

# Humanoid Soccer Robot Motion Planning using GraphPlan

Nima Shafii, Lucio Sanchez Passos, Luis Paulo Reis,  
*Artificial Intelligence and Computer Science Laboratory  
 Faculty of Engineering of the University of Porto - FEUP  
 Porto, Portugal*  
 {nima.shafii, pro09026, lpreis}@fe.up.pt

Nuno Lau  
*Instituto de Engenharia Electronica e Telematica de  
 Aveiro (IEETA), Universidade de Aveiro,  
 Aveiro, Portugal*  
 Nunolau@ua.pt

**Abstract—** Humanoid robotics is being studied by various fields, because it can be an interesting test bench for different technologies. Furthermore, methodologies for planning actions in advance are widely used for mobile robot's movements in static environments and are being extended and adapted to dynamic environments, such as soccer games between humanoid robots. The choice of the methodology to achieve the plan is vital to accomplish the final goals. This paper reports the modelling and implementation of a path planning methodology for humanoid robots. The main contribution relies on applying planning techniques and iterative deepening search to determine the best sequence of actions that lead the robot to traverse the optimal path to the ball so that it can kick the ball to a given pre-defined target. Some results are presented and discussed to prove the efficiency and reliability of the approach.

**Keywords -** Path Planning; Humanoid Robots; Robotic Soccer.

## I. INTRODUCTION

Robotics is a branch of engineering that involves various fields as mechanics, electronics and computer science. Thus, robots with a human form always aroused curiosity and, in the last few years, the idea to have a perfect humanoid robot is becoming more real. In the humanoid robot field, RoboCup which includes a soccer competition using NAO humanoid), is giving huge contributions.

One interesting way to achieve better results is to apply Artificial Intelligence (AI), in scenarios as learning how to walk, planning objects manipulation, path planning, and so forth. The context of this work is path planning for humanoid robots in a soccer based environment.

In such continue and real-time environment, Footstep Planning [1] may used in the path planning task. It discretizes the possible actions of the robot into a small set of well-chosen actions as different foot placements is not a proper solution due to the fact that its search graph grows exponentially with the number of steps. Only a small number of footsteps can be explored in a real-time system [2].

We have to compute a trajectory for the body-center of a humanoid robot by approximating the shape of the robot. The shape of this trajectory depends on the humanoid robots' ability of low level biped locomotion skills, such as walking curving and etc.

In previous works in this research topic, humanoids often have basic low skills for performing their tasks and following their path trajectory, including forward walk, turn, and different

curve skills [3]. Fortunately, in our recent project on biped locomotion we can achieve to omni directional walking, it means that the robot is able to walk in forward and backward direction, different type of walking while curving and turning on the spot [4], [5]. Therefore, it has the capability of following many path trajectories. This flexibility is an advantage, however it increases the complexity of the path planner dramatically.

In motion planning the number of degrees of freedom is usually small which opens the door for the application of search techniques. Humanoid robots usually have more than ten degrees of freedom. Their feet can be placed with a great precision and changing the body posture allows to overcome obstacles that wheeled robots fail in passing through.

In this paper, we present a scenario where the robot, starting from a position far from a ball, has to approach it and then kick it to the target, using the best possible path. To execute this, the robot has to construct the space state and search for the solution, spending the shortest possible time. Also, environment and robot constraints were reviewed for system implementation. We simulated it with RoboCup official 3D simulator to evaluate the planner performance.

The remaining of this paper is organized as follows. Section II presents the characteristics of an humanoid robot and the humanoid simulator used In the next section, we discuss the motion planning related work to understand what already is developed relating it with humanoid robotics. The fourth section presents our methodological approach, experimental results, and finally section V presents the conclusions and suggestions for future work.

## II. HUMANOID ROBOTICS

A humanoid robot is a robot with its overall appearance, based on human body, and with the ability to move on its own legs. The number of joint actuators indicates the number of Degree of Freedom (DOF). Like humans, humanoid's body moves in three planes, including transverse (axial), frontal (coronal) and sagittal Planes. Sagittal plane indicates the vertical plane running from front to back and dividing the body into left and right sides. Frontal or Coronal plane is a plane, perpendicular to the sagittal plane, running from side to side and dividing the body into front and back.

The creation of humanoid robots has been motivated by the idea of having a device capable of operating in environments made by and for humans with minimal change to those environments. These machines are expected to perform autonomously most of the functions a person is capable of. These include climbing stairs, reaching for objects, etc.

One of the best known applications of humanoid robotics is the humanoid league of RoboCup. The goal of the RoboCup initiative is to develop a team of humanoid robots able to win against the official human World Soccer Champion team until 2050. The humanoid league is concerned with skills as: dynamic walking, running, kicking the ball while maintaining balance, teamwork, and self-localization. The smaller competition category is KidSize (30-60cm). There is also a humanoid standard platform league which uses the NAO humanoid robot [6].

The French company Aldebaran Robotics developed NAO. The first prototype was released in 2005. Fig. 1 shows NAO, a 58cm height robot with 22 degree of freedom (DOF), counting on different sensors distributed in key parts of it [7]. The Nao model is used in the RoboCup 3D simulation league, using Linux and based on spark technology. In this work we used the RoboCup 3D simulator to implement and test our prototype. Further information about the simulator will be given on subsection 5.2.


	Height	58 cm	
	Mass (including batteries)	4,3 kg	
	Degrees of Freedom	22	
	CPU	x86 AMD GEODE 500 MHz	
	Sensors	Head	2 cameras and microphones
		Chest	Ultrasound, 3-axis accelerometer and a 2-axis gyro-meter
		Legs	4 Force Sensitive Resistors
	Autonomy	90 min. (constant walking)	
Programming Languages	C++, C, Python, Urbi, .Net		

Figure 1. General Characteristics of Nao

### III. RELATED CONCEPTS AND TECHNOLOGIES

Robots have four basic components: sensing, actuating, planning, and control. Since we are focusing in the planning component, it is logic to ask: what is planning? To plan is abstract the process to choose and, also, organize a sequence of actions to achieve some goal. Nevertheless, to construct a plan for the real world is a hard task, so the classical planning often uses some constraints, such as: atomic time, closed world, deterministic effects of actions, and the planner has omniscience of knowledge [8].

There are several strategies of planning; however, we are going to focus on the one used in this work. First, explaining the data structure used to represent the problem (and consequently the search space), then the search technique and the approach used to model the problem.

There are various ways to search a tree structure and here we are going to focus on uninformed search (also called blind

search), specifically on Iterative deepening depth-first search (or Iterative deepening search). The uninformed search means that the algorithm has no additional information other than the capacities to generate successors and to distinguish a goal state from a non-goal state.

Iterative deepening search is a general strategy, often used in combination with depth-first search, which finds the best depth limit. Its algorithm (shown in Figure 2) also combines the benefits of depth-first search (modest memory requirements, precisely  $O(b^d)$ ), and breadth-first search (branching factor is finite and optimal when the path cost is a non-decreasing function of the depth of the node) [8]. However, it may seem wasteful, because states are generated multiple times, being this not very costly. Also, based on the total number of nodes the time complexity is  $O(b^d)$ .

```

function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution, or failure
inputs: problem, a problem

for depth ← 0 to ∞ do
    result ← DEPTH-LIMITED-SEARCH(problem, depth)
    if result ≠ cutoff then return result
    
```

Figure 2. Pseudo-code of Iterative deepening search [8]

Albeit we did not use GRAPHPLAN directly in this work, it has a great importance for this project in the task of problem modeling. GRAPHPLAN is a planning technique, which represents its plans on graph form. Two nodes compose this graph: propositional nodes (indicate the world state in a time instant. i.e. preconditions), and action nodes (represents the possible actions that can be performed with the indicated preconditions); also, the actions create a new propositional nodes with their effects. So, this idea of preconditions and possible action was used to model our problem (presented in subsection 5.1). Here we just gave a brief overview of GRAPHPLAN's planning graph and for further information consult reference [9].

### IV. METHODOLOGICAL APPROACH

In this section, we discuss the followed steps, covering from modeling to system implementation. In general, the methodology consists of: defining our scenario, finding all constraints related to it, designing the system architecture, integrating the developed module with the simulator, and validating the approach showing robot's trajectory constructed with the planner.

#### A. Modeling

In this section, we describe how our problem was modeled, and which abstractions were used to achieve it. We also describe how to identify complexities and constraints and, from that, simplify the model without losing key aspects, such as a certain degree of realism. We divided the modeling in various steps to be more intelligible, because it has a long sequence of details to be understood.

The problem was retired from soccer games with humanoid robots, being the chosen situation extremely frequent during

these games. Thus, solving this situation efficiently is vital in the team overall performance. The scenario is presented in Figure 3, consisting in a robot distant  $d$  from the ball with a different orientation of the ball's. The final goal is to put the ball in the target by kicking it and achieving to the condition of kicking in the most optimal way.

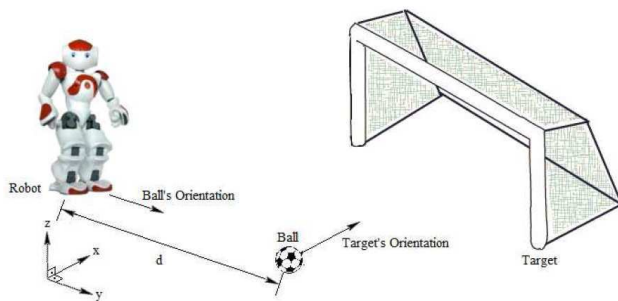


Figure 3. Used scenario

Further, observing the real robot operations during a soccer game the following actions were identified:

- Straight walk
- Curve walk
- Turn in place
- Kick

Also, the possible states were abstracted using some questions. We found all states cited above and the questions they try to answer:

- Having ball (Ha Ball) - is the ball near to robot or not?
- Ball aligned (Ball Dir) - is robot's direction aligned with the direction of ball to target?
- Ball in the Target (Ball Targ) - is the ball in target or not? (after kicking).

Table 1 shows the correlation between action and states. It was based on GRAPHPLAN modeling, extracting concepts of precondition to execute an action and effects of each action. Thus, to construct all correlations we enumerate various situations and extract them. The action Curve Walk is more complex than others, so we need to analyze it deeper.

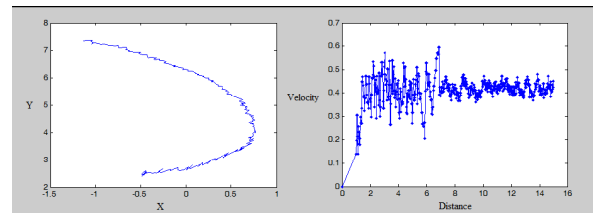
TABLE I. PRECONDITIONS AND EFFECTS OF ALL ACTIONS

Actions	Preconditions	Effects
Straight walk	$\neg$ Ha_Ball, Ball_Dir, $\neg$ Ball_Targ	Ha_Ball, Ball_Dir, $\neg$ Ball_Targ
Curve walk	$\neg$ Ha_Ball, $\neg$ Ball_Dir, $\neg$ Ball_Targ	Ball_Dir, $\neg$ Ball_Targ (Cannot determine state of Ha_Ball)
Turn in place	$\neg$ Ball_Dir, $\neg$ Ball_Targ	Ball_Dir, $\neg$ Ball_Targ
Kick	Ha_Ball, Ball_Dir, $\neg$ Ball_Targ	$\neg$ Ha_Ball, Ball_Dir, Ball_Targ

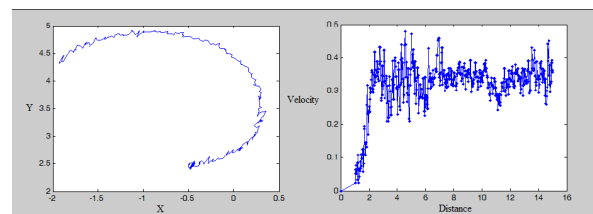
The robot Curve Walk has restrictions. It cannot perform curves with all radius, only discrete and established curves, due to the characteristics of the robot's joints. Consequently, we cannot determine the effect of this action with respect of having or not the ball. To suppress this, a tree structure was used. Additionally, the number of possible curves needed to be

decreased because of the limited capacity robot's processing power, being 3 curves the optimal number found.

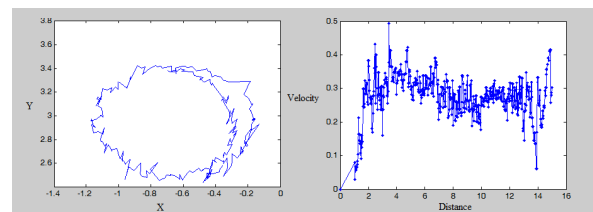
Therefore, the trajectory and velocity of the used curves are not perfect, due to the learning process of the robot. In Figure 4, the variation in the performed path can be observed. Finally, after modeling the problem the prototype was developed and will be explained in detail in the next section.



(a) Radius of 1.5



(b) Radius of 1



(c) Radius of 0.5

Figure 4. Left) Path trajectory for different Curves walkings. Right) Velocity for each radius of curve walking

### B. Prototype Development

The next step in our approach is to implement the prototype using the model described before. In order to fully achieve a functional system, more than just the planner itself is needed and the presentation of the complete architecture is the goal of this section.

In the broad view, the prototype will be a module inside the robot's agent already built, i.e., the humanoid robot is controlled by a software agent, counting on different modules (or behaviors), to guide the robot's action. This feature makes the platform scalable, so new modules can easily be added. In addition, to code the prototype C++ language was used.

Focusing in the system, the environment, as told before, is extremely dynamic and stochastic, because of that we cannot plan with some world state and then execute it without verifying again if the world state changed. We want a plan for the next 10 seconds, but how to solve it considering the

environment characteristics? By using an architecture, which will control the current plan execution and analyze if the planner's goal was achieved, and even predict if it can be achieved in the current conditions. However, to plan is a very time-consuming task. Because of planning complexity, and in order to have the robot operating in real-time, the robot has to avoid to plan frequently.



Figure 5. Process sequence in encountering a change in the environment

Figure 5 describes the process sequence in encountering a change in the environment; bein its dynamic explained as:

- The environment changes, so the roadmap (or strategy) must be updated to plan a path or not. After that the path planning block, by checking the updated roadmap, is going to create a new path graph. Further, the graph is updated and put in the roadmap to be checked again after the environment changing. The previous plan and the time of creating the plan is going to be in the roadmap.

Our general approach is presented in Figure 6. The sensor signals enter in the Monitor block, which decides if the plan must continue or not based on the world conditions and information of the current plan. Then, the modified world state has to go to the Planner, also details of the plan are sent to the Monitor for future controlling task described above. The planner's output is a sequence of actions for the robot, in our case they are represented by a set of speed and angle information.

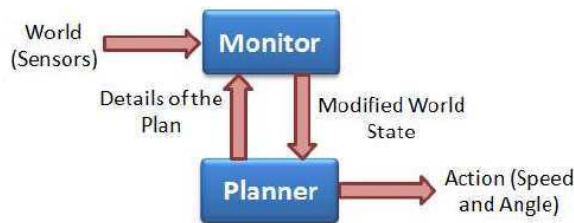


Figure 6. General approach diagram

Concerning the changing in the environment, we define three situations that determine that the plan has to be rebuilt: the ball position changed more than a threshold, the difference between robot's predicted position and robot's current position

is more than a threshold, and if after 10 seconds the robot does not reach the ball. Thus, define thresholds for ball and robot's relative positions is a hard task, demanding high numbers of experiences.

Inside the Planner, iterative deepening search is used to explore the tree. The tree expansion is shown in Figure 7, for each node the  $n$  possible actions are applied (preconditions must be considered), generating at most  $n$  prediction conditions. The graph growth is executed until the goal has been found, with maximum of 10 cycles (i.e. 10 seconds).

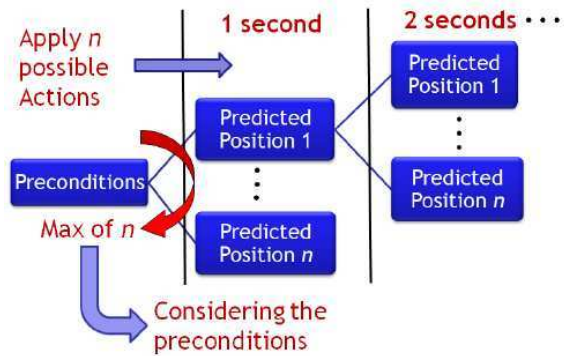


Figure 7. Manner to explore the tree

For sure, our goal state will be one of the states present in the tree. Therefore, combining the exploring the tree (shown before), with the sequence of time quantas, the first time the goal state in the tree is reached, we obtain the optimal and fastest solution.

We faced a time complexity constraint, presented in the modeling section. Thus, the robot plans for the next 10 seconds, each action is executed in 1 second. So, using the equation  $O(b^d)$  the  $d$  is 10 (extension of the plan) and  $b$  is 6 (number of possible actions). In respect of that, we can increase time efficiency of the algorithm in 3 ways: increasing execution time for the actions, reducing possible actions (using heuristics), discovering more preconditions and constraints.

This section aims, first, to prove the prototype functionality and, then evaluate the performance of our approach in some established situations. Several illustrative examples, showing how the system behaved in simulated environment, are presented here. We presented a tree-based approach integrated with planning paradigms for exploring and finding optimum path to achieve to ball in order to kick the ball to defined target.

The Planner tries to plan different skills with different characteristics in order to control the path trajectory of the humanoid robots. Further, two experiences were realized to see how the prototype behaves when it is coupled together with others modules of the agent. The same scenario was used and just the robot's position was varied in order to confirm the plan construction for different situations. The simulator Rcssserver3D [10] was used to test the scenario described above.

In this scenario, the robot is initialized in a position  $r$  in Cartesian coordinates where its center is the center of the field.

It plans to move the ball (position  $b$ ) to the goal target where its global position in the field is  $t$ . For the first experience, the values were  $r = (-1.7; -1.11)$ ,  $b = (0; 0)$ , and  $t = (7; 0)$ . The second experience is similar to the first one, however the value of  $r$  is  $(-3.09; 0.37)$ .

Figure 8, illustrates the expanded tree used to explore and search aiming to find the optimal plan and, also, shows the resultant plan as a set of positions extracted from the tree. The route of the tree is the Cartesian position of the player in the field. Therefore, Figure 9 shows the robot's position trajectory while it utilizes the produced path plan to reach the ball. As plan's results, we can see the ball's positions in the field when it was kicked by the robot being its trajectory also shown.

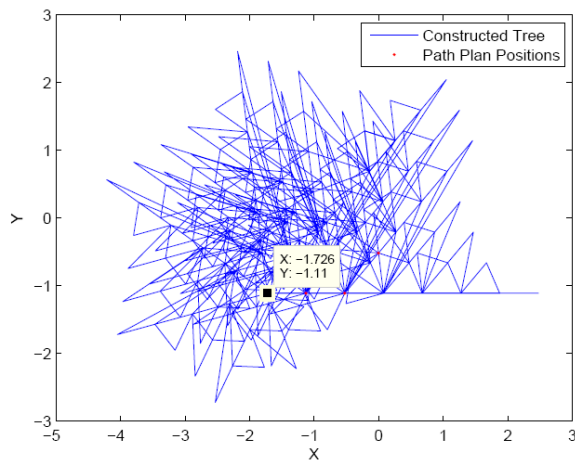


Figure 8. Tree of plans and chosen plan

In Table 2, the details of the path plan can be found. The plan starts from the robot's position and ends having the ball, which satisfies the precondition for the kick. Also, Figure 9 illustrates the robot's trajectory while it uses the path.

TABLE II. SEQUENCE OF POSITION FOR THE DESIGNED PLAN

Action Order	Position X	Position Y
1	-1.72624	-1.11048
2	-1.12624	-1.11048
3	-0.526237	-1.11048
4	-0.526237	-1.11048
5	-0.526237	-1.11048
6	-0.00301497	-0.515413

We observed certain advantages as: the robot using the prototype could accomplish the goal, i.e. walk to the ball in the shortest time possible (optimal path), then kick the ball in the target direction. Nonetheless, as observed in Figure 8, the search space is overly explored resulting in a loss of processing time (e.g. the robot explore and search also in some direction opposite the ball).

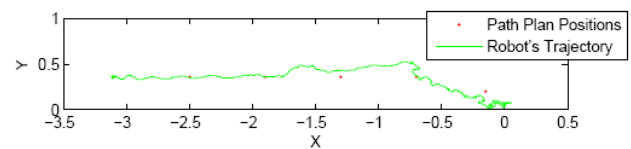


Figure 9. Robot's and ball's trajectories (situation 2)

Finally, comparing Figure 8 and 9, we can see that the robot's speed interferes in the path plan execution, i.e., when the robot walks at high speeds it tends to leave the projected trajectory. This result that in the end of the plan the robot cannot achieves to the desired point and, consequently, monitor has to replan.

## V. CONCLUSIONS

Planning aims to find a sequence of actions to reach to a final goal state, being its techniques extremely used for motion planning. A complex environment to deal with is the soccer game. It is even more complex if the scenario includes humanoid robots, which are being increasingly applied in this environment. Also, for humanoid robots, the planning techniques are not developed enough.

A new approach was presented to correlate path planning for humanoid robot in the soccer game environment. Further, the situation of a robot trying to reach the ball and kick it to the target, was modeled using preconditions and actions concepts which results in a realist, but simple, model of the problem. Furthermore, we implemented the prototype (modeled as a iterative deepening search), that could achieve the optimal solution. Experimental results validate the approach, in path planning issue, for humanoid soccer robots, to perform their tasks in the best manner.

There is a sequence of works that could follow the present work, we point some of them. First, use learning methods to discover unknown parameters, such as optimize the planner's time window. Also, add more heuristics and analyze more preconditions to reduce possible actions, i.e., we need to render the model in more detail to search new relations between preconditions and actions. To insert capabilities for obstacle avoidance is a simple next step because the system already has access to the robot position, so it just needs the obstacle positions and filter the plans that originate collisions.

## ACKNOWLEDGEMENTS

The first author is supported by FCT under grant SFRH/BD/66597/2009. This work has been partially funded by FCT Project ACORD - Adaptive Coordination of Robotic Teams (PTDC/EIA/70695/2006).

## REFERENCES

- [1] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, H. Inoue, Online footstep planning for humanoid robots. In Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA03). (2003) 932-937
- [2] O. Lorch, A. Albert, J. Denk, M. Gerecke, R. Cupec, J. Seara, W. Gerth, G. Schmidt, Experiments in vision-guided biped walking. Vol. 3. (2002) 2484 - 2490
- [3] J. Gutmann, M. Fukuchi, M. Fujita, Real-time path planning for humanoid robot navigation. In Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI05). (2005) 1232-1237

- [4] N. Shafii, L. Paulo, N. Lau, Biped walking using coronal and sagittal movements based on truncated fourier series. In Proc. of the RoboCup-2010: Robot Soccer World Cup XIII. (2010)
- [5] N. Shafii, A. Khorsandian, A. Abdolmaleki, B. Jozi: An optimized gait generator based on fourier series towards fast and robust biped locomotion involving arms swing. (aug. 2009) pp. 2018 –2023
- [6] Robotics, A.: Nao webpage (2010) Available in <http://www.aldebaranrobotics.com/en/naoeducation>, accessed in September 14.
- [7] A.J.S.B. Tay: Walking nao omnidirectional bipedal locomotion (August 2009) Master Thesis, University of New South Walse, School of Computer Science and Engineering.
- [8] S.J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach. Pearson Education (2003)
- [9] S. Kambhampati, E. Parker, E. Lambrecht, Understanding and extending graphplan. In Proc. 4th European Conference on Planning. (1997) 260–272
- [10] SourceForge: The robocup soccer simulator (2010) Available in <http://www.aldebaranrobotics.com/en/naoeducation>, accessed in September