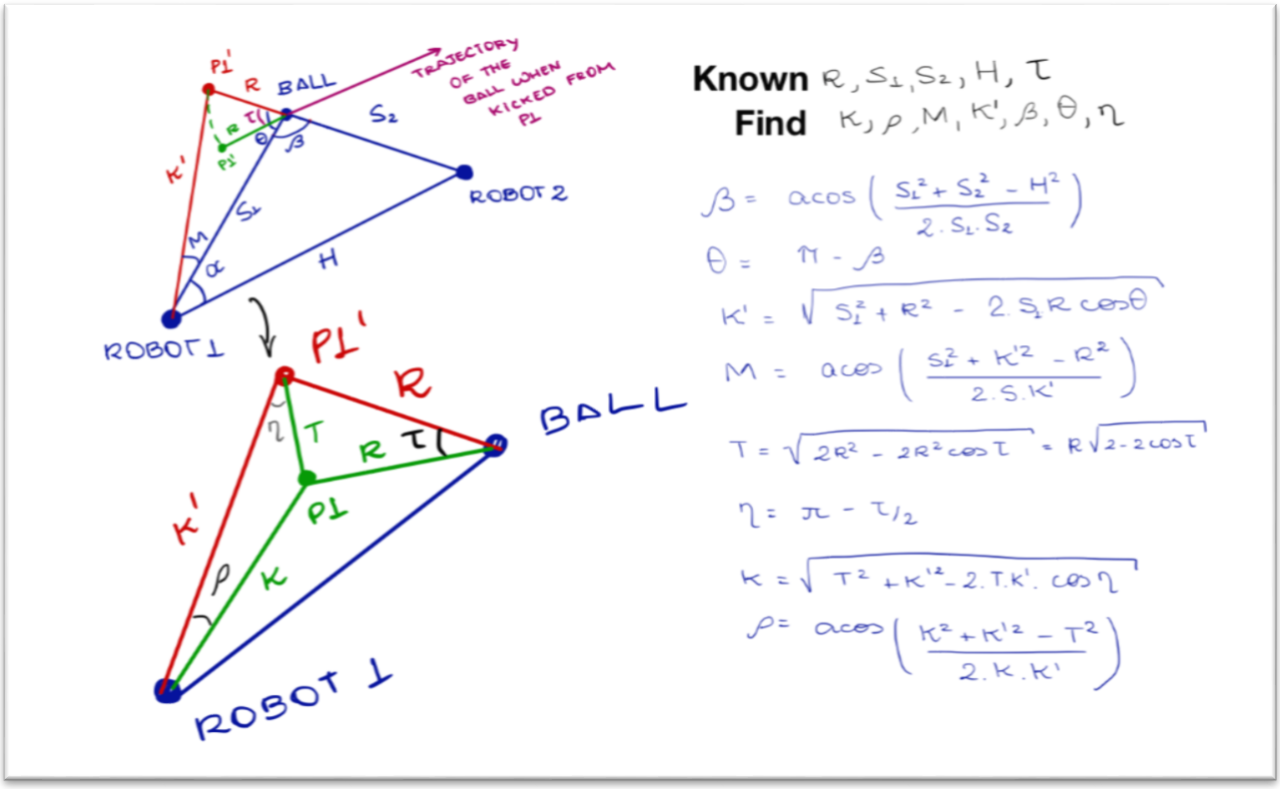

Figure 2. Finding point P1. Point P1 is the location such that if the ball is kicked from there, it should reach the front of the partner robot 2.

Once the environment has been recognized, the robot that is closer to the ball takes the action of passing the ball its partner robot. For this, firstly the robot finds the distance and orientation to a point P1 (Figure 2). Then it rotates until it is looking to the point P1 and finally 'walks' to point P1.

When the robot arrives at point P1, it rotates until it sees the center of the ball directly in front of it. The robot finally kicks the ball and waits three seconds until it communicates with the partner robot to give the permission to continue.

The partner robot starts to rotate anticlockwise until it sees the ball. Once the ball has been detected, the robot aligns with it and calculates again the distance to the ball and the orientation. With the information of the position of the ball and the marker with respect to itself, the robot finds a desired position P2 such that P2, the center of the ball and the center of the goal are collinear, as illustrated in Figure 3.

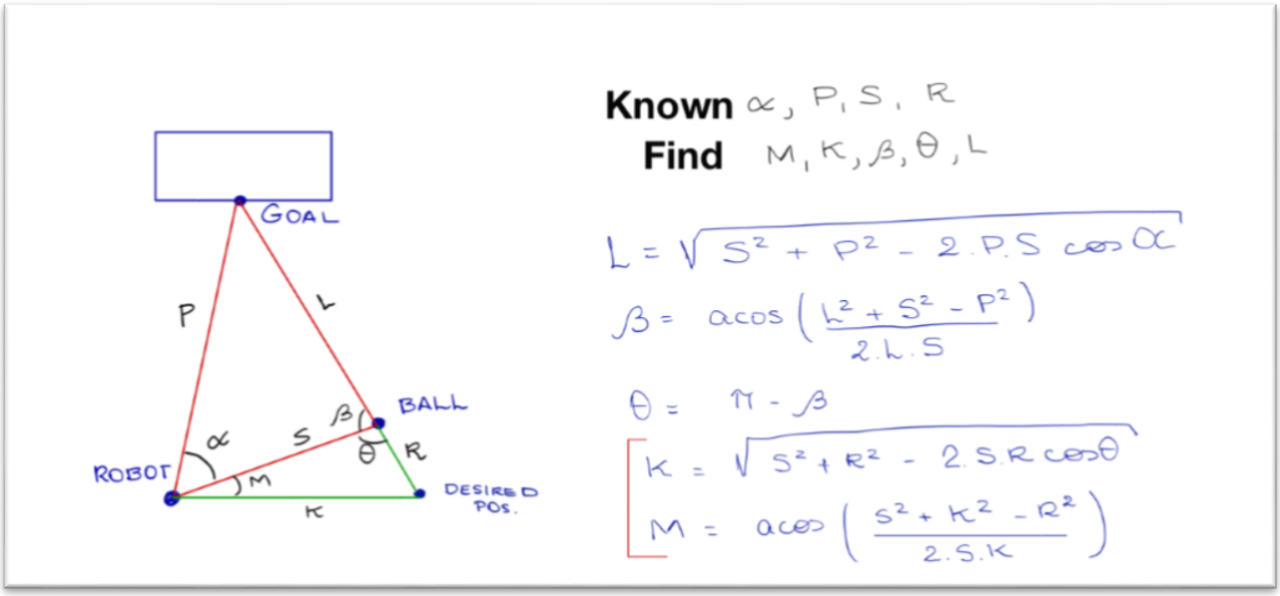

Figure 3. Finding the desired position to score the goal

The robot rotates until it sees P2 directly in front of it and walks to P2. It then finds the ball again, aligns with it and kicks the ball to score the goal.