RoboCup Simulation in MATLAB

Robotics Team Design Project - Team 1

Group organisation

Chose a team captain

Research analysis

- Kinematics
- Behavioural

Implementation

- Focused on algorithmic approach
- Can be generalized to behaviour

Name	Responsibilities
Oscar	Path Planning, Visualisation, Github
Alvaro	Ball Dynamics
Shiyi	Gamestate Mechanics, Dribble
Abubakar	Kinematics, Behavioural Logic
Jiwei	Metrics and Validation
San	Player Class Mechanics, Sensor Modeling

NAO - Specification

Data

- Physical Dimension
  - Mass, width, height
- Sensors
  - Camera, Gyroscope
- Actuators
  - 25 movable joint

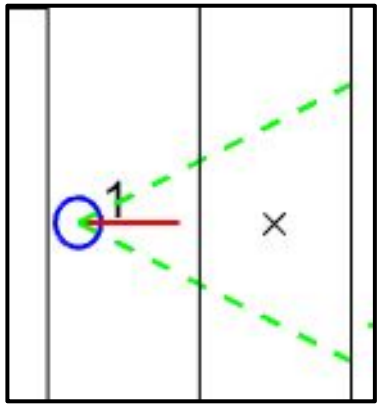
Methods

- Softbank NAO6 website
- NAO6 Datasheet
- RoboCup Competition video on YouTube

NAO - Modelling

Sensors

- Robot and ball searching algorithm



Kinematics

- Controller - Differential drive
- Implement max velocity and angular velocity

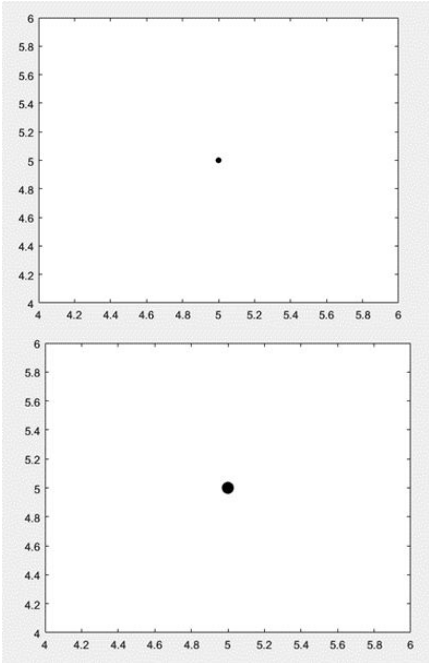
Collision and Fall condition

- Simple collision check
- Average time to get up

Ball

Ball dynamics

- Passing
- Shooting
- dribbling



$$Position = Position + (Velocity * dt)\ [Equation\ 1]$$

$$Velocity = Velocity - (c * Velocity)\ [Equation\ 2]$$

Requirements

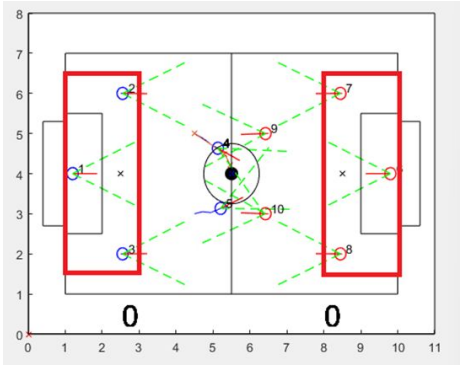
- Plotting function (45 cm)
- Mathematical model

Ball

Shooting

Requirements

- Conditions being met
- Shooting function (work-kinetic theorem)



$$W = F * d = \Delta K; \qquad K = \frac{1}{2} * m * V^2$$

$$W = K_{final} - K_{initial}; \qquad K_{initial} = 0 \; as \; V_{initial} = 0$$

$$W = K_{final} = F * d; \qquad \frac{1}{2} * m * V^2 = F * d$$

$$V = \sqrt{\frac{2 * (F * d)}{m}}; \qquad [Equation\ 3]$$

Passing

Requirements

- Conditions being met
- Check for player to pass
- Passing function (work-kinetic theorem)

$$F = a * d \;\; [Equation\ 4]$$

Force Multiplier	Damping coefficient
0.0565	0.5
0.0365	0.4
0.012	0.2
0.006	0.1

Ball

Dribbling

RobotDribble Function

- Calculate the robot speed vector
- Set the speed and position of the ball
- Update dribbling robot ID and ball direction

Dribbling Conditions

- The robot near the ball
- No other robot dribble

Gamestate

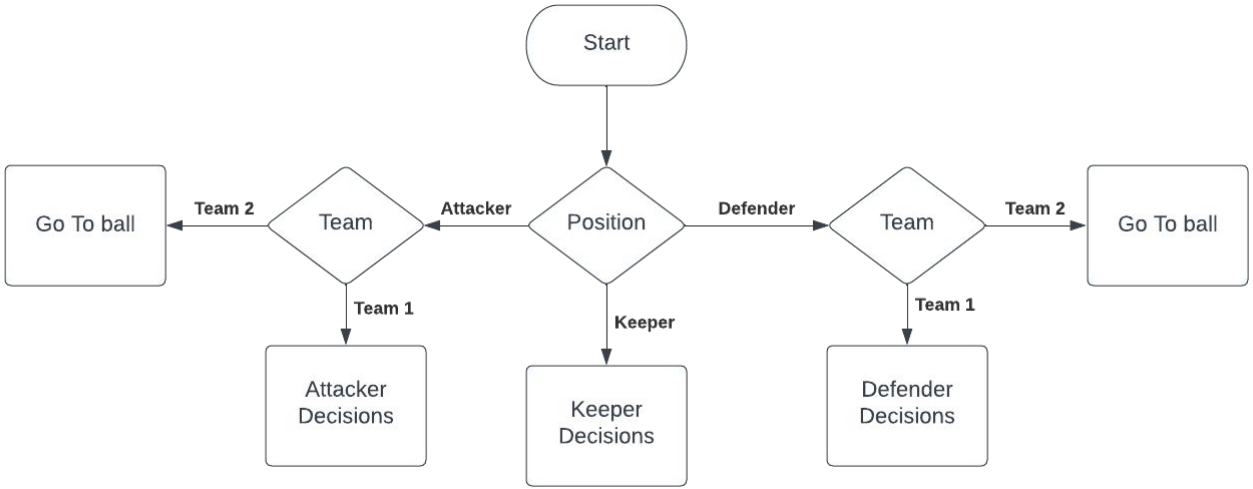
BallTracker Class

- Monitor Ball Position
- Judge the State of the Ball
- Update Team Score

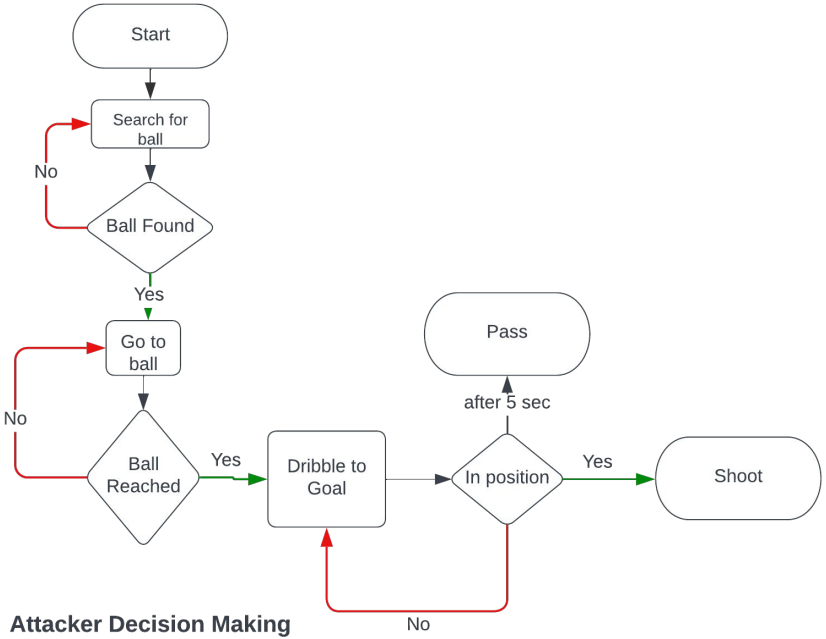
Importance

- Track Correctly
- Fairness and Accuracy

Decision making:

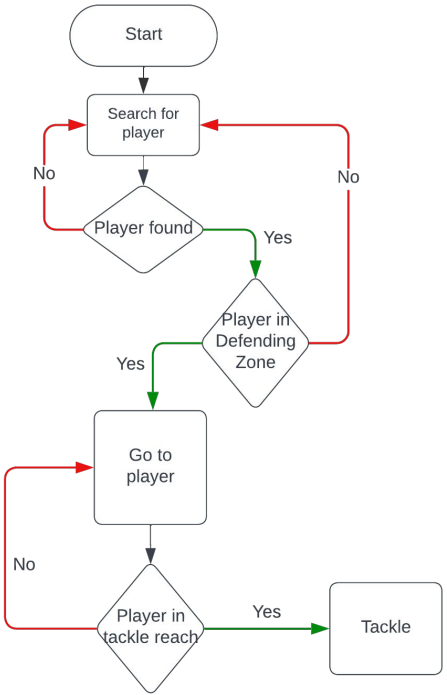


Attacker



Attacker Decision Making

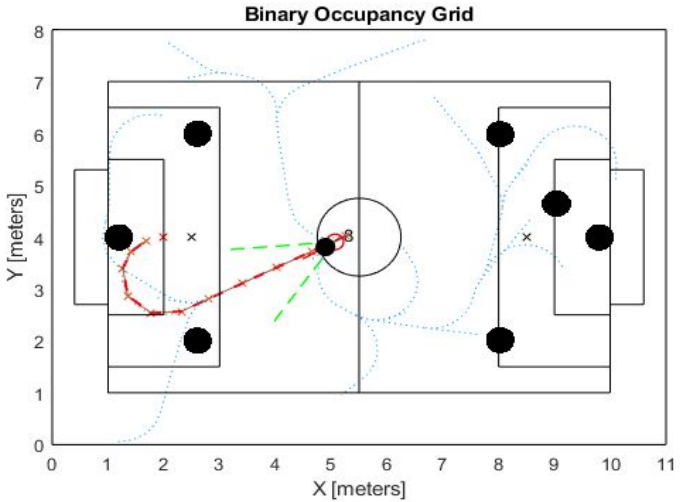
Defender



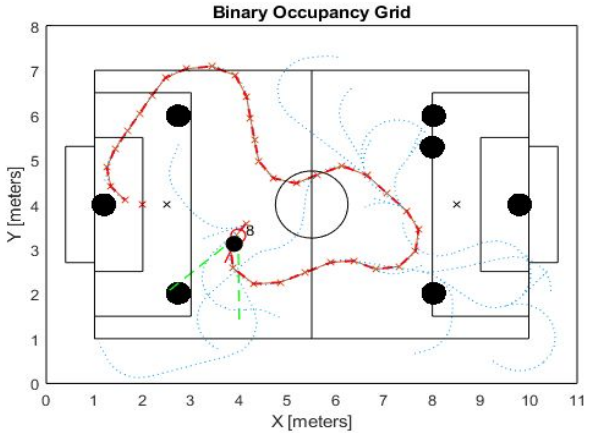
Defender Decision Making

Path Planning - RRT

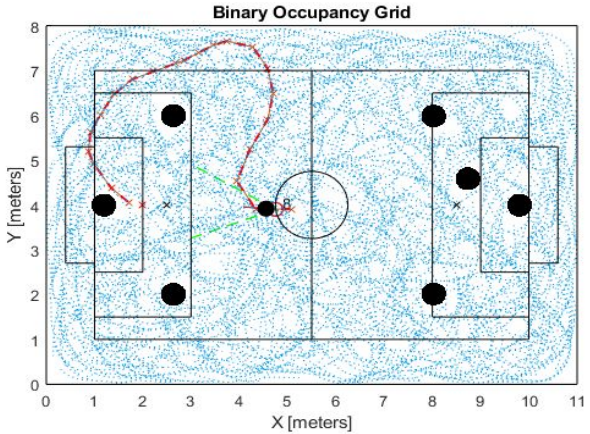
This builds a tree by sampling and connecting random points. Eventually finding a feasible path.



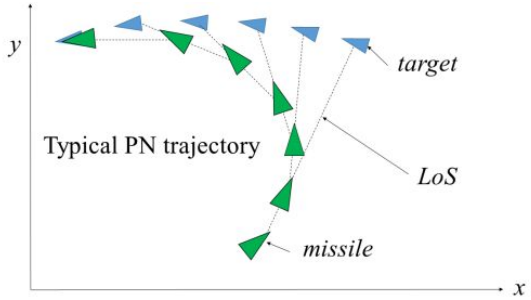
The path is not always optimal...



Not always very efficient...



Path planning - PN Target - Interceptor



1. Calculate the Line-of-Sight (LoS) vector:

LoS = target_position - interceptor_position

2. Calculate the relative velocity:

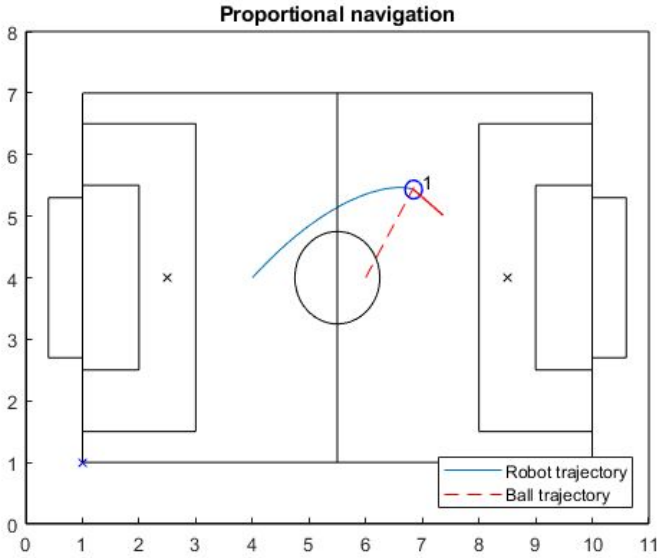
relative_velocity = target_velocity - interceptor_velocity

3. Estimate the Line-of-Sight (LoS) rate:

LoS_rate = (LoS_vector × relative_velocity) / |LoS_vector|^2

4. Compute the acceleration command:

acceleration_control = N × LoS_rate



Validation

- Visualisation
  - Robots, ball, pitch
  - Optional visualizations
- Metrics - benchmarking
  - Time record
  - Number of goals: offensive style
  - Number of outs: defensive style
  - Average number

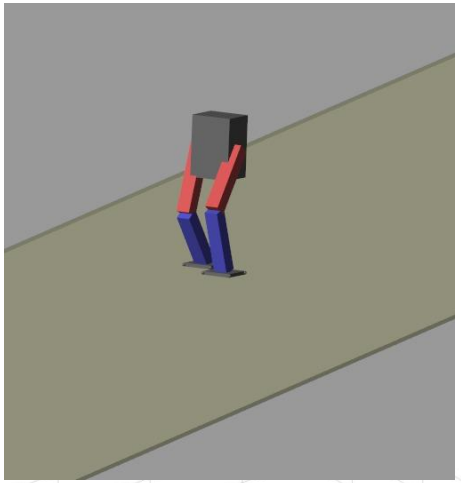
Team performance analysis

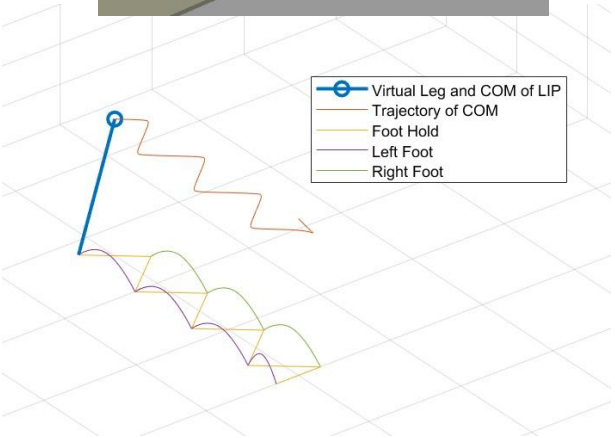
Name	Time
Oscar	113.25 hours
Alvaro	97 hours
Abuakar	93.5 hours
Sanbok	96.5 hours
Shiyi	75 hours
Jiwei	74.5 hours
Total	529.75 hours

Costs		
Staff costs	Expert costs	Total costs
£54, 975	£12, 000	£66, 975

Improvements - Model representation

- **NAO**
  - Use Kinematic model rather than differential drive
  - Follow path calculated by this simulation.
  - Control using voltages
- **Ball model** - spin mechanics, bouncing
- **Collision checks**: Could include direction and mass as well as 3D space.





Improvements - Algorithms

Path planning

RRT

- More optimal paths
  - RRT* can refine the generated paths, improving their optimality over time
- Faster convergence
  - Informed RRT* - Using heuristic information such as Euclidean distance, Potential fields, learned heuristics using reinforcement learning

Target-Interceptor

- **Real world**
  - Simulate sensors measurements of the ball
- **Improved Robustness**
  - Adaptive control to compensate for uncertainties and nonlinearity in the system.

Improvements - Decision making

- Inter robot communication
  - Give ability for teamwork
  - Additional information - ball state, other robots
- Prioritisation
  - Weighted decisions
  - Neural network
  - Fuzzy logic
- Benchmarking teams
  - Compare metrics
  - Run multiple simulations, averaging

Improvements - Team work

Coding practices

- Define code standards
- Set up a Continuous Integration pipeline to automatically run unit tests
- Improving documentation

Effective Teamwork Strategies

- Accountability
  - Working in groups of 2
  - Setting milestone for temselves
- Enhance Communication
  - Shorter meetings, more often
  - Making sure everyone understands

In conclusion

- All tools are present for a simple 2D simulation although a full game is not yet played...
- We chose to focus on algorithms which means that behaviour can be modeled more generally using these
- Next step is to iterate over the current simulation implementing the improvements.

Video