# EE4306 - Distributed Autonomous Robotic Systems

## FCDurian

*Broekhuizen Casper (A0149200U)*
*Niemelä Petri (A0149463X)*
*Shridhar Mohit (A0105912N)*
*Tran Khanh Huy Paolo (A0149449N)*

**Singapore, 16th April 2016**

## INTRODUCTION

The objective of this project was to develop a rudimentary robot-soccer control and planning system. Over the course of the semester, we built a primitive AI capable of engaging in a 3-on-3 soccer match with 2-wheeled robots.

Implementation Stage:

1. Penalty Kick
2. Ball Dribbling and Passing
3. Team Competition

Software and Tools:

1. V-REP PRO EDU v.3.3.0
2. Python

## STAGE 1

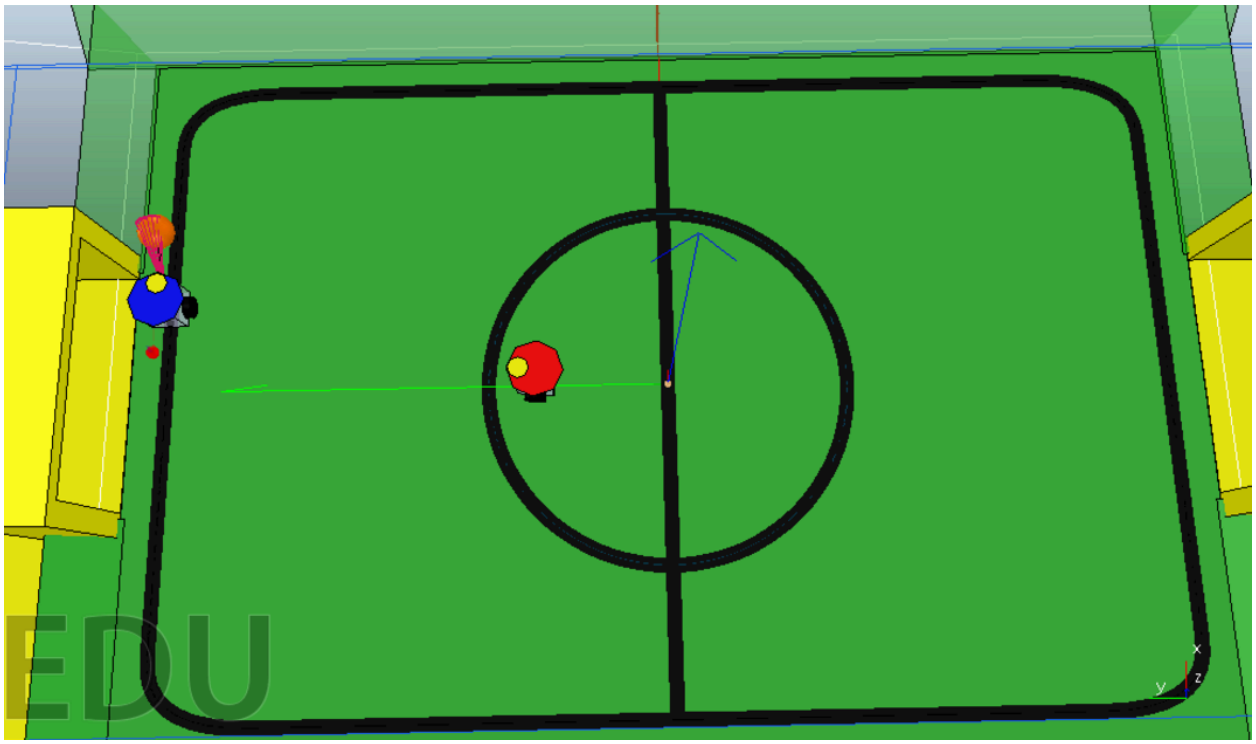
### Objective

Design a control system for a goalkeeper, capable of blocking three penalty shots, shot in an arbitrary at goal

### Strategy

The goalkeeper executes the following actions to save the ball:

I. Extrapolate the ball's trajectory towards the goal-post based on its velocity
II. Compute the intersection point between the estimated trajectory and its area of action
III. Move towards the intersection point and kick the ball away



The ball's estimated pose (red-dot above) on the goalpost line is computed by:

$$\Delta t = \frac{|b_y - g_y|}{B_y}$$

$$T_x = B_x \Delta t + b_x$$

Where $(b_x, b_y)$ are the ball positions, $(B_x, B_y)$ are the ball velocities, $(g_x, g_y)$ is the goal center position, and $(T_x, T_y)$ is the estimated ball pose on the goal line. After computing the intersection point, the goalkeeper will orient itself and drive towards the location (p-controller) to kick the ball away from the goal post. Then quickly return to the middle of the goal area.

This simple 'intercept & kick' system was sufficient for the penalty shoot-out stage. However, with the introduction of multiple strikers in Stage 3, this defense strategy can be easily broken with a naive pass-and-shoot technique. The wheel velocities were also carefully tweaked to prevent the robot from tipping over during rapid accelerations caused by sudden direction changes.

## Structure of Algorithm

The ball and player are saved in their own class. Where there position, velocity, etc. are updated every cycle, movements are added to a queue and executed every update of the player object. The required movements are calculated and put in the queue.

This structure has been modified as seen later in the report (Stage 3 - Structure of Algorithm)

## STAGE 2

### Objectives

I. Locate the ball
II. Navigate towards the ball
III. Dribble around the static obstacles
IV. Pass the ball between the specified zones
V. Score a goal

### Strategy

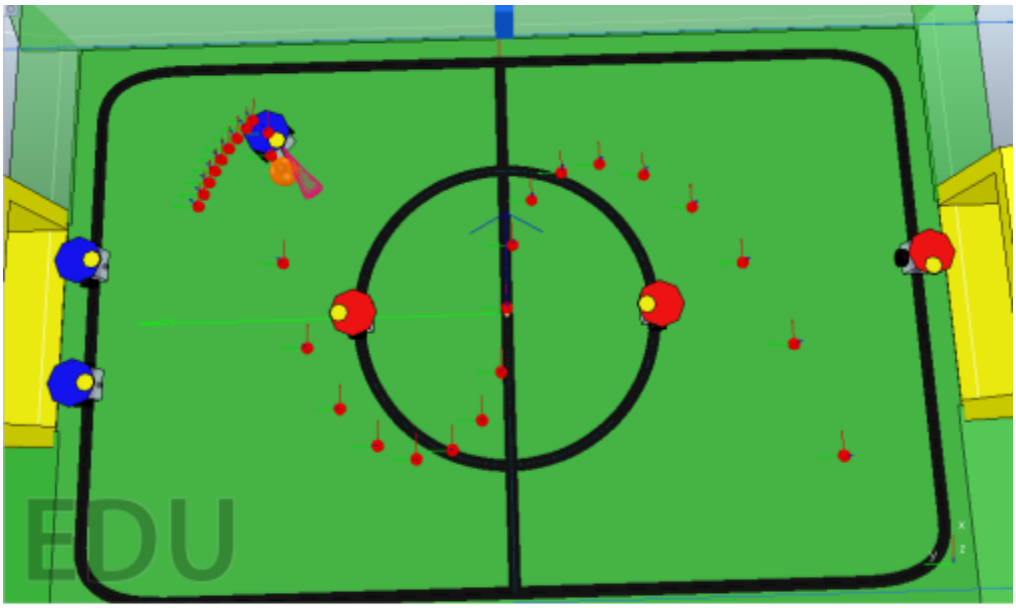



**Figure 1**

#### Planning

Initially, the robot positions itself behind the ball at an approach angle corresponding to the dribble direction. This positioning is based on a heuristic: if the ball is on the upper half, navigate to the mid-point between the top wall and the ball, else if the ball is on the bottom half, navigate to the mid-point between the bottom wall and the ball. This effectively ensures that the bot doesn't accidently knock the ball over before executing the dribble.

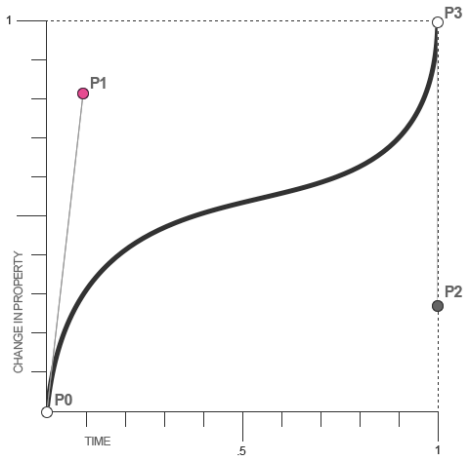
The path planning is executed in two stages: global and local. We use bezier curves to generate a smooth global trajectory around the static obstacles. This trajectory provides a set of waypoints for global navigation (sparse red-dots above). Locally, we generate smaller bezier trajectories to kick the ball towards the next waypoint.

A bezier curve is an *bernstein* polynomial defined by two tangents (current heading and desired heading). Using this type of path planning the system is unlikely to plan over the ball's physical position and ensures the ball is approached from a specific desired approach.

*Bezier curve formula*

$$r = 1 - t$$

$$pos = r^3 p_0 + 3r^2 t p_1 + 3r t^2 p_2 + t^3 p_3$$




$$tan = 3r^2(p_1 - p_0) + 6rt(p_2 - p_1) + 3t^3(p_3 - p_2)$$

## Dribbling phase

Dribbling is executed by pushing the ball over the approximated path. This is done by calculating a new movement for the player to approach the ball at a specific angle and giving it a small "nudge" this movement is also planned as a bezier curves, whereas waypoints for the robot's movement are placed in a queue and executed by a p-controller every update cycle.

In order to make the dribble methodology more robust, the robot backs off every nudge and waits for the ball to slow done if the ball-velocity is too high. The backoff technique ensures that the ball isn't accidently hit when performing rotational adjustments.

This dribble methodology is pretty robust, but slow in terms of execution. The 'wait & nudge' method was sufficient for Stage 2, but in an actual soccer match it would be too slow to be of any practical use. So for Stage 3, we implemented a different dribbling mechanism, which will discussed in a later section.

## Passing phase

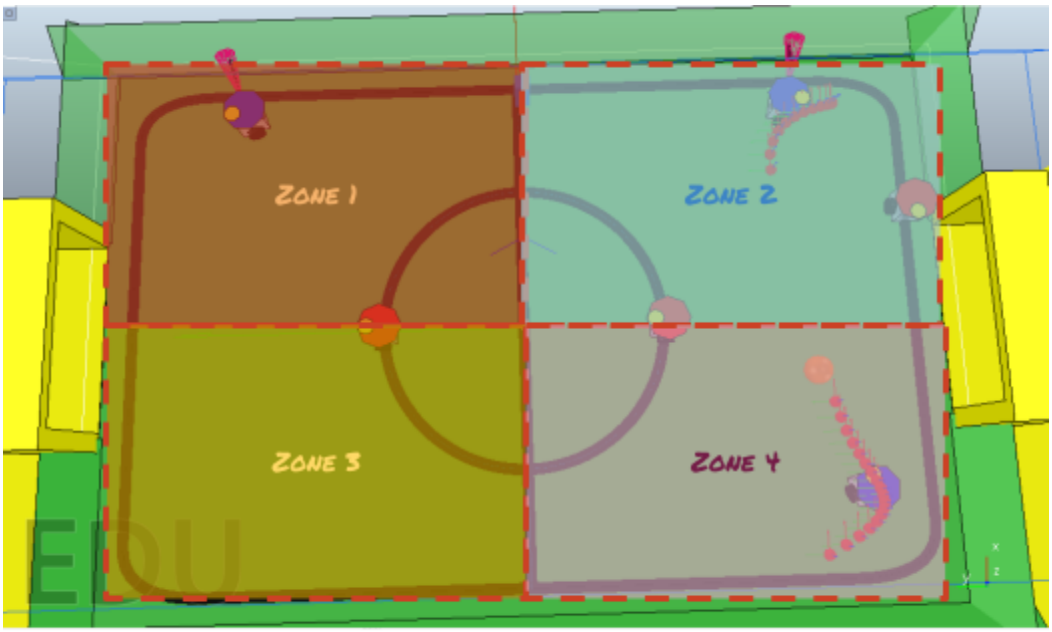



**Figure 2**

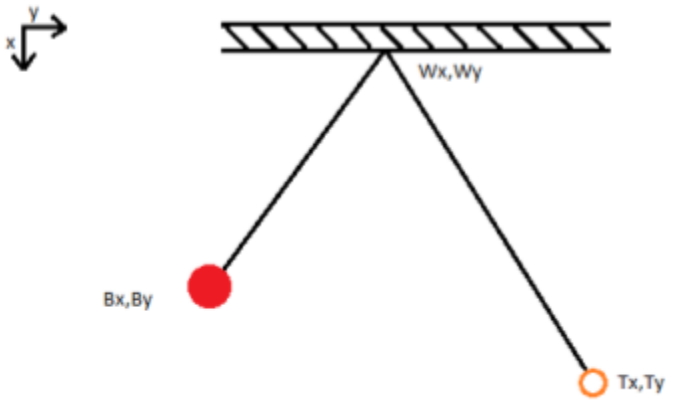
The passing phase is very similar to the dribbling strategy. The passer will generate a smooth bezier approach towards the ball before kicking it. The kick angle is aligned towards the receiver, and the kick power is computed based on the euclidean distance between the ball and receiver's position (linear relationship $power = dist * m + c$, where $m$ and $c$ are constants). Instead of specifying a receiver, the passer can also dictate an 'anticipate position', and shoot the ball towards it, while the receiver slowly drives up to the spot. Currently, the system waits for the ball to stop rolling before executing a pass for better control.

After each kick, the control system checks if the ball landed in the desired zone (see Figure 2). If not, the passer recomputes the bezier plan to kick the ball towards the right zone. Given the uncertainty added by the p-controller and the physics engine, there is a distinct possibility that the ball might get accidentally knocked into the goal during a pass.

It's also quite probable that the ball might get stuck up against the wall. In this case, the bezier planning would generate trajectory points beyond the bounding walls, which would be physically impossible to execute. So we implemented a 'wallshot' technique to bounce the ball off the wall if the last five trajectory points were beyond the field boundaries. We essentially compute an

approach angle to the kick that would cause the ball to bounce back to the desired target position.

$$w_y = \frac{w_x - b_x}{w_x - t_x + 1}(t_y - b_y) + b_y$$




$w$ : wall

$b$ : ball

$t$ : target

We add fixed constant to the kick velocity for wallshots, which reduced the probability of the ball landing exactly at the target position. But the method is quite effective in moving the ball away from the wall.

## Structure of Algorithm

The ball and players are saved in their own class. Where there position, velocity, etc. are updated every cycle, movements are added to a queue and executed every update of the player object. The required movements are calculated and put in the queue.

All calculations, approximations and planning of movements are done in the main program where the trajectories and movements are put in the player object's action queue which is executed and updated every cycle.

## STAGE 3

### Objectives

I. Control a team of three robots to score and defend goals
II. Keep at least one robot on the opponent side of the field

### Strategy

The three robots have fixed roles:

1. Goalkeeper
2. Defender
3. Striker

Contrary to the previous stage, the players in the stage use a combination of bezier and linear planning.

Every role has a common behaviour that is a frustration function. When they are stuck, they will try to move backward or rotate first and then try to perform their original task again.

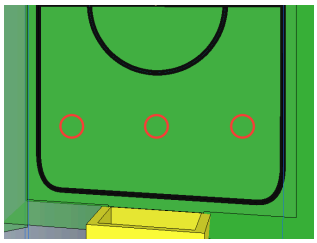
#### Goalkeeper

For the goalkeeper the same goalkeeper was implemented as discussed in stage 1.

#### Defender

The defender reacts when the ball is in its own half of the field. Its role is to push the ball to the other half.

The defender occupies three positions on its own field, based on the ball location respectively on the other field. Left fullback, right fullback and centre back



When the ball enters the defenders half, it will plan a path to get behind the ball in order to kick it back. The path to get behind it is approximated using bezier curves and the kick is a linear

movement towards the ball. Since the path is recalculated every few hundred milliseconds, this also has the effect of intercepting a ball.

### Striker

The striker is always placed on the offense side. When the ball is beyond the mid-field line, the striker will try to get behind the ball as quickly as possible. And then it will push the ball towards the goal post while correcting its heading carefully to stay in control of the ball. If the striker loses control of the ball, it will try to get behind the ball and repeat the process.

## Structure of Algorithm

For stage 3 a completely new approach of structure has been made. Since a lot of robots use the same functionality (kick, dribble, pass, etc.) one action class was made holding all functions to make these movements happen. Than for every role (goalkeeper, defender, striker) a new class was made which makes decision what actions to pursue based on the current situation. This new approach made the program more flexible, cleaner, more robust and better at multiple robots working parallel. This new approach was implemented in Stage 1.

Stage 2 still uses the ‘old’ approach due to time constraints.

# CONCLUSIONS

During the span of the project, problems or hurdles one may encounter at having a distributed autonomous system over multiple robots have been made more clear, some of these problems include motion planning, cooperation of robots and *‘selfishness’* of specific agents. The project also helped to understand the improvements one can make using fuzzy logic or other approaches such as genetic algorithms and artificial neural networks. If the project would have run for a more extended period of time, other improvements could have been made using such approaches. Also the code could be made more structured and a more advanced path planning could be implemented but overall the team is satisfied with the results.