Geometrical approach to On line Trajectory generation, Obstacle avoidance and Footstep planning for a Humanoid Robot

Kaustubh Nawade[1] V. Aditya[2] Apoorv Shrivastava[3] B. K. Rout[4]

Abstract— This paper presents a unique real time technique for path generation in dynamic environment using simple geometry, suitable for humanoid robots. We show that the proposed algorithm reduces computation to find the optimal collision free path by utilizing the geometrical shapes of the obstacles to navigate from a known initial point to final point. The trajectory is then translated to known parameters of humanoid gait model thus effectively computing the footsteps for the robot. This algorithm has been simulated, implemented and tested on humanoid robot AcYut, indigenously developed at Centre for Robotics and Intelligence Systems, BITS-Pilani.

Index Terms— Motion and Trajectory Generation, Humanoid Robots, Humanoid and Bipedal Locomotion, RoboCup

I. Introduction

A basic skill in autonomous systems is the ability to reach a desired location by making collision free paths. Many path finding algorithms have been designed till date to address this issue. Once a path is made it is then mapped to the gait target vector of the humanoid robot and footsteps are planed accordingly.

Mainly such prototypes have an on board processing system and thus the computational resources are compromised due to restriction in power, weight, heat dissipation issues etc. Since many situations encountered in real world scenario are dynamic, it is extremely essential that the algorithm for such calculations should be robust and computationally minimalistic. Moreover the challenge further increases due to the limited capabilities of the robot's sensors and their noise, thus limiting its ability to perceive the environment accurately.

The usage scenario for our case is humanoid robot soccer at RoboCup. It presents a challenging and an interesting goal to achieve. In this paper we try to address one of the most primary higher level tasks. Once the ball position is known and the robot is localized in the environment, the most important challenge is to find the smallest trajectory from the current position to the ball avoiding opponents and the team members in the way. Once the path is known it is then simply mapped to gait generation algorithm and the bot follows the steps planned for it.

However many path planning algorithms have high time and space complexity due to the formation and computation of

[1]B.E. (Hons.) Electrical & Electronics & M.Sc. (Hons.) Physics Undergraduate, BITS - Pilani `kaustubh.nawade@gmail.com`
[2]B.E. (Hons.) Computer Science Undergraduate, BITS - Pilani `42vaditya@gmail.com`
[3]B.E. (Hons.) Mechanical, BITS - Pilani `apoorvs92@gmail.com`
[3]Associate Professor, Dept. of Mech. Engg., BITS - Pilani `rout@pilani.bits-pilani.ac.in`

grid-based operations. In our case we can take advantage of the fact that obstacles sizes are comparable to that of the robot itself and there are very few obstacles present on the way to the ball. The algorithm in this paper minimizes this cost and is capable of finding the most optimum path even for a large number of obstacles. To give a comprehensive solution we also try to address the issue of mapping the planned trajectory to the humanoid robot's gait generation module, i.e. footstep planning.

The initial section of the paper reviews the existing literature on path planning and various current algorithms that can be used in such scenario. The subsequent sections describe a geometry based approach to path planning and discusses the algorithm in detail. We then discuss the mapping of this path to the gait of the humanoid robot which uses 3D-LIPM as the walking pattern generator. Last few sections shows the details of the tests and comparison of the algorithm with a few existing ones to finally conclude the paper.

II. Related Work

The problem of motion planning in the existing literature can be broadly categorized into two primary alternatives based on the method of computation of a trajectory. While one focuses on directly computing footstep location based on certain heuristic [1][2], other focuses on path planning and then mapping gait motion to the generated path. In this paper we focus on the latter as it cleanly decouples the method into two computationally less expensive methods.

In Robocup the world scenario is known and the environment is dynamic, i.e. obstacles are moving. Hence it is not important that the path calculated should be correct at all iterations. The system can accept wrong solution once in a few frames as long as it is fast. Thus the key challenge in motion planning algorithms is to achieve the computation complexity to sustain real time planning and execution. Lina Ourima [3] explored a path planning approach based on geometry using polygon and Beizer curves. This approach was used in the Robocup small size league by FU-Fighter's. Bennewitz and Burgard[4] search for an optimal path using grids and A* algorithm. For multirobot system it resolves conflicts between moving robots. However such search algorithms based on grid compromise on resolution for computation. A alternative solution to this problem was given by Ricards Steffens[5] exploring multiresolution path planning for dynamic environments. Other common approaches used by many teams is that of Artificial Potential Mapping[6]. This work mainly treats obstacles and test bot as positive point charges and then finds the most optimum path to the

negatively charged final position, based on field potential in the neighborhood. Though this algorithm can give high frame rates due to low complexity it lags to due to problems in local minima and the path final path is not very smooth. D* Lite[7] is also a widely followed approach for dynamic path planning

A comprehensive solution to motion planning is discussed in [8]. The approach begins making a simple path and generates paths of higher complexity till it satisfies the desired criteria, also accounting for stepping over obstacles along the way. A mapping from gait velocity to footstep location has been detailed in [9] by Andreas Schmitz of Team Nimbro, Robocup Teen size league. In this paper we design a position controller as greater accuracy is achieved by directly transforming path to footstep locations, instead of feeding gait velocities which would then be transformed in to footstep locations.

The driving intent behind this paper is to make a comprehensive motion model system which can work in real time. The idea of path planning and then footstep planning is outlined in such a way that it can be easily ported to another humanoid system without having to make any low level changes.

III. ROBOT TEST PLATFORM

The implementation and testing of the algorithm is carried out on a indigenously developed humanoid AcYut. The humanoid has been used in the recent versions of Robocup tournament and has 24 degrees of freedom. The environment is perceived by a IDS UI-1221LE fish eye lens at 74 fps. The humanoid is equipped with an onboard x86 Intel Atom processor running at 1.8 GHz. It has 4 Gb of RAM and 32 GB of persistent flash memory. The processor has low power consumption with the trade off of low computational power. Hence the requirement of developing software systems and algorithms with low computational complexity.

IV. METHODOLOGY

A. Usage Scenario

For a robot soccer game in the humanoid teen size league at RoboCup each team has two autonomous mobile robots, a number which will increase in the near future. Each bot is around 20 cm in diameter and around 100 cm in height. The field is rectangular, 9m X 6m and a vision sensor usually a camera is mounted on top of the robot.

In the test robot platform, a fish eye lens is used since it gives a wide field of vision, thus giving clearer image frames, since tilting and panning are not necessary. Thus in one frame the two opponent bots, one team member bot, referee, robot handlers are perceived, and designated as obstacles and the path to the ball has to be navigated around them. The obstacles and the ball are moving in real time and hence the path to be computed also needs to be dynamic and updated almost every second for an accurate path. Moreover, the data computed by localization is not accurate, neither are the steps followed by the robot on the path generated. An algorithm like grid search can be very costly as the computation may take up to a few seconds and the bot will have to stand still in that duration which is not desirable.

B. Algorithm

In this approach of geometric path planning all obstacles are represented by polygons. In the implementation discussed in this paper we treat them as circles since complex representation of the shape of the obstacles is neither useful nor trivial for the current problem specification.This is a valid assumption as the obstacles are mainly humanoids which can be generalized to a circular projection on the 2D surface. A graph called the visibility graph[12] is constructed between the start and end point where the nodes are points on the polygons and the best path is calculated accordingly as per cost. There is an edge connecting the two nodes only if the edge is not intersecting or cutting any other polygon. In the event of an intersection, the nodes are connected between the two polygons and the intersecting polygon. Connect implies adding cost for the Edges (E) that connect the Vertex (V) as per the Euclidean distances between them.

Algorithm 1 Calculate Visibility Adjacency Graph G

1: Initialize a = Start Point, b = End Point
2: Connect a and b by a straight line
3: **procedure** VISIBILITYGRAPH(a, b)
4:     Connect a and b in G
5:     **if** Line Intersects an Obstacle **then**
6:         Tangents from a and b to the first obstacle encountered o, at $o1$, $o2$, $o3$, $o4$
7:         Disconnect a and b in G
8:         Connect $o1$ with $o3$ and $o2$ with $o4$ in G
9:         **for** $i \leftarrow o1$ and $o2$ **do**
10:             Connect a and i in G
11:             VisibilityGraph(a, i)
12:         **end for**
13:         **for** $i \leftarrow o3$ and $o4$ **do**
14:             Connect i and b in G
15:             VisibilityGraph(i, b)
16:         **end for**
17:     **else**
18:         Continue
19:     **end if**
20: **end procedure**

Thus in this case since the polygons are treated as circles, if a common tangent to the two circles intersects a circle then common tangents are drawn to this new circle. This process is repeated until we have the entire visibility graph, connecting the start and end point as shown step by step in Algorithm 1. The function of Visibility Graph is called recursively on each of the edge joining two nodes. Thus the number of points in the graph depends on the number of obstacles present on the field which is usually in 3 and in maximum cases 5 including possible erroneous cases. Then to calculate the final trajectory the least cost is found for a path connecting the start and the end point. The entire
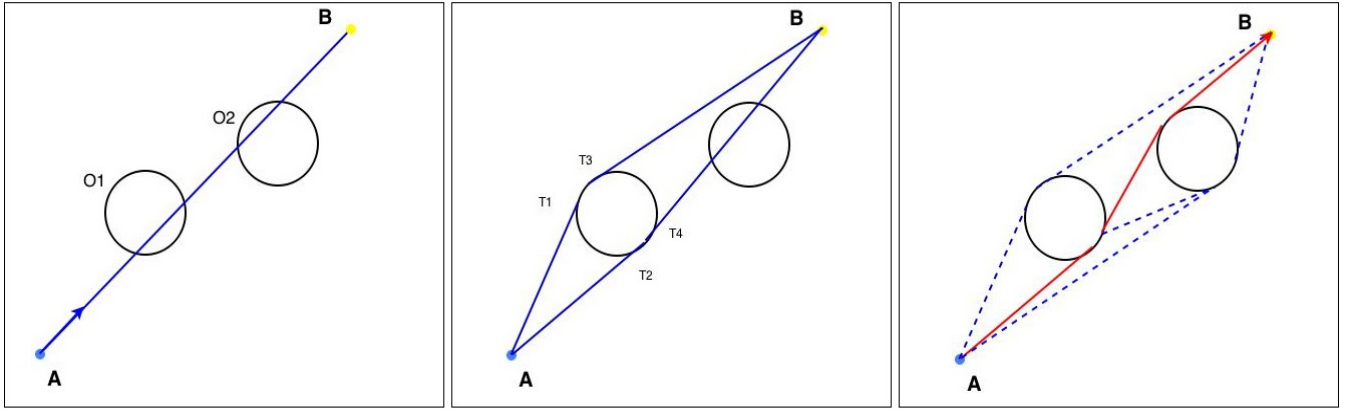
Fig. 1: (a)Start point A and End Point B hindered by two obstacles (b) Drawing tangents to the first obstacle encountered at T_1, T_2, T_3 and T_4. (c) Recursively following the same function. Final path as shown in red.

algorithm is diagrammatically represented in Fig. 1.

Now to find the shortest path between the start and end point its all about finding the shortest cost between them in the Visibility Graph. For this purpose Dijkstra's Algorithm[10] is used as it is asymptotically the fastest known single source shortest path algorithm for arbitrary directed graphs with unbounded non-negative weights. Any other graph search algorithm can also be used in this case.

C. Orientation Circles

Joining line between the start and end points is not sufficient to give complete path between the two points. It often is required that the bot face a particular direction once it reaches the destination. In this scenario in particular, the humanoid is required to reach the ball, dribble or kick ball towards the goal posts to simply ensure that it does not shoot in the wrong direction or pass the ball to another player or hit another obstacle. One alternative is to simple reach the final point and then orient itself correctly, which leads to dependency on side walk in a circle. A much efficient solution would be consider the correct orientation in the path itself such that when the bot reaches the final point it finds itself oriented in a correct manner. This can be done by simple addition of dummy obstacles of required radius orthogonal to the facing direction, and choosing the final path intelligently. The diagrammatic representation is as shown in the figure 2.

Another use of introduction of dummy obstacles is to start the tracing of the path smoothly. It often happens that the bot is facing a direction and need to start walking at a specified angle to the current gait vector. Since instantaneous turning of the point can prove to be difficulty or time consuming an obstacle can be placed to ensure smooth transition.

D. Edge Cases and Erroneous Inputs

Since the algorithm is dynamic some cases will arise when the robot, obstacle, ball all are very near, especially when the bot progressively reaching its target. In such cases due to erroneous inputs from the camera, sometimes the bot or end point can be found inside the obstacle itself, making
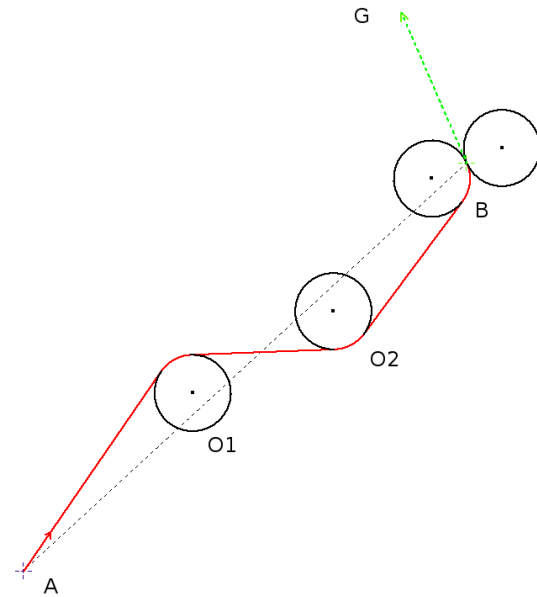

Fig. 2: *Use of orientation circle or dummy obstacles to align the robot correctly. The robot starts from point A, navigates it way around the obstacles O_1 and O_2, to reach the point B facing the direction G. .*

it impossible to draw tangents. Such cases are ignored and since the frame rate is as high as 50 fps these instance do not matter much and are taken care of in the subsequent frames. Other cases include when obstacles intersect. This may be because of the erroneous inputs if obstacles are behind one another or because of the generic assumption that obstacles all have the same radius. This in addition to the fact that obstacles radius is its radius added with the half of radius of the bot itself so that the test subject can easily maneuver between obstacles. To solve this problem any two such intersecting obstacles are treated as one obstacle, by making a best fitting polygon usually an ellipse on top of the obstacles. The algorithm can be now executed normally considering this new ellipse.
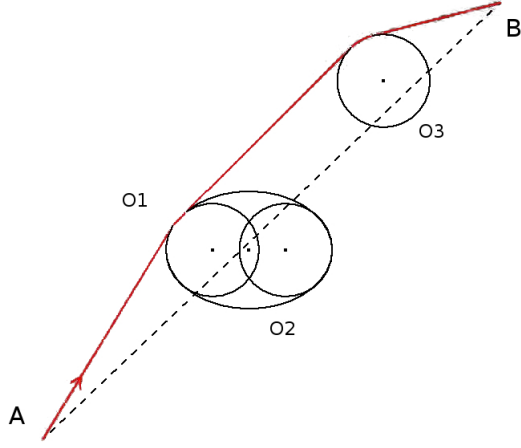
Fig. 3: *Dealing with intersection of obstacles due to erronous inputs.*

E. Step Planning

Once the path is generated it is then mapped to the humanoid gait Humanoid walk comprises the following movements in a cycle

1) Single Support Phase (SSP): A leg is raised from the ground, with the other is the support foot.
2) Double Support Phase (DSP): Both feet are on the ground.
3) The moving leg and support leg are interchanged.

3D Linear Inverted Pendulum Model describes the motion of the Centre of Mass (CoM) in the robot[12]. The motion of support leg is opposite of the CoM, since the support leg behaves as a pivot which moves the CoM of the robot. Therefore the 3D LIPM algorithm can be used to develop a relation between step lengths and CoM velocities. The motion of the robot's supporting leg is defined by the x-variable in the Eq 1. Tc and Ts are constants, Ts being the SSP time of the robot. The initial and final states of the bot (before and after the step) are represented by i and f. Velocity and position are denoted by v and x.

$$\begin{pmatrix} x_f \\ v_f \end{pmatrix} = \begin{pmatrix} \cosh\left(\frac{T_s}{T_c}\right) & T_c \sinh\left(\frac{T_s}{T_c}\right) \\ \frac{\sinh\left(\frac{T_s}{T_c}\right)}{T_c} & \cosh\left(\frac{T_s}{T_c}\right) \end{pmatrix} \begin{pmatrix} x_i \\ v_i \end{pmatrix} \quad (1)$$

It is convenient to rewrite the above equations for the derivation as

$$x_f = A \sinh\left(T_s/T_c + \phi\right) \quad (2)$$

$$v_f = A \cosh\left(T_s/T_c + \phi\right) \quad (3)$$

where,

$$A = v_i \sqrt{T_c^2 - \left(\frac{x_i}{T_c}\right)^2} \quad (4)$$

$$\phi = \sinh^{-1}\left(\frac{x_i}{A}\right) \quad (5)$$

This reformulation represents a loss in generality, since it implicitly introduces the assumption that the humanoid only moves forward, and does not perform maneuvers such as a back walk. This implementation also introduces certain constraints on the maximum and minimum future velocity relative to the present velocity. However this reformulation is acceptable on humanoid robots since back walk can be modeled by mirroring the forward walk with a transitional motion in place, and using negative velocities to reverse the direction of motion.

By observing these constraints, a theoretical limit is placed on the acceleration of the humanoid which serves to restrain its motion and prevent it from attempting to increase its velocity excessively in a single step.In order to develop an effective Walk-Path interface, one must generate footstep positions which obey these constraints. The constraints are derived in the form of maximum and minimum ratios of initial and final velocity over a single step (v_f/v_i)

The constraints are derived to obtain the following range:

$$\left(\frac{v_f}{v_d}\right)_{min} = p - \sqrt{p^2 - 1} \quad (6)$$

and

$$\left(\frac{v_f}{v_d}\right)_{max} = p \quad (7)$$

Or,

$$(v_f)_{min} = \left(p - \sqrt{p^2 - 1}\right) v_d \quad (8)$$

$$(v_f)_{max} = p v_d \quad (9)$$

where

$$p = \cosh\left(\frac{T_s}{T_c}\right) \quad (10)$$

It must be noted that the above range is independent of all other earlier states, such as v_i and y_i, which means that the range applies to any step considered independently.

The footstep planning interface computes step positions which obey the specified constraints. The motion is broadly divided into two categories:

- Straight Line Path
- Circular Path

When the designated trajectory is a straight line, the robot accelerates to its maximum velocity at a rate which obeys the velocity constraints. When the trajectory is circular, the robot must decrease its velocity to avoid instability. However, it is not possible to have a constant velocity when navigating a circular path, because the steps taken by the foot on the inner side of the circle are shorter than the outer foot. Therefore, the velocity stabilizes by oscillating between two different values.

A second consideration is the transition between a straight line trajectory to a circular one, and vice versa. At any frame the new path generated may possess a straight line or a circle. Given the present velocity and step location, the interface must generate stable transition of velocity (acceleration or

deceleration) to adapt to the new trajectory. This situation is accommodated by allowing a margin of one or two steps to transition safely to the new trajectory. The inaccuracy introduced by this margin is negligible due to the high step frequency of the robot.

The footstep positions generated by a given path have been pictorially presented in Fig 4. It may be observed from the image that the step lengths are considerably less along the circular arcs. This indicates the decreased velocity of the robot when turning.
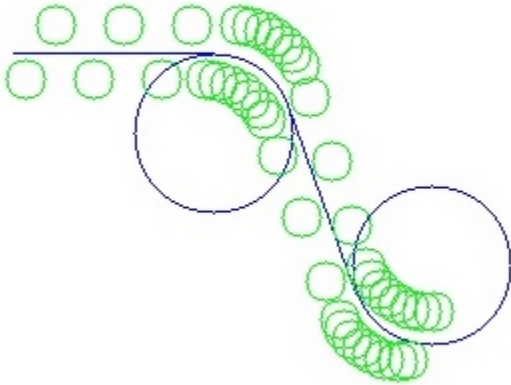

Fig. 5: *The use of path planning for multiple obstacles.*

Fig. 4: *Footstep locations for the given path.*

In the dynamic scenario some oscillation at a point were observed where the robot has to choose between two equally probable paths. Such edge cases can be easily be solved by either random selection or adjusting the frequency of mapping final path to the gait vector.

V. EXPERIMENT AND RESULTS

Visibility graph in this case consists of $(4 * n + 2)$ number of points in the worst case, where n in the number of obstacles intersecting the line joining the start and end point, i.e. coming in the way of the bot to reach the goal position. The visibility graph can be calculated in worst case time of $O(n^2)$. Further more the Dijkstra's has a wost case performance of $O(4n + 4n * log(4n))$. Thus this algorithm decreases the number of node points on which a graph search can be applied with ease.

The footstep planning interface along with the path planning were integrated with the behavior layer of the humanoid for testing purposes. For the experimental setup, two obstacles of radius 15 cm each were placed in the way of the bot and the ball. The bot had to navigate its way around the obstacles and align itself to hit the ball towards the goalposts. Both the scenarios where obstacles were static and dynamic were devised to test the efficacy of the algorithm. The tests were conducted for a total of more than 200 configurations in both cases. The advantages which were very apparent that the bot was able to correct its trajectory when it lost track of the path, due to loosing its stability or some mechanical disturbances like friction.
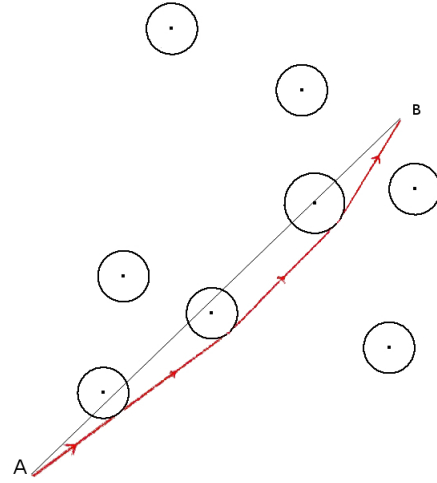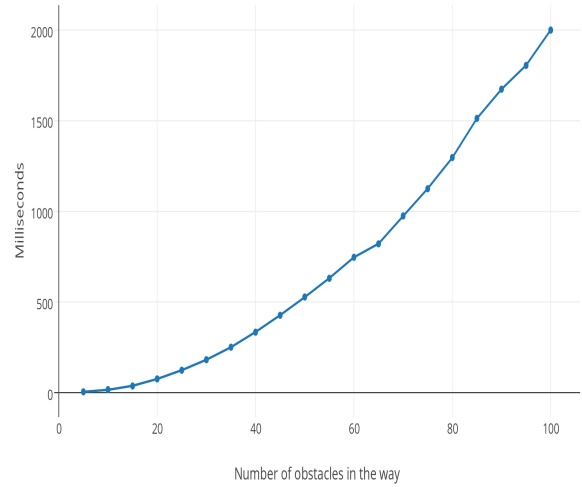

Fig. 6: *Time dependency of the algorithm on the number of obstacles in the robot's way.*

The number of obstacles particularly in Robocup are small. There can be at a time a maximum of 3 obstacles. The worst case scenario that all three obstacles are in the robot's way, would involve the calculation of shortest path algorithm on a visibility graph consisting of just 14 points. When tested on the actual platform with 3 obstacles the average time taken to calculate was around 4 ms. This time is sufficient to give the entire cognition module a frame rate of 50 fps. Thus at such high frame rates, the dynamic nature of the obstacles is

easily taken care of. The noise from the sensors like camera and the deviation of walk are thus handled naturally as a result of this. Fig. 5 shows the results of using the algorithm for an arbitrary configuration that might occur in a technical challenge in Robocup where the bot has to navigate its way and dribble a ball along with it. Testing this algorithm for up to 100 obstacles in the robots way gave the results as shown in Fig 6.

VI. COMPARISON

The algorithm presented here is compared with the other standard planning algorithms in terms of time taken, and no. of operations. The algorithm discussed in this paper is applicable to very specific cases, especially when the size of the obstacle is comparable to that of the robot itself. One of the main advantages of this geometry based planning algorithm is the non dependency on grids. Usually algorithms based on grids computationally expensive or do not have high resolution. The algorithm presented works in a grid less environment and only considers the nodes of interest, instead of analyzing each and every point on the map. Moreover it is not restricted to frame area of any size whatsoever, meaning the obstacles, final position can be located anywhere with reference to the navigating object without any cost to computation.

For comparing this algorithm with the other standard ones in the literature we considered the setup of 2 obstacles along the way to the final point as shown in Fig. 7. The results of the comparison are as tabulated in Table 1.

Algorithm	Time (ms)	No. of Operations
Geometry based	3.6	84
A^*,	5	243
Breadth First Search	13	1013
Best First Search	5	95
Dijkstra	18	1161
Artificial Potential Mapping	6	57

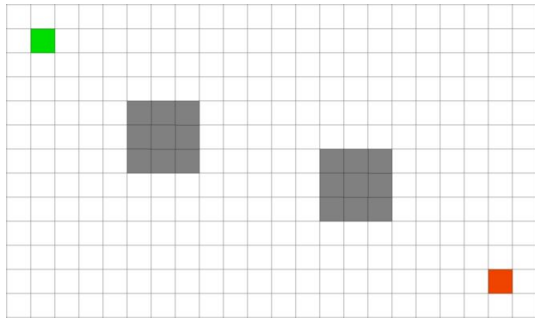Table 1: Comparison of Various Path Planning Algorithms

Fig. 7: *Testing variables for various algorithms above.*

VII. CONCLUSION

We have presented an unique way for path planning which is best suitable to the case when the obstacles size is comparable to the robots. In such cases we can take advantage of the robots shape which can be approximated to simple circles to find the best way possible around them using simple geometry. This way we reduce the number of node points on which a graph search can be applied. This reduces the computation time significantly thus giving better frame rates for the execution of code on an on board processor. This way the dynamic nature of the robot soccer games can easily be taken care of along with any noise from the sensors and deviation from the walk. This is then ported to the 3D-LIPM walking pattern generator to adjust the footsteps, gait in such a way that the robot is stable while following the given path.

VIII. ACKNOWLEDGMENT

REFERENCES

[1] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade, "Footstep planning for the Honda ASIMO humanoid", in *Proc. of the IEEE Int. Conf. on Robotics and Automation* (ICRA?05), April 2005.

[2] Y. Ayaz, T. Owa, T. Tsujita, A. Konno, K. Munawar, and M. Uchiyama, "Footstep planning for humanoid robots among obstacles of various types", in *Proc. IEEE-RAS Conf. Humanoid Robots*, Dec. 2009, pp. 361-366.

[3] Lina Ourima, "Fast Geometric Path Planning for the Small Size Robocup League", Friei Universitat Berlin, June 2004

[4] Maren Bennewitz and Wolfram Burgard, Finding Solvable Priority Schemes for Decoupled Path Planning Techniques for Teams of Mobile Robots, *PROCEEDINGS OF THE 9TH INTERNATIONAL SYMPOSIUM ON INTELLIGENT ROBOTIC SYSTEMS*, 2001

[5] Ricarda Steffens et. al., Multiresolution Path Planning in Dynamic Environments for the Standard Platform League, *Proceedings of the 5th Workshop on Humanoid Soccer Robots @ Humanoids 2010*, Dec. 2010

[6] Oussama Khatib, Real-time obstacle avoidance for manipulators and mobile robots," *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, St. Louis, MO, May 1985

[7] Sven Koenig et. al., D* Lite, American Association for Artificial Intelligence, 2002

[8] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami, ?Planning biped navigation strategies in complex environments,? in *Proc. of the IEEERAS/RSJ Int. Conf. on Humanoid Robots (Humanoids?03)*, Munich, Germany, October 2003.

[9] Andreas Schmitz, Marcell Missura, and Sven Behnke, "Real-time trajectory generation by offline footstep planning for a humanoid soccer robot", In *Proceedings of 15th RoboCup International Symposium, Istanbul*, 2011.

[10] Huijuan Wang et. al., Application of Dijkstra algorithm in robot path-planning,

[11] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi and H. Hirukawa, "The 3D linear inverted pendulum model: a simple modeling for a biped walking pattern generation", *In IROS*, 2001, Vol. 1, pp. 239-246

[12] H. Alt and E. Welzi, Visibility Graphs and Obstacle Avoiding shortest paths, *Zeitschrift fr Operations Research*,1998