NEAR EAST UNIVERSITY

INSTITUTE OF GRADUATE STUDIES

DEPARTMENT OF COMPUTER ENGINEERING

Behaviour Tree Based Control of Omnidirectional Robots

M.Sc. THESIS

Nurullah Akkaya

Nicosia

September, 2021

NEAR EAST UNIVERSITY

INSTITUTE OF GRADUATE STUDIES

DEPARTMENT OF COMPUTER ENGINEERING

Behaviour Tree Based Control of Omnidirectional Robots

M.Sc. THESIS

Nurullah Akkaya

Supervisor

Prof. Dr. Rahib Abiyev

Nicosia

September, 2021

Approval

We certify that we have read the thesis submitted by Nurullah Akkaya titled **“Behaviour Tree Based Control of Omnidirectional Robots**” and that in our combined opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Educational Sciences.

Examining Committee	Name-Surname	Signature
Head of the Committee:	Assoc. Prof. Dr. Boran Sekeroglu	
Committee Member*:	Asist. Prof. Dr. Elbus Imanov	
Supervisor:	Prof. Dr. Rahib Abiyev	

Approved by the Head of the Department

5/1/2022

Prof. Dr. Rahib Abiyev
Head of Department

Approved by the Institute of Graduate Studies

…../…../20…

Prof. Dr. Kemal Hüsnü Can Başer
Head of the Institute

Declaration

Acknowledgments

I would like to emit my appreciation to my supervisor Prof. Dr. Rahib ABIYEV for wisdom and guidance. His aids were critical for my thesis.

I would like to thank my love, my family and my friends to give me personal support, and for their patience and kindness. Without them, everything would be harder.

A special thanks to my colleagues at the Near East University Robotics Laboratory for their support and motivation.

Nurullah Akkaya

Abstract

Behaviour Tree Based Control of Omnidirectional Robots

Akkaya, Nurullah
MA, Department of Computer Engineering
01, 2022, 65 pages

This thesis presents novel behaviour tree based control and improved path finding algorithm for efficient navigation of holonomic robots used in a soccer team. The hierarchical behaviour tree based control algorithm is proposed for decision making of robot to reach navigation goals. Behaviour tree (BT) has high and low level behaviours and its nodes are operating using certain behaviour rules given in the paper. High level behaviours are implemented using low level behaviours. The new behaviours of soccer robots are designed using tree structure. The use of BT approach allows to model complicated situations easily that show advantages of this technique over finite state machines and Petri set widely used in robot control. After decision makings using BTs, a path finding module is used to determine the path of robot. The next propose of the paper is, the design of an improved path finding algorithm.The widely used path finding methods are basically based on artificial potential field method, vector field histogram and A star methods that needs certain time for finding feasible path and sometimes do not complete in reasonable short time for real-time operation. Another commonly used algorithm to solve this problem is the Rapidly Exploring Random Tree (RRT) algorithm that quickly finds a feasible solution. However, frequently the RRT algorithm alone does not return better results in path length. Here, the integration of RRT with path smoothing techniques is developed. The obtained simulation and experimental results show that the constructed navigation system of soccer robots efficiently finds desirable and feasible solutions in short amount of time.

Key Words: behaviour trees, fuzzy logic, path finding, obstacle avoidance

Table of Contents

List of Figures

List of Tables

CHAPTER I

Introduction

Holonomic robots are increasingly utilized in industrial and academic settings. Mobile robots of this type can move independently in multiple degrees of freedom. The ability to move in multiple degrees of freedom makes the holonomic robots suited to applications in dynamically changing environments [1, 2]. One example of such application is mobile robots used by the small sized league (SSL) teams that compete in annual RoboCup competition.

Because of this, holonomic robots are actively researched for different academic and industrial applications [3]. Holonomic robot control is a challenging and important problem that deals with the designing systems that controls the movement of mobile robots.

The primary aim of mobile robot motion planning is to steer a robot towards a direction, towards a goal position while avoiding collision with obstacles in the environment while tracking its exact location in the environment. Mobile robots operates in environments which may contain fixed areas containing non moving obstacles to dynamically changing areas due to moving obstacles.

Various control algorithms have been implemented in the past that enables navigation of mobile robots in dynamically changing environments. One frequently used solution for this problem is the forming of a world model of the environment the robot is in using the information gathered from the sensors, make choices about the new goal positions for the robots, and apply navigation algorithms that moves the robots to their new designated positions while avoiding collision with the obstacles [4, 5].

As shown in Fig.3 Control and navigation system for soccer robots is made up of modules. System is made up of various subsystems, vision subsystem, decision making subsystem, path planning and collision avoidance subsystem, and various control algorithms. The vision module uses SSL vision system which is the shared vision system of RoboCup Small Size League (SSL). Decision making module implements the artificial intelligence system for mobile robots. This module is responsible for analysing the current state of the environment and make strategic decisions that moves the robots to new locations and plans a collision free path for the mobile robots. In the literature various approaches have been implemented in order to develop decision making system for soccer robots. Some of these are, finite state machines (FSM) [5, 6, 7], Petri nets [8, 9], behaviour based control [10, 11, 12] and topological graph map [13].

In this thesis our first motivation is to design a hierarchical decision making system based on Behaviour Trees (BT). BT based approach simplifies the decision making process that involves complex and parallel logic by using higher level abstraction nodes that are composed of lower level nodes for decision making for mobile robots. Higher level nodes are composed of intermediate/lower level nodes. Each non-leaf node in the behaviour tree has a predefined role for performing control on their child nodes. Leaf nodes in the tree are called action nodes that perform various operations on the robots. Depending on the problem at hand the proposed BT based decision making system can easily be extended and adapted.

In mobile robot control an important problem motion planning which is the process of finding a collision free path to target. Given a map of the world and a target location to navigate to, motion planning is used to search for a path from the robots current location to another target location via a set of intermediate waypoints. Often times, the world/environment where robot operates in is not fixed, i.e., during motion of the robot, it has to deal with other dynamic obstacles or robots, consequently execution is often associated with unpredictability. For collision free navigation to target, the path planning algorithm has to be used in conjunction with an obstacle detection and collision avoidance algorithm.

Various techniques for mobile robot navigation using motion planning combined with collision avoidance have been developed. Frequently used techniques are, Vector Field Histogram (VFH) Technique [14], Artificial Potential Fields (APF) [15], fuzzy navigation techniques, local navigation, Rule Based Methods [16] and Rules Learning Techniques [17], Dynamic Window Approach (DWA) [18], VFH+ Technique [19], Agoraphilic Algorithm [20], Search Trees [21, 22, 23, 24, 25], A star (A*) [26] etc.

In the thesis our next motivation is to design an efficient algorithm that finds a feasible close to optimal navigation route in a short amount of time. For implementing an efficient and fast path finding system the use of a path optimization algorithm is implemented that shortens the path of the robot using the environment information.

Organization of this thesis is as follows. In Section 4, decision making based on BTs and the design of BT based control system for mobile soccer robots is explained. In Section 5, the implementation of an efficient and fast path finding algorithm is presented.

CHAPTER II

State of Art of Control System Mobile Robots

Decision Making

Decision making (DM) module implements the artificial intelligence system for mobile robots. This module is responsible for analysing the current state of the enviroment and make strategic decisions that navigates the robots to new locations using a collision free path. Various different approaches have been considered in the literature in order to develop decision making system for mobile robots.

Finite state machines (FSM) is a commonly used approach for mobile robot control [5, 6, 7, 27]. FSM approach makes use of input or event information to determine the state of the FSM model.

In a finite state machine each robot occupies a state. Behaviors and actions are associated with each state. So, while the robot remains in a particular state, it will continue executing the same action. FSMs goes through their states by using transitions Each transition makes the FSM switch from one state to another based on a set of associated conditions. When the FSM determines that the conditions of a transition are met, then the FSM switches state to the transitions target state. A FSM is said to be triggered when a transitions conditions are met, a FSM is said to be fired when the transition is followed to a new state.

Modelling complex behaviours with hierarchical FSMs has expressiveness but representing complex logic increases the number of states and transitions between states needed to represent that logic. Modelling these situations becomes a complex and error prone process when using finite state machines.

Petri nets are favoured over FSMs because of their larger modelling ability and compared to FSM same state space can be modelled using a smaller

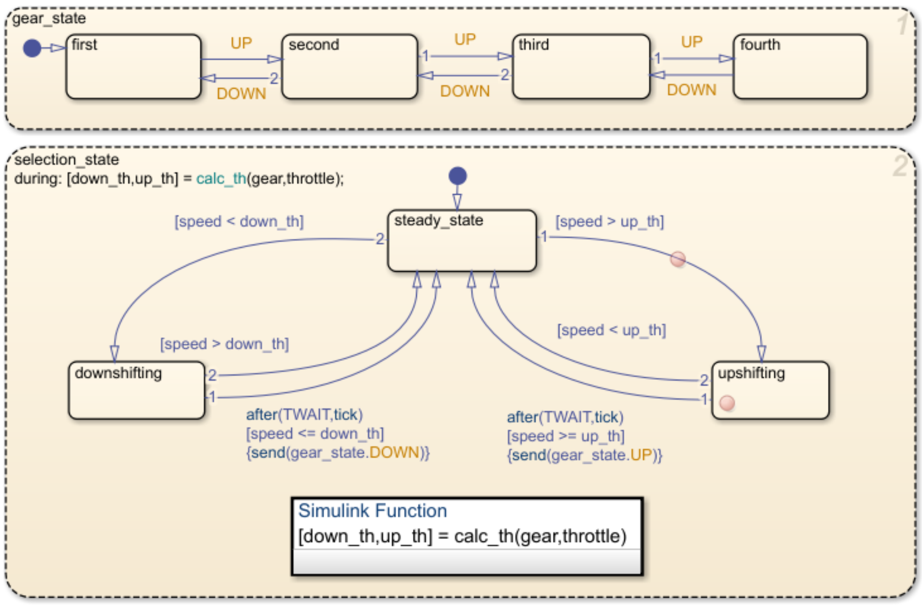


Figure 1: A FSM modelling a car gearbox.

graph [8, 9].

However when Petri nets are composed it leads to to an exponential growth in the state space. Petri nets allows modularity in behaviour logic, as each behaviour can be implemented independently and later combined with other behaviours. But for some situations Petri nets can become too large to represent all of the states of the system which makes the analysis of the system difficult.

In this thesis our first motivation is to design a hierarchical decision making system based on Behaviour Trees (BT). BT based approach simplifies the decision making process that involves complex and parallel logic by using higher level abstraction nodes that are composed of lower level nodes for decision making for mobile robots. Higher level nodes are made up of intermediate/or low level layers. Each non leaf node in the tree has a predefined role for performing control on their child nodes. Leaf nodes in the tree are called action nodes they perform operations such as kicking motor controller etc. on the robots. Depending on the situation the described BT based algorithm can easily be extended and adapted to various situations.

Motion Planning and Collision Avoidance

In robot navigation motion planning is an important task. Given a representation of the environment and a target location to navigate to, the path planning or motion planning is used in order to find a set of waypoints that takes the robot from the robots current coordinate location to a target location. Frequently, the world which robot navigates in is not fixed, while the robot navigates it has to deal with with other moving obstacles or robots so execution is often associated with unpredictability. For a collision free navigation to the goal, the motion planning algorithm has to be used in conjunction with an obstacle detection and collision avoidance algorithm. Various methodologies have been developed and used for mobile robot navigation using motion planning and collision avoidance. Some methods that are frequently used are, Vector Field Histogram (VFH) Technique [14], Rule Based Methods [16], Agoraphilic Algorithm [20], Artificial Potential Fields (APF) [15], Dynamic Window Approach (DWA) [18], fuzzy navigation techniques, local navigation, VFH+ Technique [19] and Rules Learning Techniques [17], A star (A*) [26], Search Trees [21, 22, 23, 24, 25] etc.

The Artificial Potential Fields (APF) is another commonly employed method in mobile robot navigation that it makes use of attractive and repulsive fields [5] for target locations and obstacles respectively, these fields generates forces on the mobile robot. These forces when combined pushes the robot towards a target location and away from the obstacles.

This algorithm can be used effectively in simple environments however in complex environments potential field algorithm has several disadvantages,

- the generated field may contain a local minima which can cause the mobile robot to get stuck, this is especially a problem when operating in an environment with a lot of obstacles.

- the mobile robot may be pushed away from the target by a moving obstacles.
- the generated field may cause oscillations in mobile robot motion.
- the mobile robot unable to pass through small gaps

Vector Field Histogram (VFH) [14, 19] is another method used for motion planning it makes use of a two dimensional Cartesian histogram grid as the environment and combines it with the concept of potential fields. The VFH algorithm is able to produce a detailed spatial representation of the robot's environment sufficient for navigating among densely populated environments. [14, 19, 28, 29].

DWA developed by Fox [18] is used to navigate the mobile robot using a model of valid actuations achievable by the robot. By considering all the possible actuations achievable by the mobile robot, selecting and applying those that steers the robot away from other obstacles, the robot is able to navigate through the environment.

In case of Agoraphilic Algorithm [20], a force called an attraction is generated by the free space and this force keeps the robot moving towards the target.

In Rule-based algorithms [16] simple rules of some form are used. Rule based systems are easy to use, but they are often designed for specific enviroment they are used in thus has limited to no use outside this enviroment.

A* algorithm is another widely used method. A* is built upon Edsger Dijkstra's algorithm [30]. A* uses breadth-fist search to find a optimal path from start node to end node by using various heuristics depending on the environment. [26]. The computational complexity of A* depends on the cost function being used.

Fuzzy logic, genetic algorithms (GA) and Neural networks (NN) are widely used in motion planning [29, 31, 32, 33, 34, 35] . Fuzzy navigation is based on the construction of fuzzy rule base by a domain expert or obtained after trial-and-error experiments. Approaches based on GA and NN requires time for learning the parameters for the controllers.

Most of the environment where robots navigate is identified with uncertainty, moving obstacles and fast-changing dynamic areas. Soccer playing robots are operating in an environment where there are moving obstacles in the form of other soccer playing robots.

Control system for soccer playing robots must solve above mentioned following set of problems.

- Computer Vision
- Decision Making
- Motion Planning
- Velocity Control

The control system for soccer robots must overcome all these problems in fraction of a second, usually 33 ms. Using the above mentioned algorithms soccer playing robot motion planning can not be handled in acceptable time frame for real-time operation of soccer playing robots. In this case instead of finding an optimal path, finding a close to optimal robust path in short time becomes feasible.

Rapidly-exploring Random Trees (RRTs) is an motion planning algorithm developed by Kuffner and LaValle and can be used to find a suboptimal path in very short amount of time. Which makes it ideal to be used for robot motion planning in dynamic environments [22, 23, 24].

RRT Connect is an improvement over RRT, compared to RRT algorithms single tree RRT Connect uses two trees one at the start and another at the goal states and grows them towards each other. [22]. RRT efficiently solves the motion planning problem, it is more efficient than search based before mentioned algorithms even in high dimensional problems. [22, 23, 24, 25, 36, 37].

But RRT algorithm does not guarantee to find a optimal motion plan, and at times the plan generated can be very suboptimal. Various algorithms have been designed in order to improve RRT.

In [38] authors optimize the length of path by increasing sample size. The authors proposed RRT* an extension to the rapidly-exploring random graph (RRG) and RRT algorithm. Using results from random geometric graph theory, these methods keep the asymptotic computation complexity of RRTs under certain conditions. The algorithm does large number of iterations in order to achieve optimal path, which results in more computation time.

In [39] authors proposed an algorithm, that reduces the state space explored by adding edges to the current roadmap in order to enable finding higher quality paths. This algorithm reduces the number of collision checks while still keeping the optimality guarantees.

In the thesis our next aim is to design an efficient navigation algorithm that is able find a feasible non-optimal path in short amount of time. To implement an efficient navigation system a path smoothing algorithm is implemented that allows us to optimize the given calculated motion plan of robot using the local environment state.

Problem Statement

BT based decision making system along with motion planning using RRT-Plan and path optimization motion planning algorithms are used for the control of the holonomic soccer playing robots used for RoboCup Small Size League by the NEUIslanders robotic football team. The robots are manufactured and assembled by NEU robotics laboratory. Here the problem is making strategic planning using the current state of the game and the current locations of robots and decide for each robot new goal positions and navigate each robot to its new destination location.

CHAPTER III

Structure of Control System of The Omnidirectional Robots

NEUIslanders is a RoboCup Small Size League team that lunched at the applied artificial intelligence laboratory of the Near East University. The team has been an active member of RoboCup Small Size League since 2012. The team has various achievements through out its lifetime such as,

- 3rd place in 2016 European Championship
- 1st place in Division B RoboCup Championship in Canada,2018.

Vision System

Perception system of soccer robots relies on a centralized, shared vision system, called SSL-Vision [40]. The vision system depicted in Fig.2 is designed for four cameras that are placed at the top of the soccer field. SSL-Vision software is used for tracking the ball and the robots on the field. Detection results are broadcasted to the teams using UDP protocol. Google Protocol Buffers is used to encode Data packets. SSL-Vision uses multiple overhead cameras however it does not carry out any sort of sensor fusion, fusing the detection results coming from multiple cameras are left to individual teams. SSL-Vision also does not carry out any sort of motion tracking or smoothing, individual teams are responsible for implementing their own tracking and smoothing algorithms. In the case of NeuIslanders besides what vision server sends us, system also keep track of certain other things such as,

- Robot speeds.

- Robot headings.
- Ball speeds.
- Ball headings.

For simulation we rely on grSim. Receiving data from grSim is exactly like receiving from SSL-Vision using Google Protobuf library. Sending data to Simulator is also possible using Google Protobuf.

Detection results arriving from SSL-Vision use a standardized coordinate system (Cartesian). The x and y axes corresponds to horizontal and vertical axes respectively. Detection results sent from SSL-Vision to team computers contains the locations robots on the soccer field, dimensions of the field and the location of the ball. Each team is responsible for their analysis of the data received from the SSL-Vision system for their decision making.

Both SSL-Vision and grSim use a standard cartesian coordinate system in meters and radians. Pretend you are looking down on a robot: +x goes out to your right (East), +y goes up (North). The center of the soccer field is (0,0). When the robot is initialized, it is facing in the direction of the opponents goal. Headings are given in radians, with East = 0, North = /2, South = -/2 and so on.

Control System

The RoboCup Small Size League (SSL) is a fast-moving, dynamic environment suited for cooperative multi-robot research. In the Small Size League, robots are able to travel on the game field in less two seconds, which requires algorithms that are capable of coordination, decision making and motion planning in fractionsof a second, most teams usually operate at 30 Hz.

As depicted in Fig.3 control system is made up of following modules.

- World

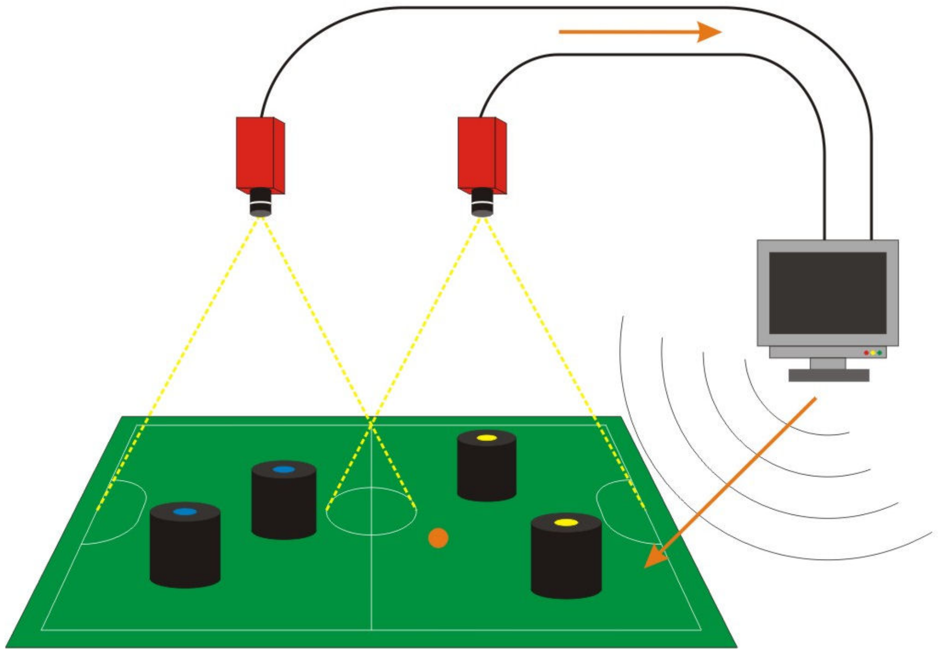


Figure 2: SSL Vision Architecture

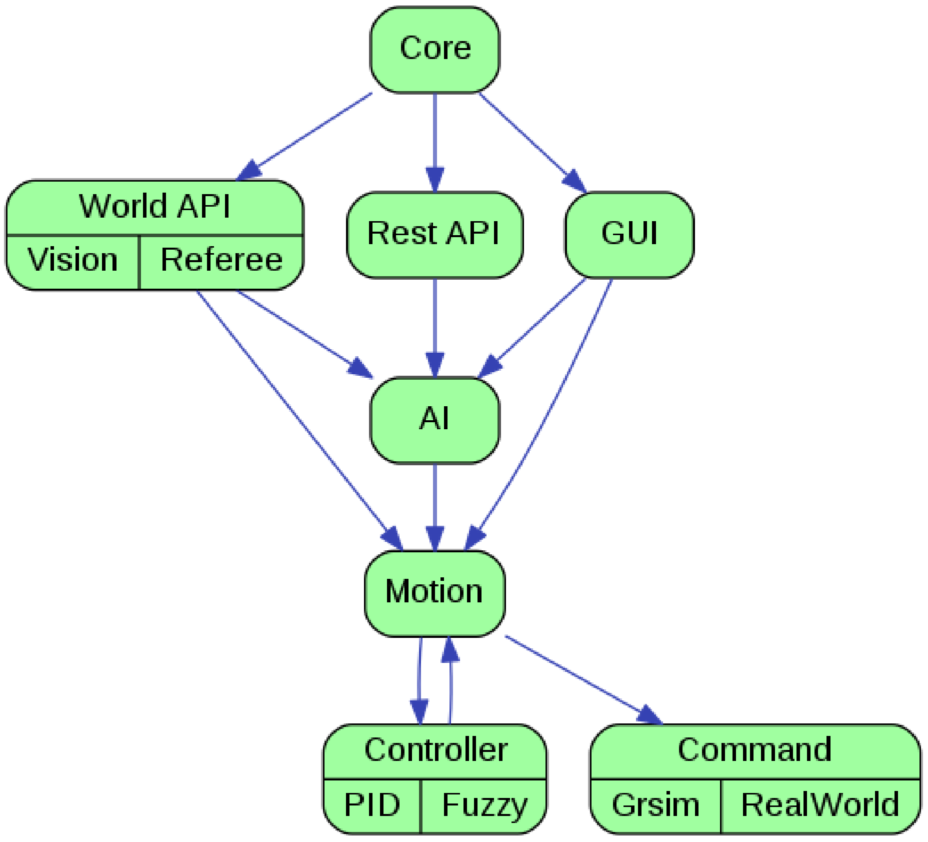


Figure 3: NEUIslanders Module Structure

  - Keeps track of the state of the world.
  - Handles communication with SSL-Vision, grSim and the Referee system.

- Motion - Responsible all low level robot movement.
- Controller - Provides a common interface that supports various types of controllers. Currently supports PID and Fuzzy Logic Controllers.
- Control - Low level commands are sent to the robots via the 'control' interface, AI doesn't know if it is controlling virtual robots in a simulation or real-life robots on the field, it just sends commands, this interface dispatches them to correct underlying implementation depending on the situation.
- DM - Main Playbook.
- UI - User Interface.

Once a decision is made by the central decision making system, the commands are dispatched to the robots via a wireless connection to the micro-controllers on the mobile robots. Depending on the command received robots are capable of executing various actions. i.e kick the ball by discharging a capacitor or move the a destination position by regulating the rotational speeds of its four omni-wheels.

Fig.4 shows how the data flows through the NeuIslanders AI system. SSL-Vision software is used to track the robots on the field along with the ball. This detection data is received by the tracker module which handles motion tracking of objects on the field. Tracker module is responsible for keeping the world model updated with locations of home and away team robots DM system uses this world model to find new locations for soccer robots to move to. A path is calculated that will navigate the robots without collision to their destination. Once motor velocities are calculated they are sent to the robots via wireless link.

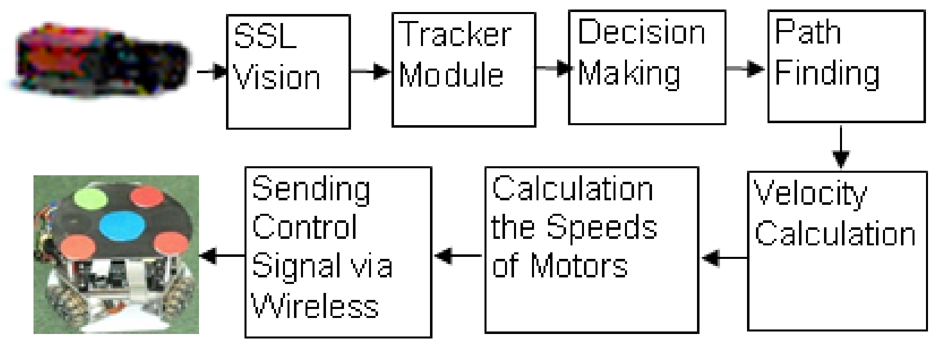


Figure 4: Data flow through NeuIslanders control system

Hardware

The robots has diameter of 175 mm with a height of 145 mm. Designed robots uses holonomic wheels which gives them 3 degrees of freedom. Robots are able move in both X, Y axes and rotate simultaneously. Fig.8 depicts the latest generation of our robots. In order to let the robot move in any direction without turning robot is equipped with four holonomic wheels each has a diameter of 61mm. Wheel orientation drastically effects the performance of both the speed and acceleration of the robot. 3mm thick 7075 aluminium is used for the chassis. The wheel orientation for the current generation of robots robot is 33 degrees for the front wheels and 45 degrees for the rear wheels. Holonomic wheels are driven by brushless DC motors and brushless DC motor drivers. Brushless motors of the robot are connected to the wheels via gear mechanisms and controlled by Electronic speed controllers which are driven by a onboard microcontroller. Each robot has a microcontroller board is responsible for controlling various control circuits on board such as motors and kicking mechanism circuit. The main processor for the robot is a The

Teensy 3.6 (Fig. 6) Microcontroller communicates with the computer using wireless link based on XBee Radios. Fig.7 depicts the soccer robot control structure.

Dimensions	Ø178x145mm
Weight	3150gr
Driving Motors	Maxon EC-45 Flat 30Watt with 2048ppr Encoder
Driving Gear Ratio	72:20
Dribbler Motor	Maxon EC-16 15Watt
Dribbler Gear Ratio	24:48
Kick Speed	Up to 8m/s (electronically limited)
Communication	XBEE 60 mW

Figure 5: NEUIslanders Robot Specifications

Robots have 3 degrees of freedom, robots move by varying the angular velocity of the each wheel. This allows control of rotation and velocity of the robot. With the help of holonomic wheels robots can at any angle, move in any direction. Robots are able to change their orientation and velocity separately. This gives the robots the ability such as turning in place, moving in any direction regardless of the initial orientation. For an holomonic robot to move at a certain velocity, each wheel needs to rotate at a particular speed in comparison to the other motors. The intended direction of travel of the robot is calculated by AI system in terms of V_x, V_y and R_w. Multiplying velocity coupling matrix with our intended velocity results in a sequence of motor speeds. The formula used to calculate the angular speeds for each wheel for a robot that has n=4 wheels, is given below,

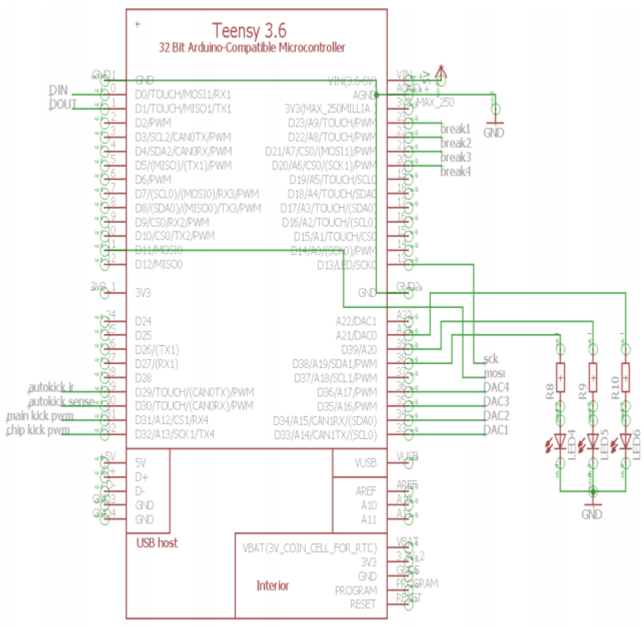


Figure 6: Main Processor Schematic

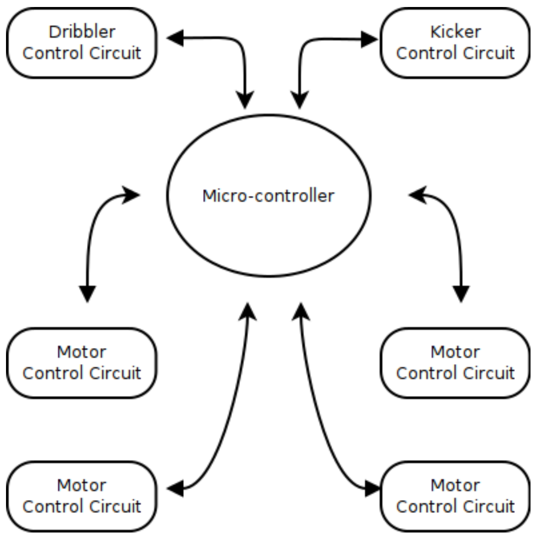


Figure 7: Control Structure of the Robot

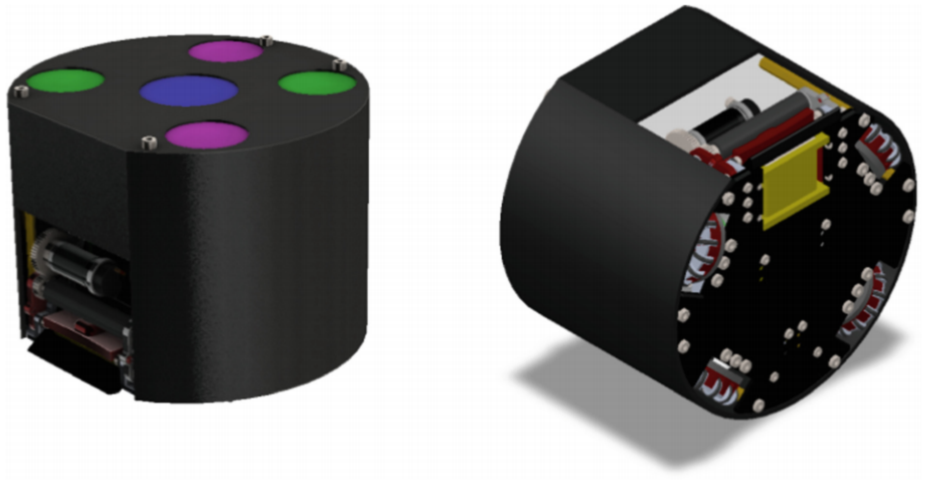


Figure 8: Latest Generation of NeuIslanders Robots

$$\begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix}^T = \begin{bmatrix} -sin\theta_1 & cos\theta_1 & 1 \\ -sin\theta_2 & cos\theta_2 & 1 \\ .. & .. & .. \\ -sin\theta_n & cos\theta_n & 1 \end{bmatrix} \times \begin{bmatrix} v_x & v_y & R_W \end{bmatrix}^T$$

- v_1, v_2, v_3, v_4 - represents individual motor speeds.
- v_x and v_y - robot velocity in horizontal and vertical direction.
- R_W - angular speed.

Motion module calculates required velocity in meters/sec, a conversion to radians/sec is required,

$$W = \frac{2 \times v}{d} \tag{1}$$

- W - Speed - Angular ($\frac{rad}{sec}$)
- v - Speed - Linear ($\frac{meter}{sec}$)

- d - Wheel Diameter (meter)

CHAPTER IV

Behaviour Tree Based Control

Desicion making system used for NeuIslanders is implemented using the concept of behavior trees. BTs are a method of arranging collections of robot behavior and the decision processes of how to move between them. BTs shares a lot of similarities to FSMs but unlike a FSM they are very easy to compose, fast to execute, lets the programmer see and maintain the logic easily, which makes them ideal for implementing complex and parallel decision making process.

Behaviour tree

Behaviour trees combines a number of decision making techniques that have been used in artificial intelligence,

- Hierarchical State Machines
- Planning
- Scheduling
- Action Execution.

Behaviour trees power comes from it's ability to interleave the logic in a way that is easy to create and maintain. Behavior trees have commonalities with hierarchical FSMs which uses, states as the main building block, BTs use nodes as the main building block. Nodes are combined and composed into trees in order to represent progressively complex behaviour. Later, these behaviours can also be composed together into more complex hierarchical trees. This composability is what gives behavior trees their power since all nodes share a common execution interface and mostly self-contained, this allows BTs to be easily composed into hierarchies of trees without having to worry about implementation details of how each subtree in the hierarchy is works.

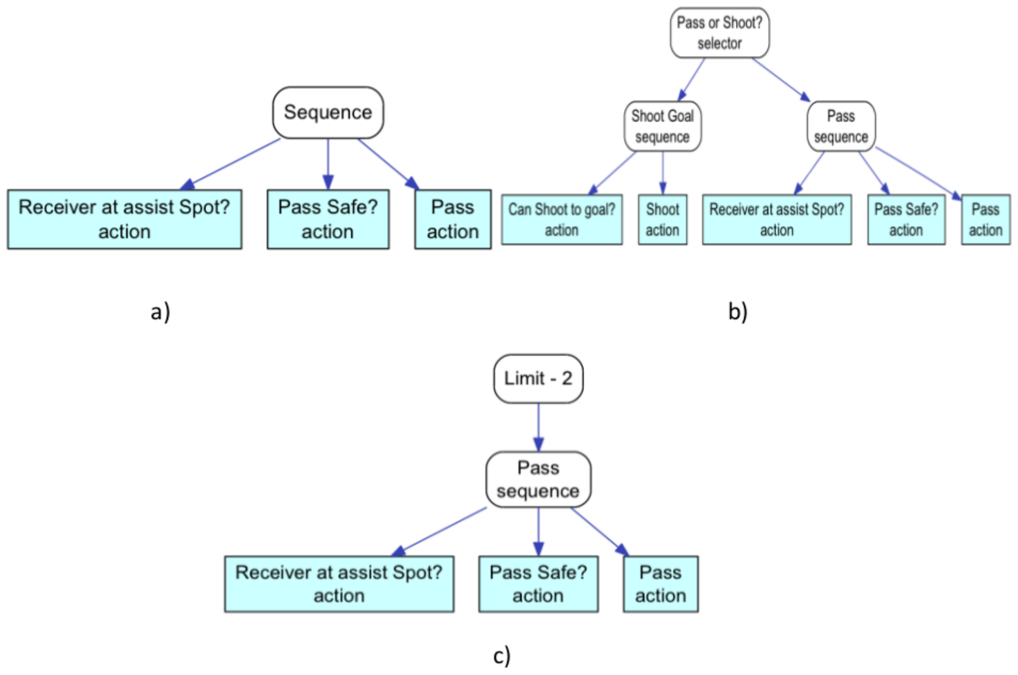


Figure 9: Various Node Types: (a) Sequence, (b) Selector, (c) Decorator

Modularity in a Bt is enabled by making behaviours composed within each other turning them into a tree structure and only allowing transitions only to these child nodes. The root node is traversed until leaf nodes are reached. These leaf nodes are where side effects are handled such as interacting the world model, controlling actuators on the robots etc.

BTs have three primary node types, action, decorator, composite as depicted in Fig.9. Decorator and composite nodes are primary means of control flow within the tree. Leaf nodes are used for side effects such as motion planning or motor control. Composite nodes are made up of selectors, sequences, and their parallel and random versions.

Interface

All nodes of a BT implements a common interface. This allows arbitrary nodes to be composed with each other without any of nodes needing to know what else the tree is made up of.

```
(defmulti exec
  "Given a node dispatch to its exec implementation."
  (fn [node & [terminate?]]
    (node :type)))
```

Sequence

Sequential logic is defined using sequence nodes. Sequence nodes represents sequence of behaviours that needs to be completed. Sequence nodes traverses its children from left to right. In order for the sequence to succeed all of its children must succeed. If any one of the children fails to execute, sequence will stop without running rest of its children and return failure (Fig.9(a)). Following is an example of how a sequence node may be implemented.

```
(defn- seq-run [children terminate?]
  (if (exec-action? terminate?)
    (if-let [s (seq children)]
      (if (exec (first s) terminate?)
        (recur (rest s) terminate?)
        false)
      true)
    false))
```

Fig.9(a) depicts what a pass sequence might look like, first we check if the receiving robot is at the determined assist position and ready to receive a pass, then we check whether a pass can be executed safely. When both of these conditions are satisfied, tree will execute the pass, depending on the results of pass tree will return success or failure up the tree.

Selector

Primary selection mechanism within the tree is the selector node. Selector node tries to execute its children in order and as soon as one of its children returns success (true) selector itself will return success, otherwise if no children returns success selector will return failure(false). (Fig.9 (b)). This property is used to decide how the tree is iterated and which behaviour is choosen to run next. The example tree in Fig.9 (b) is used by the DM to select between executing a pass or shooting at the goal directly. Selector node tries to execute its child nodes from left to right starting with the Shoot Goal sequence. If the sequence containing Can Shoot to goal? and Shoot returns success, the selector node will succeed. If the Shoot Goal sequence fails, we try to executing a Pass sequence. If one of the two sequences succeeds selector will succeed if both sequences fail selector will fail.

```
(defn- select [children terminate?]
  (if (exec-action? terminate?)
    (if-let[s (seq children)]
      (if-not (exec (first s) terminate?)
        (recur (rest s) terminate?)
        true)
      false)
    false))
```

Decorator

Decorator nodes are named after the software design pattern. Fig.9(c). Unlike other node types decorators have only one child. Decorator nodes is used to change how the branch behaves in various ways. Fig.9(c) depicts the same Pass sequence composed with a decorator as its root. In this case a limit decorator. When Pass is decorated with a limit decorator, it will try to execute a pass. If the pass succeeds tree will return success, when the tree fails, it will retry at most n more times (1 times for this tree) to execute the pass. If the pass does not succeed within n tries then tree will return failure. Following is an example of how various different decorator nodes may be implemented.

```
(defmethod exec :inverter [{children :children} & [terminate?]]
  (not (exec children terminate?)))

(defmethod exec :forever [{children :children} & [terminate?]]
  (loop []
    (if (exec-action? terminate?)
      (do (exec children terminate?)
          (recur))
      false)))

(defmethod exec :until-success [{children :children} & [terminate?]]
  (loop []
    (if (and (exec-action? terminate?)
             (not (exec children terminate?)))
      (recur)
      true)))

(defmethod exec :limit [{children :children times :times} & [terminate?]]
  (loop [i times]
    (if (and (pos? i)
             (exec-action? terminate?))
      (if (not (exec children terminate?))
      (recur (dec i))
      true)
    false)))
```

Simplest decorator node is the interter node which just reverses the return value. If the child returns true it makes it false and vice versa. Another usefull node is the forever node it will execute the child in an infinite loop without checking the return value. until-success on the other hand executes its children while it keeps failing and will return success when it succeeds.

Fuzzy Selector

Soccer robots are operating in dynamic, uncertain and unpredictable environments. Making decisions in this environment requires accurate evaluation. For cases such as this, fuzzy sets provides powerful tools for the representation of uncertain and vague data. A fuzzy inference system makes a decision by applying approximate reasoning.

In soccer playing robot control, fuzzy logic includes a range of degrees for decisions. These degrees are represented using membership functions. This node acts as a Selector node however it uses fuzzy inference for node selection. Each children of the node has a membership function associated.

For example, classical Move behaviour can be implemented using Run, Walk and Turn nodes (Fig.11(a)). Selection of these nodes are dependent on a set of conditions. Depending on the state of the environment the membership degree is determined for the each of the children. A fragment of the membership rules are given below.

```
If ball is near and ball speed is low then execute sequence walk
If ball is far away and ball speed is high then execute sequence run
If ball is far away and ball speed is very low then execute sequence stay
```

Figure 10: Fuzzy Selector Rule Base

The inputs and outputs of the fuzzy rule base (RB) are determined using performance characteristics of small size soccer robots using expert knowledge and experimental data. The implementation of the fuzzy-selector node is done using the formulas given below.

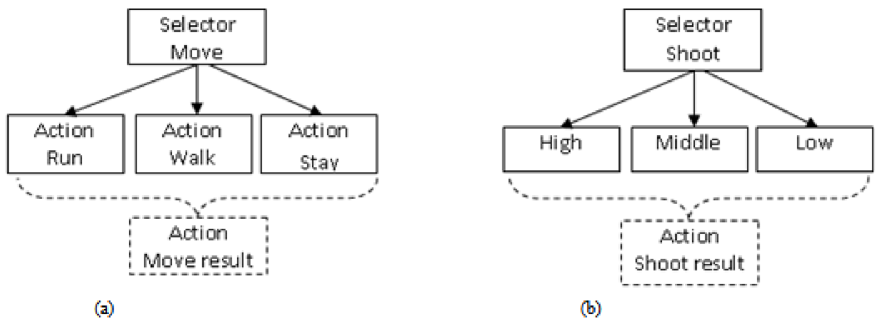


Figure 11: Selector BTs. (a) Move, (b) Shoot

Output of the behaviour tree is determined using the fuzzy rule base. A fuzzy inference engine is used to select the output of the tree. The inference engine used in the BT node is implemented using max-min composition.

The current values of input signals coming from the world model are entered in to the fuzzy rule base, after fuzzification, each input signals degree of membership to the current fuzzy term in rule base is calculated.

After calculating the degree of membership of the input signals for each active rules in the fuzzy rule base, fuzzy logic inference is performed using max-min composition of Zade. Centre of average method is used in the defuzzification process of fuzzy output signal.

Hierarchical Behaviours

Using above described architecture, complicated behaviours can be described by composing BTs. Soccer robots are controlled by combining lower level behaviours which handles simple tasks such as kick or intercept the ball or moving to a location, into more complex higher level behaviours such as passing, defending.

In order to keep the as modular as possible, high level strategies are split into three categories these are, Skills, Tactics, and Plays.

- Skills - These are trees that control a single robot. They may include trees like intercepting the ball, moving to a target position or executing a pass.
- Tactics - These trees coordinate a single or many robot and represent more complex behavior than before mentioned skills. These may implement things such as passing or defensive behaviours.
- Plays - These trees handles coordination of the whole team of robots. Contains trees such as kick off behaviour, defensive games and offensive games.

As shown in Fig.12 when combined, skills, tactics, and plays implement a tree with the Play at the root and other behaviors as its child.

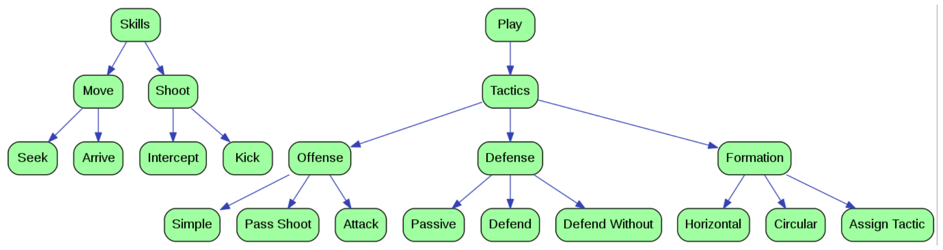


Figure 12: Plays & Skills

Each low-level skill implements a specific low-level goal:

- Move - moves robots around the field
- Pass - uses two robots to execute a pass

- Intercept ball - moves robot to a position to intercept the ball
- Defend Goal - goal keeper behaviour
- Shoot - shoots towards a target location
- Dribble ball - moves the robot while dribbling the ball
- Penalty taker - shooting that is specialized for penalties
- Penalty keeper - goal keeper specialized for penalties

Fig.13 shows the Pass tree. Pass tree controls two robots in parallel using a parallel version of sequence node which behaves exactly like a sequence node but runs its child nodes in parallel, tree aligns the passing robot and the receiving robot for a pass, then the tree checks if a pass can be executed safely, if a pass is safe passing robot shoots the ball and wait for the ball to start rolling, once that succeeds, tree waits until one of two conditions are met, ball gets within a robot diameter of the receiving robot or ball stops rolling at which point receiving robot moves to intercept the ball.

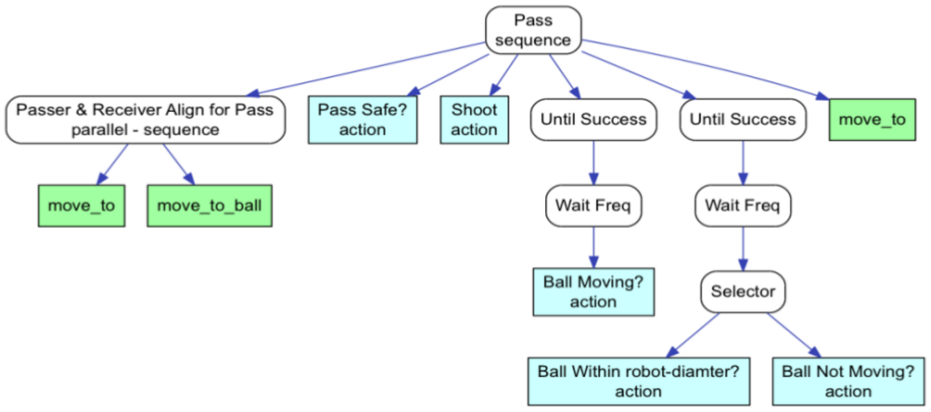


Figure 13: Pass behaviour tree

Skills and tactics are composed to form plays. Some plays include,

- Formations - moves the robots to various hard coded locations on the field.
- Offensive Game - A play that focuses on attacking the opponent team.
- Defensive Game - A play that focuses on defending against opponent team.
- Game Selection - Selects main play to execute depending on the game state. Fig.16

Fig.14 shows the three robot "Defensive Game" play. This tree uses a decorator node called =interrupter=. Interrupter node has three child nodes, one children waits for an interruption event, one child executes the primary tree and last child is responsible for cleanup in case the primary tree is interrupted. Interrupter run the main tree normally, if the main tree returns success it is passed up the tree. But if the main tree is executing and interruption event occurs primary tree will be terminated result of cleanup is sent up the tree. This decorator node is used to allow a BT to react to events happening during the game. At the top level shown tree waits for an Ball Out of Play? event, while the ball is in the field of game defensive game keeps executing but as soon as ball goes out of field, all robots are stopped.

Inner subtree uses another interrupter node to handle switching from a defensive tactic to an offensive tactic. The tree waits for a Home Controls Ball? event until home team takes possession of the ball, tree controls three robots and tries to take possession of the ball interposing the ball. When the ball is intercepted Home Controls Ball? event interrupts defensive play and executes Put Ball Back in Play tree and we can safely switch to offensive game play.

Fig.16 depics how main plays are selected within the tree. Fig.16 uses an interrupter node to wait for referee events.

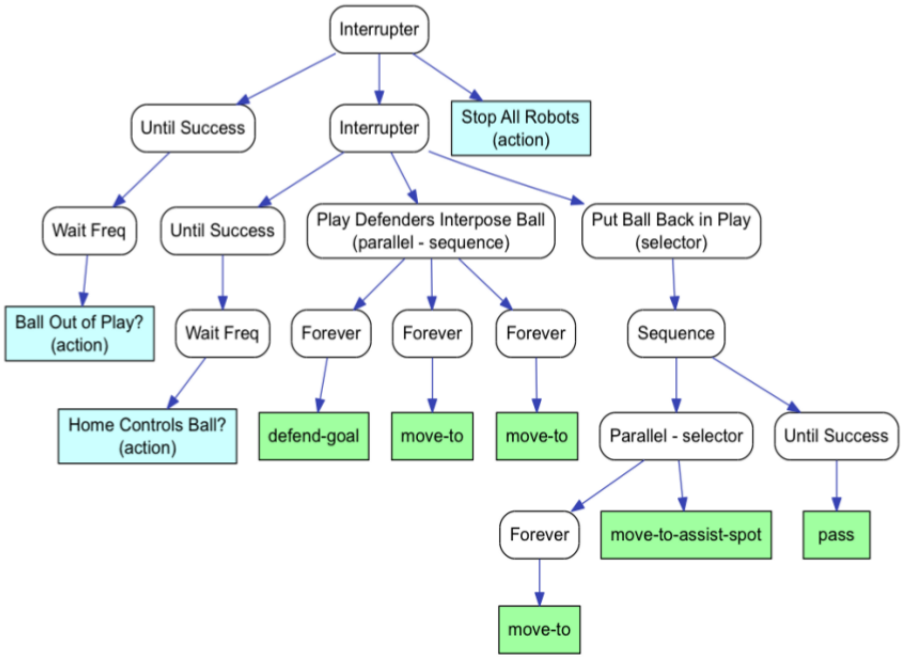


Figure 14: Defensive Play Behaviour

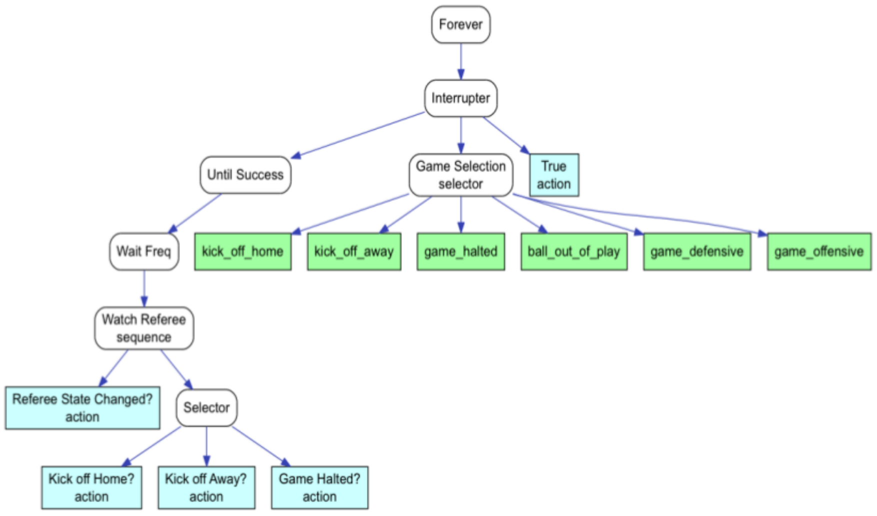


Figure 15: Play Selection Behaviour

Interrupter node runs in parallel, two subtrees Game Selection tree and Watch Referee tree, Game Selection is run as long as there is no referee event received. When a referee event occurs Watch Referee will succeed and interrupt Game Selection this behaviour lets the tree to start with a tactic that suited to the last referee event received.

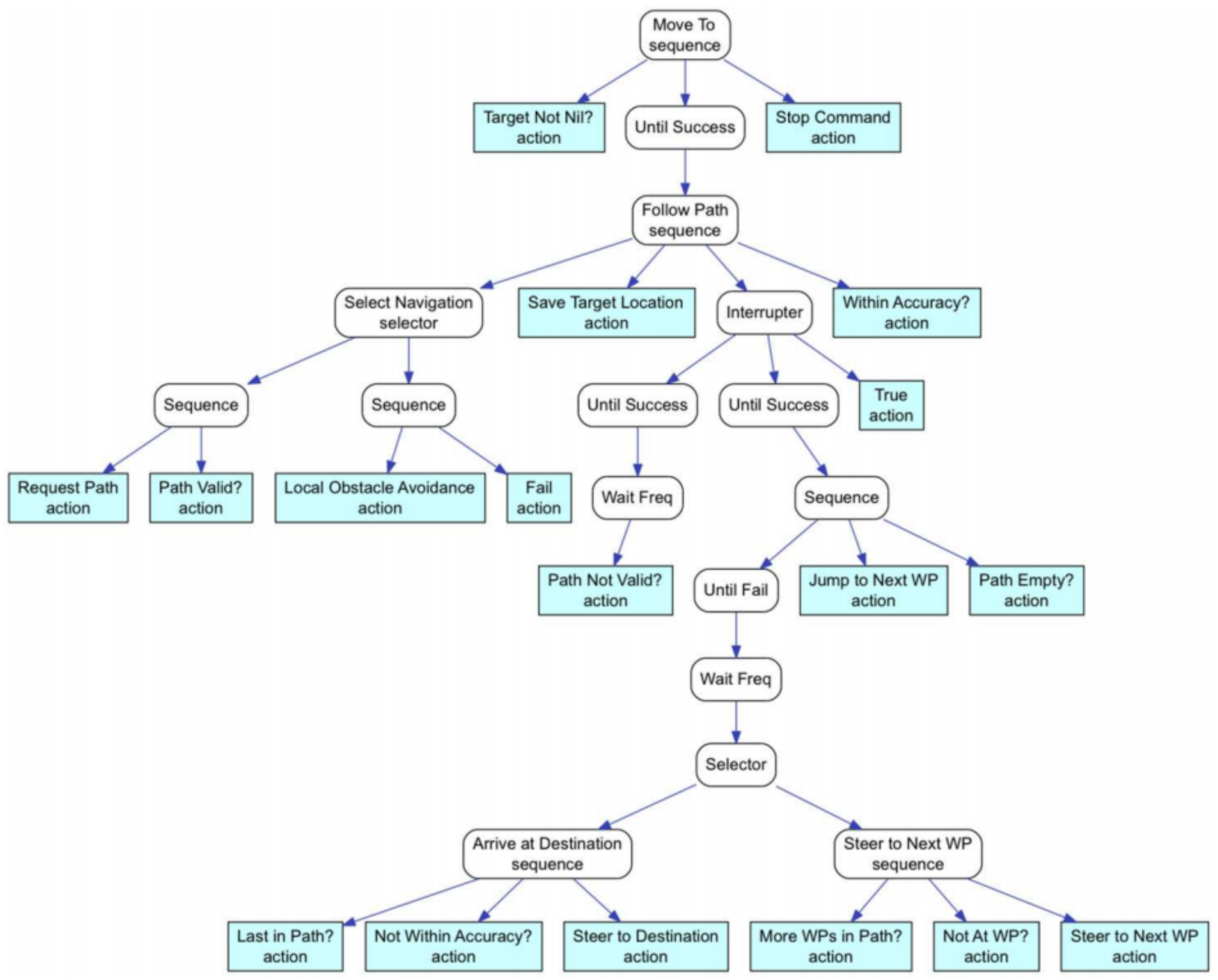


Figure 16: Move To via Path Behaviour

CHAPTER V

Robot Navigation

Path Planning

In mobile robot navigation one challenging task is to design a path finding algorithm that works effectively in dynamic fast changing environments. A commonly used algorithm for path finding in dynamic environments is Rapidly exploring random tree (RRT) algorithm [22, 23, 24, 41, 42, 43, 44] . RRT algorithm can be used to efficiently search high-dimensional, nonconvex search environments. RRT is used to search for a route from the start location to the target location by expanding a tree structure. RRT algorithm works as follows,

- The tree is initialized with start location as root node.
- A random location is picked within the valid search enviroment.
- A vertex in the tree which is closest to the random location chosen in the previous step is searched.
- Create there a new leaf by moving a some distance from this vertex towards the chosen point direction.
- Loop over step 2. to 4. Until a branch of the tree gets to the goal location

The main idea behind RRT-Plan is to guide the search to unexplored regions of the search space by picking regions in the state space and guiding the tree search towards these regions. The algorithm is implemented using a tree structure and works by growing the tree from the initial configuration until one of the branches runs into the target state. RRT-Plan algorithm tries to expand the tree by adding new vertexes that are biased using a randomly selected state.

Fig.17 depicts motion planning algorithm RRT-Plan. As depicted, the inputs for RRT-Plan algorithm are representation of the world, start and target locations, RRT tree structure and exploration factor for each step.

During the execution of the algorithm, we begin by choosing where to explore the search space using chooseTarget. It randomly selects a point in the search space biased towards the target location.

That causes tree to expand towards the goal by minimizing the objective function which in the case of motion planning is distance to goal location. Next we continue by finding the closest node in the tree to the point returned from chooseTarget by using a function called nearest. In order to pick where to explore next, algorithm checks if the distance between the node and target location is less than the distance between generated point and target location At iteration when a new node is calculated, the distance between this nearest node and target location is checked. If this distance is shorted than epsilon it is assumed goal location is reached and the tree is returned as result of RRT algorithm, otherwise tree is extended and explored. During extension phase of the tree the collision with obstacles are tested. If there is collision with an obstacle the tree is not extended towards that direction, otherwise the tree will be extended towards the chosen location and RRT algorithm will be iterated once more until goal location is reached.

The RRT-Plan is a very simple algorithm, it is very cheap to calculate however the path returned is not guaranteed to be optimal. RRT algorithm has some disadvantages such as the path found by the RRT is not guaranteed to be the shortest or each search will result in a different path. Fig.18 shows result of path finding operation. Due to its tree structure RRT-Plan algorithm finds multiple paths in the world in short time, then selects a path that gets to the target.

```
function RRT-Plan (world, start, goal, epsilon, p-goal, tree)
  target  chooseTarget(world, pGoal, goal)
  nearest  nearest(tree, target)
  if( distance(nearest,goal) < epsilon )
   return tree
  else
   explored  explore(world, nearest, target, epsilon)
   if (explored != nil )
    addNode(tree,explored)
   RRT-Plan(world, start, goal, epsilon, p-goal, tree)

function chooseTarget(world, pGoal, goal)
  p  UniformRandom in [0.0 .. 1.0]
  if 0 < p < pGoal
   return goal;
  else
   return randomState(world)

function nearest(tree,target)
  point  first(tree)
  for each node in rest(tree)
   if distance(node, target) < distance(point, target)
    point  node
  return point

function explore(world, u, v, epsilon)
  explored  extend(u, v, epsilon)
  if checkCollision(world, explored) = false then
   return explored
  else
   return nil
```

Figure 17: RRT Path Finding Algorithm

For navigating mobile robots in highly dynamic environments, determination of shortest route in a small amount of runtime is important. As depicted in Fig.18 the route found by the RRT-Plan algorithm is suboptimal. It may contain many unnecessary or redundant waypoints which makes it longer. In order to deal with this problem and further optimize the path a simple and efficient smoothing algorithm [45, 46] is applied to the calculated path which reduces the number of waypoints on the path.

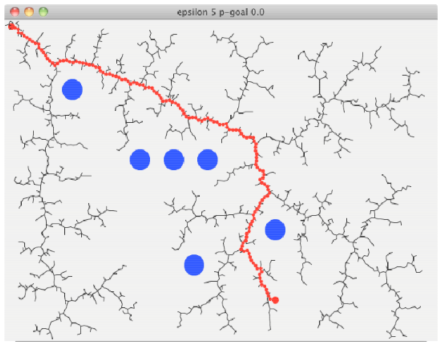
smooth-path is a recursive algorithm that removes waypoints that are reachable from a given waypoint. Fig.19 demonstrates the motion plan after smoothing. Given two waypoints that are reachable from each other A and B. Fig.19 Path smoothing algorithm deletes any waypoints in between A and B since B can be reached from A directly without going through any intermediary waypoints.

Proposed algorithm for smoothing the path shown in Fig.20. Path smoothing algorithm begins with first waypoint in the path, then drop-while-walkable function is called to find a waypoint that is farthest from the current waypoint that is reachable without colliding with any obstacles. This farthest waypoint is added to a new smoothed path. Same operations are repeated again using farthest waypoint. This is repeated until there are no more waypoints left on the path. As shown Fig.19 a dashed line represents the RRT path solid line represents the smoothed path. As a result of smoothing the path is optimized.

Combination of RRT and path optimization algorithms allows fast path finding in highly dynamic environments.

Collision Avoidance

Even though RRT motion planning is fast to calculate it still can't be run at high enough frequencies for tight maneuvers, where primary aim is not to go from A to B but to maneuver around obstacles without hitting. Navigation system for NeuIslanders works in two stages stage one uses RRT path finding



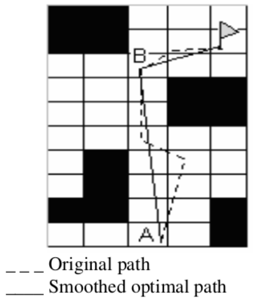


Figure 19: Optimized path after path

Figure 18: RRT path when P(Goal)=0optimization.

```
function smooth-path (isWalkable, path, curr-node, path-rest)
  if(isEmpty(rest) == false)
   path-rest = drop-while-walkable(fn(isWalkable,curr-
node,%),path-rest)
   x = first(path-rest)
   xs = rest(path-rest)
   smooth-path(isWalkable, addNode(path-rest,curr-
node), x, xs))
  else
   return addNode(path-rest,curr-node)

function drop-while-walkable (pred, path, curr-node)
  if(isEmpty(path) == false && pred(first(path)) == true)
   drop-while-walkable(pred, rest(path), first(path))
  else
   return addNode(curr-node, s)
```

Figure 20: Path Optimization Algorithm

algorithm that has been described. Stage two takes over when we get close to our target location, system switches from path finding mode to collision avoidance mode. the fuzzy logic system designed here focuses on reaching the target, while avoid hitting the obstacles. An example scenario that shows avoidance of obstacles is depicted in Fig.21. In this example, the soccer playing robot uses the distance to check if there is an obstacle or not. After the obstacle is detected, boundaries for the obstacle is calculated (left and right). During collision avoidance these obstacle boundries are enlarged to make ensure there is enough safety margin for the soccer robot. This newly calculated boundary is called the =safety boundary=. This enables safe navigation of the robot around obstacles. Thus each obstacle on the field has two boundaries, a tight boundary and a larger safety boundary, as depicted in Fig.21.

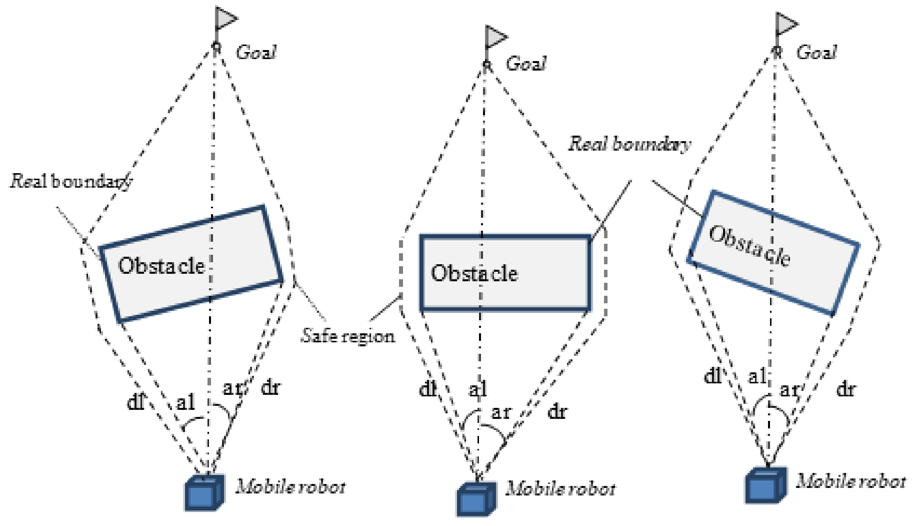
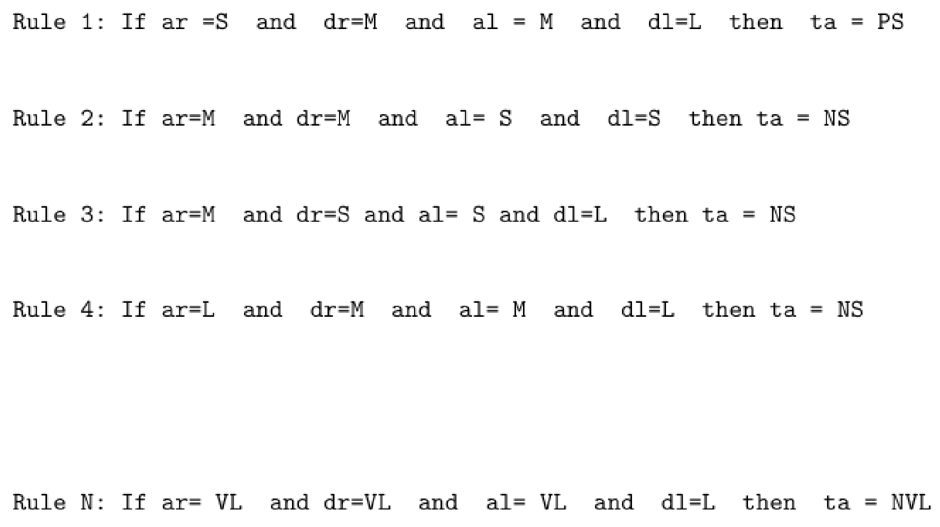


Figure 21: collision avoidance

A knowledge base is used to implement the collision avoidance algorithm. This knowledge base is made up of If-Then rules which describes the logic of how the mobile robot should avoid the obstacles. There are multiple strategies to use depending on the state of the robot and dimensions of the obstacle. The goal of the designed avoidance system is to avoid the obstacles while arriving at the goal location using shortest path. Three different strategies are shown in Fig.21 that demonstrates the required turn angle of the mobile robot for avoidance. Each strategy can be described using a If-Then rule base which is built by combining different strategies. Input variables for the rule base are the left and right angles and distance to obstacle, output variable is the turn rate for the robot. The side angles, left (al) and right (ar) are calculated using the line between robot, goal and the line between robot, real boundaries of the soccer robot. The distances, left and right are calculated using the distance between robot and left/right boundary locations. Fig.22 depicts the portion of the fuzzy If-Then rule base used for collision avoidance of robots. linguistic terms in the rule base are used to represent the values of input and output parameters.



```
Rule 1: If ar =S  and  dr=M  and  al = M  and  dl=L  then  ta = PS

Rule 2: If ar=M  and dr=M  and  al= S  and  dl=S  then ta = NS

Rule 3: If ar=M  and dr=S and al= S and dl=L  then ta = NS

Rule 4: If ar=L  and  dr=M  and  al= M  and  dl=L  then ta = NS



Rule N: If ar= VL  and dr=VL  and  al= VL  and  dl=L  then  ta = NVL
```

Figure 22: Fragment of Rule Base

In the rule base, VL, L, M, S, VS, Z are linguistic terms and represent very large, large, medium, very small, and zero, PVL, PL, PM, PS, Z, NS, NM, NL, NVL are positive very large, positive large, positive medium, positive small, zero, negative small, negative medium, negative large, negative very large. These linguistic terms describes the values of the input and output variables of the system. Target angle (ta) of the mobile robot is computed using the fuzzy inference system. A safety margin is combined with the target angle in order to find a safer target angle for the mobile robot. The primary strategy for collision avoidance is to choose a target angle that has the smallest distance from the mobile robot to the target location.

ta(k)=F(l,r,dl,dr)

- l - left angle
- r - right angle
- dl - distance left
- dr - distance right

In the fuzzy rule base the values of input and output vaiables are implemented with type-2 fuzzy sets. The type-2 fuzzy inference is used for calculating target turn angle.

Experimental Results

Experimental studies are done for RRT-Plan and path optimization algorithms in order to demonstrate the both algorithm's feasibility in controlling mobile soccer robots. For the first experiment the implementation of RRT-Plan for mobile robot motion planning has been used. For this experiment a map of a world containing obstacles is simulated. The obstacles are represented using colored circles. (Fig.23). Mobile robot uses the map of world and plans a route in real time in order to move to the target position.

The RRT-Plan algorithm has been experimented with different values of (is the distance that the tree is extended at each iteration of the algorithm.)

and P(Goal) (probability to expand the tree towards the goal). Fig.23 shows the simulation results for RRT-Plan algorithm using $= 50$, P(Goal) $= 0.3$ and $= 15$, P(Goal) $= 0.3$. As shown from the figure we can conclude that the increase of P(Goal) will result in the reduction of search space of RRT-Plan and pushes the tree faster towards the target. on the other hand effects probability of the exploration done in each iteration (Fig.23. Table 1 shows the average time it takes to calculate RRT-Plan to reach target location. Fig.23.)

Fig.24 shows the effects of P(Goal) and values on average length and time for Fig.23. As shown in Fig.24 changing P(Goal) effects the runtime of RRT-Plan algorithm. The experiment is also done for different constant values of . For small values of P(Goal) RRT-Plan takes longer to run because of large sections of map needs to be explored by the algorithm. For large values of P(Goal) the algorithm moves the search towards the goal which results in shorter runtime. But this gives too much bias towards the target which results in increased running time because of mobile robot getting stuck around the obstacles. As shown in Fig.24 experiments shoed that the optimal values of P(Goal) and is 0.4, 15. Next two dashed and dotted lines in Fig.24 are obtained when $= 30$ and $= 50$. The conclusion reached above can be also said for these cases. Fig.24 shows time versus when using different values of P(Goal). As can be said from Fig.24 the increasing decreases the runtime of RRT-Plane because of faster exploration.

The same motion planning experiment was also done using two other before mentioned algorithms namely A* and artificial potential field (APF) for the same map shown in Fig.23. A* algorithm uses grids for calculating the path, search times for different grid sizes is given in Table 2.It also should be noted that the path length of A* using a good metric is close to the path length of APF. Runtime of APF is more than A* algorithm.

As shown in the tables 1 and 2, runtime of RRT-Plan algorithm is considerably smaller than the runtime of A* or APF. This shows that using

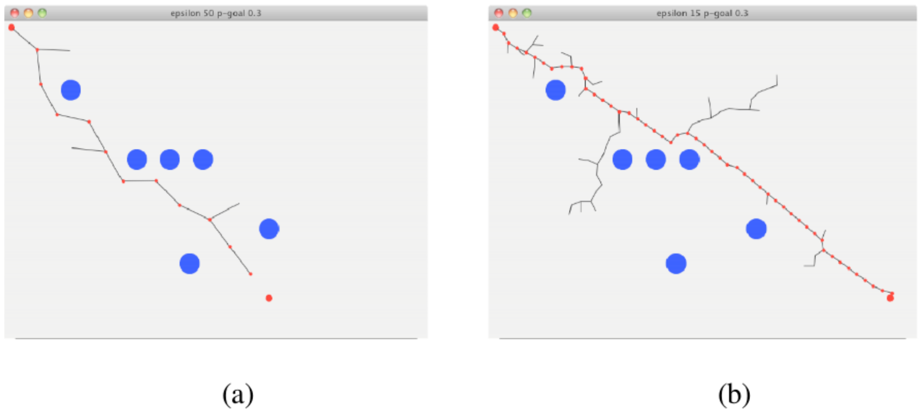


Figure 23: RRT execution results for various values (/ P(Goal)), a) 50/0.3, b)15/0.3

Table 1: Experiment results for RRT-Plan

	P(Goal)	Runtime	Distance
15	0.1	28.41	818.04
15	0.2	21.78	798.63
15	0.3	20.02	786.71
15	0.4	18.95	777.91
15	0.5	19.67	774.13
30	0.1	9.09	823.58
30	0.2	7.68	803.41
30	0.3	7.20	789.43
30	0.4	6.90	779.35
30	0.5	7.29	770.83
50	0.1	4.46	828.83
50	0.2	3.74	807.84
50	0.3	3.43	796.03
50	0.4	3.42	785.86
50	0.5	3.50	779.29

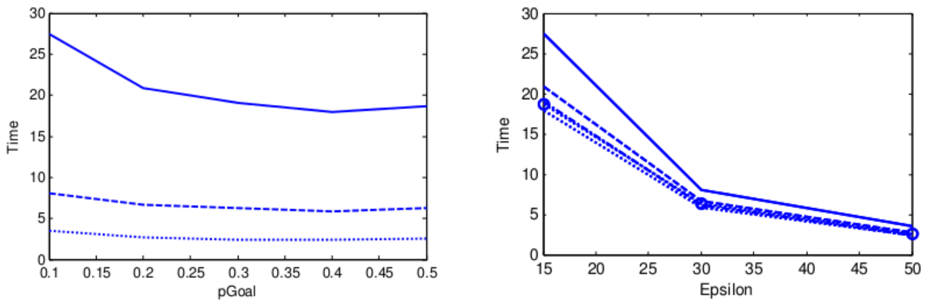


Figure 24: Effects of and P(Goal) with respect to time. a) P(Goal) vs Time, solid line =15, dotted line =50, dashed line =30 b) with respect to Time, dashed line P(Goal)=0.2, solid line P(Goal)=0.1, dotted line P(Goal)=0.4, dotted dash line P(Goal)=0.3, dotted 'o' line with P(Goal)=0.5.

Table 2: A* and APF results

Methods	Grid Size	Time	Length
A*	50	15.30	824.26
	30	15.81	699.42
	15	45.56	676.69
APF		53.93	663.96

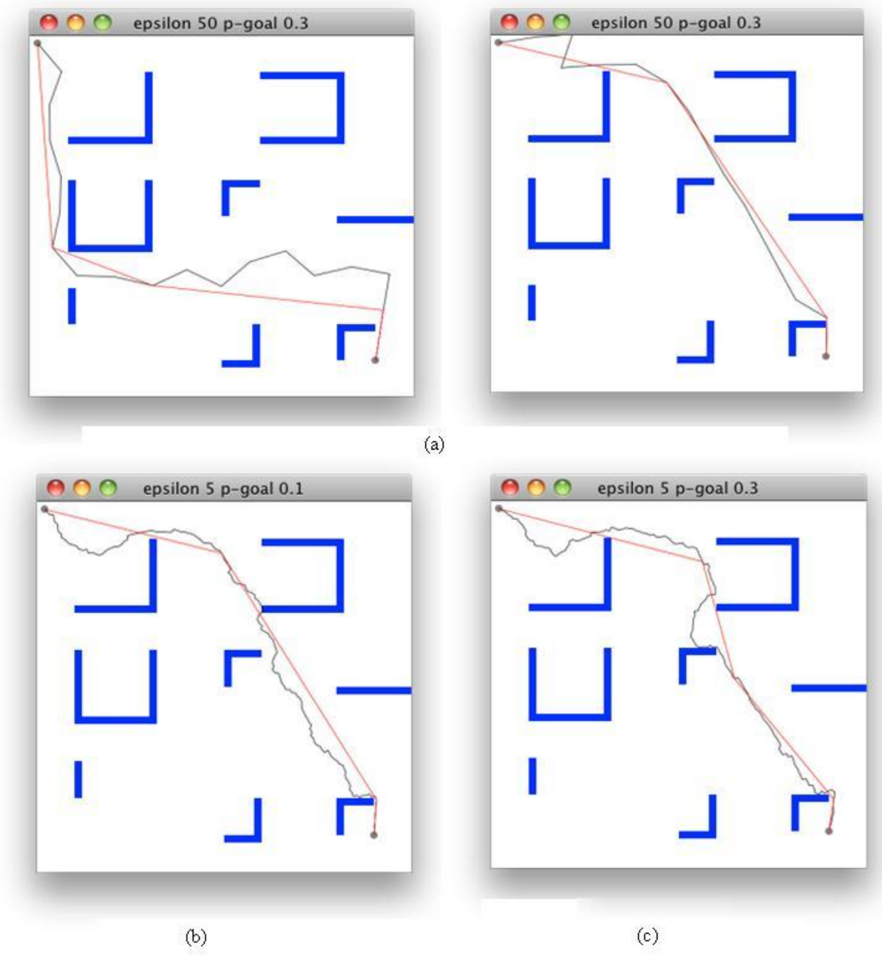


Figure 25: RRT-Plan and Path optimization algorithm results for varoius /P(Goal) pairs (Black line RRT-Plan path, red line smoothed rrt-plan path), 50,0.3 (a), 5,0.1 (b) and 5,0.3 (c).

RRT-Plan in fast changing environments requires much shorter time to plan a trajectory for the robot compared A* and APF. But one deficiency of this approach is that unlike A* algorithm which guarantees shortest path the resulting path calculated using RRT-Plan is not necessarily the shortest, which makes the planed motion to be longer than motion calculated by A* and APF. This mentioned deficiency can also concluded from comparing tables 1 and 2. To fix this deficiency, the path calculated using RRT-Plan is optimized by pruning the route using the proposed path optimization algorithm.

An example experiment of the before mentioned RRT-Plan and path optimization algorithm for motion planning problem is shown below. Fig.25 shows the result of experiment for RRT-Plan and path optimization algorithms, used for mobile robot motion planning problem for following values of, $= 50$, P(Goal) $= 0.3$ (a), $= 5$, P(Goal) $= 0.1$ (b) and $= 5$, P(Goal) $= 0.3$ (c). Dashed lines are RRT-Plan results and solid lines are path optimization results for both figures. In this experiment start location was (40,40) and the target location was (400,400). In Fig.25 (a) two different experiments are given for RRT-plan and path-smoothing algorithms, when $= 50$, P(Goal) $=$ 0.3.

Table 3 shows the experiment for RRT-Plan and path optimization algorithm depicted in Fig.25 for various values of P(Goal) and .

Experiment results shows that the runtime and path length with (columns 3 / 4) and without (columns 5 / 6) path optimization applied. When compared, the results in columns 4 / 6, it is clear that using path optimization algorithm results in shorter path length [47]. When we compare columns 3 / 7 it is shown that the runtime of motion planning algorithm is got bigger after running path optimization algorithm. Fig.25 is also simulated using A* algorithm using same experiment conditions. As demonstrated runtime is increased greatly, but average path length is roughly same as in RRT-Plan.

As depicted in before mentioned tables application of path smoothing algorithm to the result of RRT-Plan algorithm allows optimization of the path calculated by RRT-Plan and results in shorter path length.

The results of experiment for A* and APF algorithms in Fig.25 using various different grid sizes are shown in table 4. When compared these results combined with the results of table 3 it can be concluded that the runtime of RRT combined with path optimization algorithm is considerably smaller than the runtime of A* algorithm. Because of these optimizations, the path length calculated using the combination of RRT-Plan combined with path optimization algorithm is close to optimal path calculated by A* algorithm. From it can be concluded that the use of RRT-Plan combined with path optimization algorithms allows efficient control of mobile robots in dynamic fast changing environments cluttered with dynamic moving obstacles.

Table 3: Path finding and Path optimization experiment results

Epsilon	P(Goal)	Runtime	Distance	Runtime Smooth	Smoothed Distance
15	0.1	561.65	925.47	548.68	733.46
15	0.2	544.61	921.41	567.76	737.38
15	0.3	622.17	893.16	594.67	740.40
15	0.4	729.54	896.71	708.25	730.44
15	0.5	713.72	895.69	890.50	729.78
30	0.1	225.79	916.93	228.99	750.96
30	0.2	240.77	907.69	259.99	746.44
30	0.3	264.90	916.60	289.25	758.92
30	0.4	297.12	900.24	311.54	748.40
30	0.5	353.72	907.56	372.44	760.26
50	0.1	130.46	906.57	140.69	771.21
50	0.2	137.22	924.29	150.81	764.18
50	0.3	162.26	899.44	172.08	756.15
50	0.4	166.26	918.19	203.53	760.81
50	0.5	192.88	905.12	215.66	765.42

Table 4: Simulation results for A* and APF algorithm

Method	Grid Size	Runtime	Distance
A*	1	28833.62	883.89
	2.5	4834.85	870.28
	5	1332.35	819.32
APF		676.49	679.76

CHAPTER VI

Conclusions

In this thesis decision making, motion planning and collision avoidance algorithms have been designed for soccer playing robots robots. The decision making system is based on the concept using behavior trees that allows organization of collections of states and the decision process of moving between them allowing very complex behaviour to be modelled easily Modular architecture of BTs allows extending the DM for complicated states. Path finding algorithm based on RRT-Plan algorithm has been implemented. That drastically shortens the path calculated by RRT-Plan. Implemented collision avoidance algorithm is based on the type-2 fuzzy system. The algorithm uses the estimation of left and right angles and distances and their relations in fuzzy rule base. The results obtained shows that the applicability of the designed decision making, path finding and collision avoidance algorithms. The results from the real-life implementation demonstrate the effectiveness of the proposed algorithms for control of soccer playing robots in dynamically changing environments.

References

[1] Y. Liu, J. J. Zhu, R. L. Williams II, and J. Wu, "Omni-directional mobile robot controller based on trajectory linearization," Robotics and autonomous systems, vol. 56, no. 5, pp. 461–479, 2008.

[2] K. Watanabe, Y. Shiraishi, S. G. Tzafestas, J. Tang, and T. Fukuda, "Feedback control of an omnidirectional autonomous platform for mobile service robots," Journal of Intelligent and Robotic Systems, vol. 22, no. 3-4, pp. 315–330, 1998.

[3] R. H. Abiyev, N. Akkaya, E. Aytac, and D. Ibrahim, "Behaviour tree based control for efficient navigation of holonomic robots," International Journal of Robotics and Automation, vol. 29, no. 1, pp. 44–57, 2014.

[4] J. Borenstein, H. R. Everett, and L. Feng, Navigating Mobile Robots: Systems and Techniques. Natick, MA, USA: A. K. Peters, Ltd., 1996.

[5] B. Damas, P. Lima, and E. S. D. Tecnologia, "Stochastic discrete event model of a multi-robot team playing an adversarial game," in Proc. of 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles - IAV2004, 2004.

[6] A. C. Domínguez-Brito, M. Andersson, and H. I. Christensen, "A software architecture for programming robotic systems based on the discrete event system paradigm," tech. rep., Centre for Autonomous Systems, KTH, 2000.

[7] E. P. Dadios and S. H. Park, "Real time robot soccer game event detection using finite state machines with multiple fuzzy logic probability evaluators," Int. Journal of Computer Games Technology, vol. 2009, pp. 5:1–5:12, Jan. 2009.

[8] H. Costelha and P. Lima, "Modelling, analysis and execution of robotic tasks using petri nets," in Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ Int. Conference on, pp. 1449 –1454, 29 November 2007.

[9] P. L. D. Milutinovic, "Petri net models of robotic tasks," in Procedings of the IEEE Int. Conference on Robotics and Automation, 2002.

[10] M. J. Mataric, "Behavior-based control: Examples from navigation, learning, and group behavior," Journal of Experimental and Theoretical Artificial Intelligence, vol. 9, pp. 323–336, 1997.

[11] M. Mataric, "Behavior-based control: Main properties and implications," in In Proceedings, IEEE Int. Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems, pp. 46–54, 1992.

[12] C.-U. Lim, R. Baumgarten, and S. Colton, "Evolving behaviour trees for the commercial game defcon," in Proceedings of the Int. conference on Applications of Evolutionary Computation - Volume Part I, EvoApplicatons'10, (Berlin, Heidelberg), pp. 100–110, Springer-Verlag, 2010.

[13] G. Neto, H. Costelha, and P. Lima, "Topological navigation in configuration space applied to soccer robots," in Robocup 2003, LNAI, Springer-Verlag Berlin Heidelberg, 2004.

[14] J. Borenstein, Y. Koren, and S. Member, "The vector field histogram - fast obstacle avoidance for mobile robots," IEEE Journal of Robotics and Automation, vol. 7, pp. 278–288, 1991.

[15] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots in cluttered environments," in Proceedings of IEEE Int. Conference on Robotics and Automation, pp. 572 –577 vol.1, may 1990.

[16] K. Fujimura, Motion Planning in Dynamic Environments. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1992.

[17] A. F. M. Yousef Ibrahim, “Study on mobile robot navigation techniques,” in IEEE Int. Conference on Industrial Technology, vol. 1, pp. 230–236, 2004.

[18] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” in IEEE Robotics Automation Magazine, vol. 4, 1997.

[19] I. Ulrich and J. Borenstein, “Vfh+: Reliable obstacle avoidance for fast mobile robots,” in Proceedings of the IEEE Int. Conference on Robotics and Automation, 1998.

[20] M. Y. Ibrahim, “Mobile robot navigation in a cluttered environment using free space attraction agoraphilic algorithm.,” in Proceedings of the 9th Int. Conference on Computers and Industrial Engineering, vol. 1, pp. 377–382, 2002.

[21] A. Stentz and M. Hebert, “A complete navigation system for goal acquisition in unknown environments,” Autonomous Robots, vol. 2, pp. 127–145, 1995.

[22] J. Kuffner, J.J. and S. LaValle, “Rrt-connect: An efficient approach to single-query path planning,” in Proceedings of the IEEE Int. Conference on Robotics and Automation, vol. 2, pp. 995 –1001 vol.2, 2000.

[23] S. M. LaValle, J. J. Kuffner, and Jr., “Randomized kinodynamic planning,” in Int. Journal of Robotics Research, vol. 20(5), pp. 378–400, 2001.

[24] S. M. LaValle, Planning Algorithms. Cambridge, U.K.: Cambridge University Press, 2006. Available at http://planning.cs.uiuc.edu/.

[25] M. v. d. P. Maciej Kalisiak, “Rrt-blossom: Rrt with a local flood-fill behavior,” in IEEE Int. Conference on Robotics Automation, 2006.

[26] P. Hart, N. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” IEEE Transactions on Systems Science and Cybernetics, vol. 4, pp. 100 –107, july 1968.

[27] H. F. D. S. K. Y. Nobuyuki Kurihara, Ryotaku Hayashi, “Intelligent control of autonomous mobile soccer robot adapting to dynamic environment,” in RoboCup 2003, LNAI, Springer-Verlag Berlin Heidelberg, 2004.

[28] D. I. R.Abiyev and B. Erin, “Edurobot: An educational computer simulation programm for navigation of mobile robots in the presence of obstacles,” in Int. Journal of Engineering Education, vol. 26, pp. 18–29.

[29] R. Abiyev, D. Ibrahim, and B. Erin, “Navigation of mobile robots in the presence of obstacles,” Advanced Engineering Software, vol. 41, pp. 1179–1186, Oct. 2010.

[30] E. Dijkstra, “A note on two problems in connexion with graphs,” Numerische Mathematik, vol. 1, pp. 269–271, 1959.

[31] S. Tan, S. X. Yang, and A. Zhu, “A novel ga-based fuzzy controller for mobile robots in dynamic environments with moving obstacles.,” Int. Journal of Robotics and Automation, vol. 26, no. 2, 2011.

[32] K. A. S.H. Sadati and M. Behroozi., “A combination of neural network and ritz method for robust motion planning of mobile robots along calculated modular paths,” Int. Journal of Robotics Automation, vol. 23, no. 3, 2008.

[33] K. H. Sedighi, T. W. Manikas, K. Ashenayi, and R. L. Wainwright, “A genetic algorithm for autonomous navigation using variable-monotone paths,” Int. Journal of Robotics and Automation, vol. 24, no. 4, 2009.

[34] C. Son, “Intelligent robotic path finding methodologies with fuzzy/crisp entropies and learning.,” Int. Journal of Robotics and Automation, vol. 26, no. 3, 2011.

[35] B. S. V. M.Ganapathy, S. Parasuraman, “Behavior based mobile robot navigation by ai techniques: behavior selection and resolving behavior conflicts using alpha level fuzzy inference system,” Int. Journal of Automation, Robotics and Autonomous Systems, vol. 5, no. 1, 2006.

[36] J. A. Fernandez-Leon, G. G. Acosta, and M. A. Mayosky, “Behavioral control through evolutionary neurocontrollers for autonomous mobile robot navigation,” Robotics and Autonomous Systems, vol. 57, pp. 411–419, Apr. 2009.

[37] J. Bruce and M. Veloso, “Real-time randomized path planning for robot navigation,” in Int. Conference on Intelligent Robots and Systems, vol. 3, pp. 2383–2388, 2002.

[38] S. Karaman and E. Frazzoli, “Incremental sampling-based algorithms for optimal motion planning,” in Proceedings of Robotics: Science and Systems, (Zaragoza, Spain), June 2010.

[39] R. Alterovitz, S. Patil, and A. Derbakova, “Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning.,” IEEE Int. Conf on Robotics and Automation, pp. 3706–3712, 2011.

[40] S. Zickler, T. Laue, O. Birbach, M. Wongphati, and M. Veloso, “Ssl-vision: The shared vision system for the robocup small size league,” in Robot Soccer World Cup, pp. 425–436, Springer, 2009.

[41] B. Browning, J. Bruce, M. Bowling, and M. Veloso, “STP: Skills, tactics and plays for multi-robot control in adversarial environments,” Journal of Systems and Control Engineering, vol. 219, no. 1, pp. 33–52, 2005.

[42] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, “Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian Processes,” in IEEE/RSJ 2008, Int. Conference on Intelligent Robots and Systems, (Nice, France), pp. 1056–1062, 2008.

[43] S. Sengupta, “A parallel randomized path planner for robot navigation,” Int. Journal of Advanced Robotic Systems, vol. 3, no. 3, pp. 259–266, 2006.

[44] D. Ferguson and A. Stentz, “Anytime rrts,” in Proceedings of the IEEE Int. Conference on Intelligent Robots and Systems IROS, 2006.

[45] M. Buckland, Programming game AI by example. Wordware Publishing, 2005.

[46] M. Waringo and D. Henrich, “Efficient smoothing of piecewise linear paths with minimal deviation,” in IEEE/RSJ Int. Conference on Intelligent Robots and Systems, pp. 3867–3872, 2006.

[47] R. Abiyev, N. Akkaya, and E. Aytac, “Navigation of mobile robot in dynamic environments,” in IEEE Int. Conference on Computer Science and Automation Engineering (CSAE), vol. 3, pp. 480 –484, may 2012.

Appendices

NEAR EAST UNIVERSITY

ETHICAL APPROVAL DOCUMENT

Date:29/12/2021

To the Institute of Graduate Studies

For the thesis project entitled as "Behaviour Tree Based Control of Omnidirectional Robots", the researchers declare that they did not collect any data from human/animal or any other subjects. Therefore, this project does not need to go through the ethics committee evaluation.

Title: Prof. Dr.

Name Surname: Rahib Abiyev
Signature: [illegible]

Role in the Research Project: Supervisor

Skip to Main Content
Assignments
Students
Grade Book
Libraries
Calendar
Discussion
Preferences

About this page

This is your assignment inbox. To view a paper, select the paper's title. To view a Similarity Report, select the paper's Similarity Report icon in the similarity column. A ghosted icon indicates that the Similarity Report has not yet been generated.

Chapters

Inbox | Now Viewing: new papers ▼

Submit File Online Grading Report | Edit assignment settings | Email non-submitters

Delete Download move to...

☐	Author	Title	Similarity	web	publication	student papers	Grade	response	File	Paper ID	Date
☐	Nurullah Akkaya	Chapter 1	0% 0%	0%	0%	0%	--	--	download paper	1743414324	18-Jan-2022
☐	Nurullah Akkaya	Chapter 6	0% 0%	0%	0%	N/A	--	--	download paper	1743419187	18-Jan-2022
☐	Nurullah Akkaya	Chapter 2	1% 1%	1%	1%	N/A	--	--	download paper	1743425233	18-Jan-2022
☐	Nurullah Akkaya	Chapter 4	7% 7%	4%	7%	N/A	--	--	download paper	1743415516	18-Jan-2022
☐	Nurullah Akkaya	Chapter 3	8% 8%	5%	3%	N/A	--	--	download paper	1743415339	18-Jan-2022
☐	Nurullah Akkaya	Chapter 5	19% 19%	2%	19%	N/A	--	--	download paper	1743415722	18-Jan-2022

Supervisor Prof.Dr. Rashid Aliyev

[illegible]