PHILIPPINE ENGINEERING JOURNAL
PEJ 2005; **Vol. 26 No. 1**:1–20

A Path Planning Algorithm for Soccer Playing Robots Based on Repeated Modification of Bezier Polynomials

Michael Angelo A. Pedrasa*

Department of Electrical and Electronics Engineering
University of the Philippines Diliman
Quezon City 1101 PHILIPPINES

ABSTRACT

The Bezier curvature algorithm is a path-planning algorithm for soccer robots based on repeated modification of Bezier polynomials. The robot is steered by choosing an appropriate Bezier curve that connects its current position and its destination, and the robot is made to traverse the initial section of the curve. The wheel velocities of the differentially-driven robot are computed from the curvature of that curve section. The process is repeated until the robot finally reaches its destination. The algorithm was compared to two other path-planning algorithms for soccer robots: the uni-vector field and line-circle algorithms. The algorithms were made to execute 17 test cases and their performances were compared. The test cases have different robot, ball and obstacle locations. The objective of the robot is to kick the ball towards the center of the target goal. The goal success rate, kicking accuracy and frequency of collisions were compared. The results showed that the principles behind the Bezier curvature algorithm are valid, and it performed better than the two algorithms.

Keywords: path planning, robot soccer, robot soccer simulation, collision avoidance, repeated path modification

1. INTRODUCTION

A robot soccer game is played between two teams, where each team has at least three wheeled robots. A typical robot soccer setup is shown in Figure 1. During a game, a camera mounted above the playing field repeatedly captures and transmits images of the playing field to the computer. In the computer, an image processing routine calculates the ball and robot coordinates. These data are used by an artificial intelligence (AI) routine to determine how the robots would react. The commands are then sent to the robots using an RF transmitter.

The AI program running in the computer has two modules. The first module implements the game strategy. It deals with the execution of the offensive and defensive strategies as conceived by the programmer. It determines where each robot should go given a field situation. On the other hand, the second module deals with path planning. After the strategy module has determined the destination of each robot, the path planning module steers the robots so that they will reach their destination.

*Correspondence to: Department of Electrical and Electronics Engineering, University of the Philippines Diliman, Quezon City 1101 PHILIPPINES. email:mapedrasa@up.edu.ph

Received April 21, 2005
Revised January 22, 2007
Accepted March 30, 2007

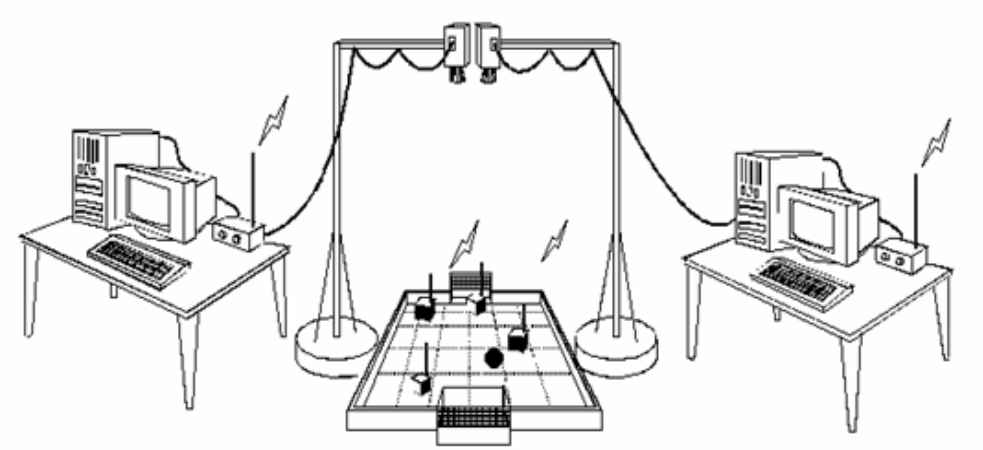

Figure 1. A Typical Robot Soccer Setup

In most robot soccer systems, path planning has two phases: trajectory generation and robot control. In the first phase, the path is generated by (1) calculating the set of points that make up the path, or by (2) calculating the heading of the robot at any point along the path, or by (3) synthesizing an equation that describes the path. The robot control phase computes the wheel velocities of the robot so it will follow the path.

The path planning module should be fast because the image-processing module consumes an ample amount of processing time, and little is left for the strategy and path-planning modules. Furthermore, the strategy module generally consumes more processing time than the path planning module. The path planning module should be simple so that it will not be computationally intensive and the robot will exhibit predictable behavior. It should be accurate so the robots would reach the planned destinations.

Path planning algorithms are classified into two categories. The first are algorithms based on repeated path modification while the second are algorithms based on topological network representations [1]. In the former, an initially formulated path is modified until it satisfies all practical constraints. In the latter, the workspace of the robot is subdivided into regions, and is represented as a topological network where a node in the network corresponds to a region. The solution to finding a path that connects one region to another is similar to finding a path that connects the corresponding nodes in the network. In this paper, a path planning algorithm called the Bezier curvature algorithm is proposed and its performance is compared to other path planners: the uni-vector and line-circle algorithms. The bezier curvature and uni-vector algorithms are based on repeated path modification while the line-circle algorithm is based on topological network representation.

Soccer robots are usually differentially driven, that is, they have two independent coaxial wheels. The robot is steered by controlling the left and right wheel velocities. Equations (1) and (2) show the relationship between the left (V_L) and right (V_R) wheel velocities, and the

curvature of its trajectory.

$$V_L = V_C \left(1 + \frac{d}{2\rho}\right) \tag{1}$$

$$V_R = V_C \left(1 - \frac{d}{2\rho}\right) \tag{2}$$

V_L = left wheel velocity
V_R = right wheel velocity
V_C = average velocity
d = robot width
ρ= curvature of trajectory

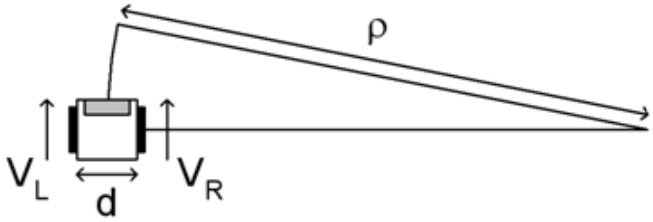

Figure 2. Modeling of a Soccer Robot

The organization of the rest of the paper is as follows. The Bezier curvature algorithm is described in Chapter 2. The comparison of the Bezier curvature to the uni-vector and line-circle algorithms is described in Chapter 3. The results are presented in Chapter 4. Finally, the conclusions are given in Chapter 5.

2. THE BEZIER CURVATURE ALGORITHM

This chapter describes the Bezier curvature algorithm. The first part describes the Bezier curve, the inspiration behind the development of the path planning algorithm. The second part focuses on the Bezier curvature algorithm.

2.1. The Bezier Curve

The Bezier curve was named after Pierre Bezier, the French engineer who used these curves to design the Renault car in the 1970's [2]. An n^{th} order Bezier curve is described by a parametric equation generated using n + 1 control points. The equation is given by

$$\mathbf{B}(u) = \sum_{k=0}^{n} \mathbf{p}_k \begin{pmatrix} n \\ k \end{pmatrix} u^k (1-u)^{n-k}, 0 \le u \le 1 \tag{3}$$

In the equation, u is the normalized time and $\mathbf{p}_k$ are the control points. As u varies from 0 to 1, a point traversing the curve passes by each control point one after the other.

The following are characteristics of the Bezier curves that were used to formulate the algorithm:

1. The curve does not pass through the control points except the first and last points.
2. The control points pull the curve towards their locations.
3. Adding multiple control points at a single position will add more weight to that point.
4. The tangent at the ends of the curve is along the line between the two points at the end.

Shown in Figure 3 is an example of a Bezier curve. The 4th order curve is defined by control points at A(0,0), B(2,0), C(3,6), D(6,6) and E(6,4). The curve is tangent to the line containing A and B and to the line containing D and E. It is also pulled towards the location of points B, C and D.

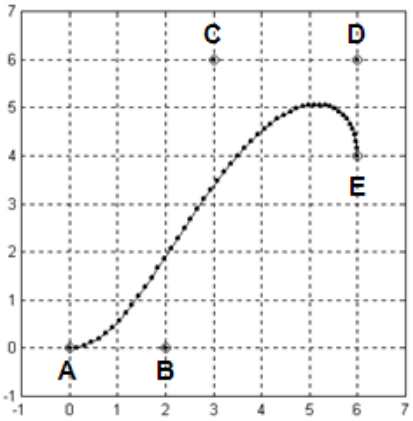

Figure 3. 4th order Bezier curve

2.2. The Bezier Curvature Algorithm

The Bezier curvature algorithm is a path planning algorithm based on repeated path modification. While the robot is moving, the path is modified to account the changes in the position of the robot and destination, and presence of obstacles.

The algorithm uses Bezier curves as paths. The Bezier curve is modified while the robot moves towards the ball, its target destination. The control points that define the shape of the curve are re-positioned until the robot reaches the ball. This approach enables the robot to avoid stationary or moving obstacles as it move. Figure 4 shows how the wheel velocities are computed.

Given the coordinates of the robot and ball, the first step is to select the control points so that the Bezier curve connects the robot and the ball. The locations of the obstacles are not considered in selecting the control points. Afterwards, a rectangular area in front of the robot is checked for the presence of obstacles. If an obstacle is present, another set of control points are selected. These control points create a new curve that would maneuver the robot around the obstacle.

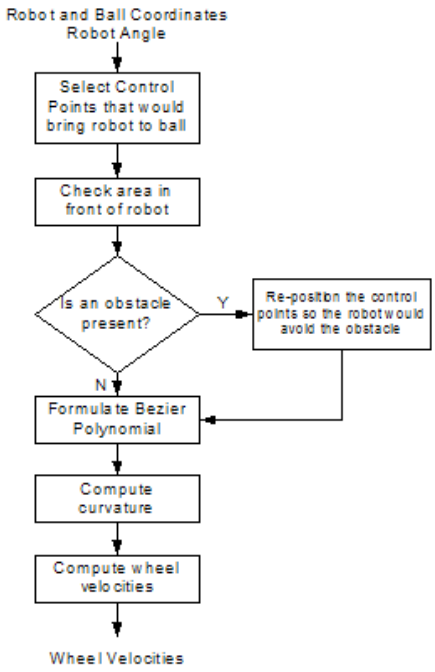

Figure 4. Computation of Wheel Velocities Using the Bezier Curvature Algorithm

The control points are then used as coefficients of the Bezier polynomial. Using the polynomial expression, the curvature at the location of the robot is computed. Then using equations (1) and (2), the wheel velocities are computed from the curvature. Since the robot covers a small distance during each cycle in the simulator, the robot is following only the initial portion of the Bezier curve. The process is repeated until the robot reaches the ball.

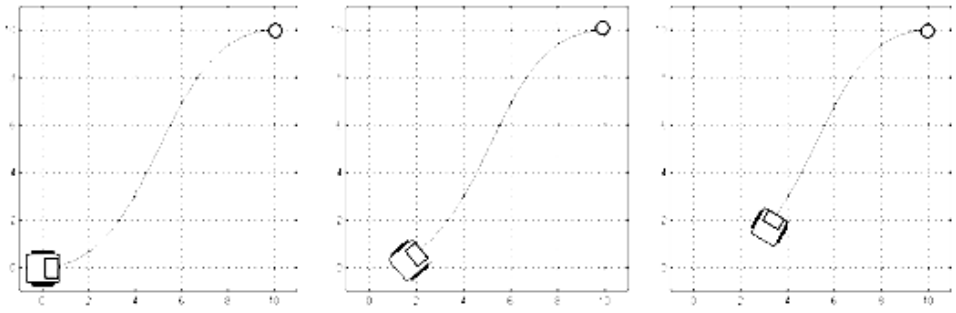

Figure 5. Navigation Using the Bezier Curvature Algorithm

Figure 5 shows the movement of the robot as travels towards the ball. In the first cycle, a

Bezier curve is generated and the robot follows the initial section of the curve. The first curve is shown in the leftmost frame. In the next cycle, the robot has moved to a new position and a new Bezier curve is generated. The robot is again directed to follow the initial section of the curve. This process is repeated until the robot reaches the ball. The second and third frames show the generated Bezier curves during the second and third cycles.

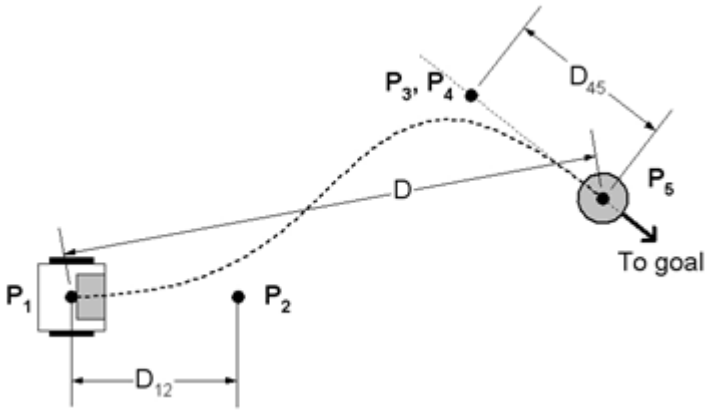

Figure 6. Selection of Control Points

The objective of the robot is to kick the ball towards the goal. The goal is a single point in space located at the middle of the opposite goal posts. This is accomplished by moving from its current position to the area behind the ball, and then makes contact with the ball while facing the goal, as indicated by the arrow. Furthermore, the robot should turn smoothly as it proceeds to the region behind the ball.

Figure 6 shows the control points chosen to form the Bezier curve. Five control points are chosen. Since the first and last control points are the endpoints of the curve, the robot and ball positions, $\mathbf{P}_1$ and $\mathbf{P}_5$, are chosen as the first and last points. To make the robot turn smoothly from its original position to face the ball, a point directly in front of the robot, $\mathbf{P}_2$, is chosen as the second control point. $\mathbf{P}_1$ and $\mathbf{P}_2$ lie on the line where the robot is facing. To steer the robot towards the region behind the ball, two control points, $\mathbf{P}_3$ and $\mathbf{P}_4$, are placed at the same location. They lie on the line containing the desired direction of the ball after impact. The distances of $\mathbf{P}_2$ from $\mathbf{P}_1$ (D_{12}) and of $\mathbf{P}_3$ and $\mathbf{P}_4$ from $\mathbf{P}_5$ (D_{45}) depend on the distance and orientation of the robot with respect to the ball.

A rectangular area in front of the robot is checked for the presence of obstacles (ally or opposing robot). If an obstacle is detected, the control points are re-positioned so the robot would avoid the obstacle. In Figure 7(a), the obstacle is to the right of the robot so the robot is maneuvers to the left. The control points are chosen so the robot would proceed to the side of the obstacle and have a line of sight to the ball. In Figure 7(b), the obstacle is to the left of the robot so the robot is maneuvered to the right. Again, $\mathbf{P}_1$ is the location of the robot, $\mathbf{P}_2$ is a point in front of the robot, $\mathbf{P}_3$ and $\mathbf{P}_4$ coincide and are located in the line of sight and $\mathbf{P}_5$ is a point beside the obstacle.

The Bezier polynomial is then formulated using the selected control points. Since five points

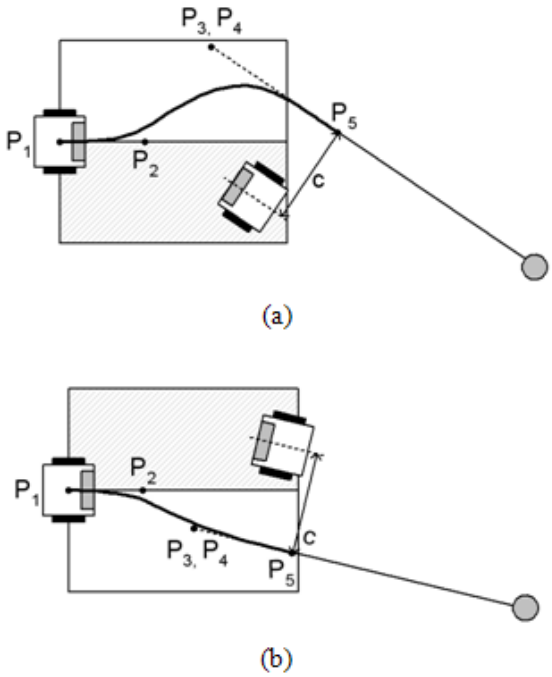

Figure 7. (a) Control points when the obstacle is to the right of the robot. (b) Control points when the obstacle is to the left of the robot.

are chosen, the Bezier curve is described by the following parametric equation:

$$\mathbf{B}(u) = \mathbf{P}_1 (1-u)^4 + \mathbf{P}_2 u (1-u)^3 + \mathbf{P}_3 u^2 (1-u)^2 + \mathbf{P}_4 u^3 (1-u) + \mathbf{P}_1 u^4 \tag{4}$$

The curvature at the location of the robot is then approximated from the parametric equation. The wheel velocities are then computed so the robot will traverse an arc with the computed curvature.

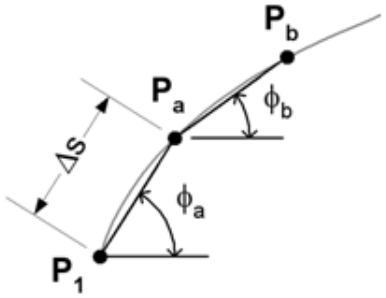

Figure 8. Approximation of Curvature at $\mathbf{P}_1$

Figure 8 shows the initial portion of the curve. $\mathbf{P}_1$(x_1,y_1) is the first control point, or the location of the robot, while $\mathbf{P}_a$ and $\mathbf{P}_b$ are points in the curve computed by $\mathbf{P}_a = \mathbf{B}(u_1) =$ (x_a,y_a) and $\mathbf{P}_b = \mathbf{B}(u_2) =$ (x_b,y_b), where $u_1 < u_2 << 1$.

Curvature is defined as the rate of change of the direction of the curve with respect to the change in its length [3], or

$$\sigma = \left| \frac{d\phi}{ds} \right| \tag{5}$$

Equation (5) may be approximated as

$$\sigma \approx \left| \frac{\Delta\phi}{\Delta s} \right| \tag{6}$$

Using Figure 9, equation (6) may be re-written as

$$\sigma = \left| \frac{\Delta\phi}{\Delta s} \right| = \left| \frac{\phi_b - \phi_a}{\Delta s} \right| \tag{7}$$

The angles ϕ_a and ϕ_b, and distance Δs are equal to

$$\phi_a = \tan^{-1} \frac{y_a - y_1}{x_a - x_1} \tag{8}$$

$$\phi_b = \tan^{-1} \frac{y_b - y_a}{x_b - x_a} \tag{9}$$

$$\Delta s = \sqrt{(x_a - x_1)^2 + (y_a - y_1)^2} \tag{10}$$

From the approximation of the curvature, the radius of curvature ρ is equal to

$$\rho = \frac{a}{\sigma} \tag{11}$$

The wheel velocities are then computed using equations (1) and (2).

3. COMPARING THE BEZIER CURVATURE WITH OTHER PATH-PLANNING ALGORITHMS

The first two sections of this chapter describe the path planners the Bezier curvature algorithm was compared with. The third section describes how the three algorithms were compared.

3.1. The Uni-Vector Algorithm

The uni-vector algorithm is a path planning algorithm based on repeated path modification that uses potential or force fields. In this algorithm, a force field is associated to each object in the soccer field. The direction of the robot is determined by adding the force fields due to all objects [4].

The force field associated with the ball is called the uni-vector field. The uni-vector field is shown in Figure 9(a). The streamlines of the force field show that the robot is guided towards the back of the ball, and push the ball towards the middle of the right goal. The magnitude of the force field at any point is constant.

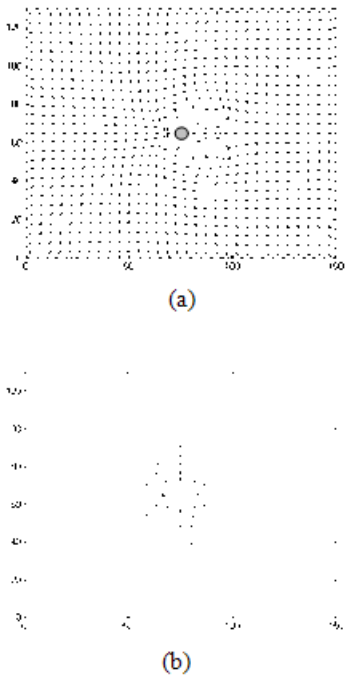

Figure 9. (a) Uni-vector field. (b) Repulsive force field associated with an obstacle.

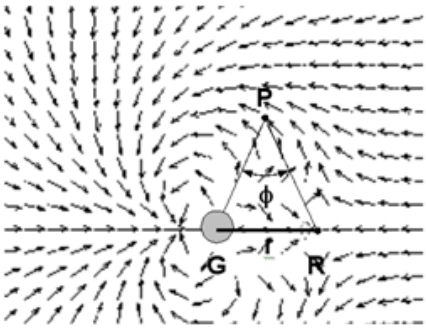

Figure 10. Computation of the Direction of the Uni-Vector Field

Figure 10 shows the relevant points used to compute for the direction of the uni-vector field. At any point, the direction is given by

$$\theta(\mathrm{P}) = \angle\mathrm{PG} - n\phi \tag{12}$$

where P is the location of the robot, G is the location of the ball, R is a point in front of the ball and ϕ is given by

$$\phi = \angle\mathrm{PR} - \angle\mathrm{PG} \tag{13}$$

The location of R sets the direction of the uni-vector field while the variables r and n determine its shape.

A repulsive field is associated to each obstacle, as shown in Figure 9(b). This force field tends to push the robot away from the obstacle. The repulsive field has no effect on a robot that is far from the obstacle. The magnitude of the obstacle field is given by

$$F_o = \frac{1}{(aD_{ro})^{\eta}} \tag{14}$$

where D_{ro} is the distance of the robot from the obstacle. The values of a and η are chosen experimentally.

3.2. The Line-Circle Algorithm

In the line-circle algorithm, the robot moves from the origin to its destination by traversing a path made up of alternating straight segments and circular arcs [5]. The robot follows a fixed pre-defined path that is computed before the robot moves.

Figures 11 and 12 show the sample paths generated using the line-circle algorithm.

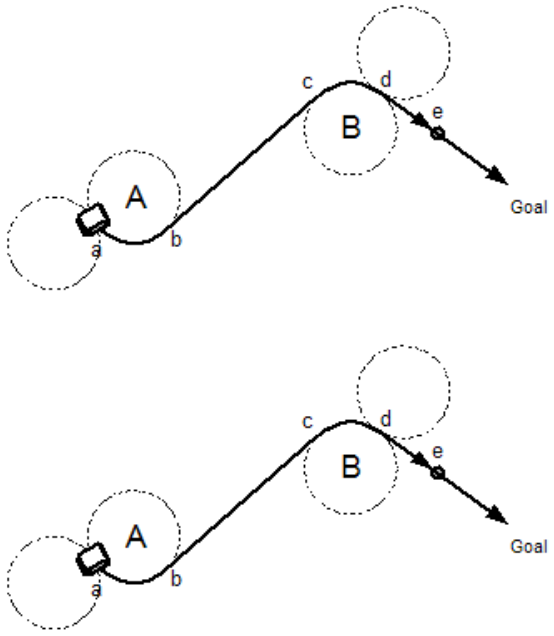

Figure 11. Generated Paths Using the Line-Circle Algorithm

3.3. Comparison of the Path Planning Algorithms

The path planners were compared by using each algorithm to steer the robot in 17 scenarios or test cases. The objective of the robot is to proceed to the location of the ball and kick it towards the center of the target goal while avoiding the other robots (obstacles). In the test

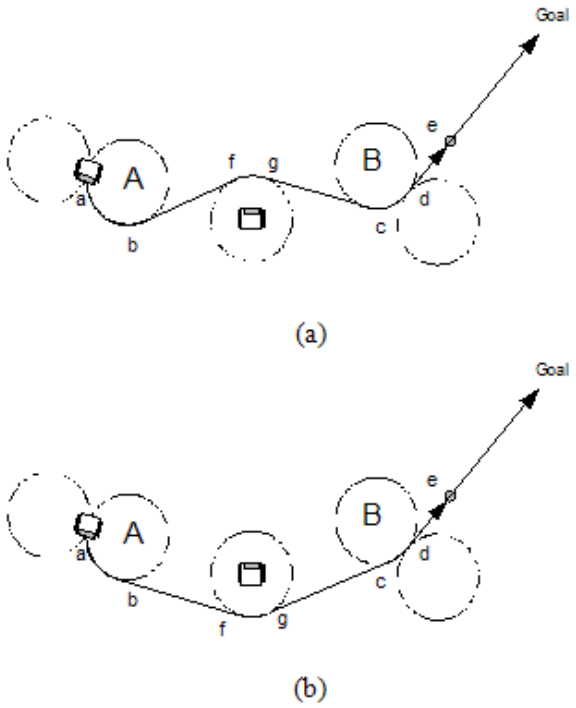

Figure 12. Paths Generated by the Line-Circle Algorithm with an Obstacle Present

cases, the robot and ball have different initial positions and at most three obstacles may be present. Table I describes the test cases.

Test case number	*Number of stationary obstacles*	*Number of moving obstacles*
1 to 8	0	0
9 and 10	1	0
11	2	0
12	3	0
13 to16	0	1
17	1	1

Table I. Summary of Test Cases

Test cases 1 to 8 evaluate the shooting skills of the robot; the robot and ball are placed in different initial locations and no obstacles are present. Test cases 9 to 12 test both the shooting skills and avoidance of stationary obstacles. Test cases 13 to 17 test both the shooting skills and avoidance of moving and stationary obstacles.

Figure 13 shows some of the test cases used (cases number 3, 8, 9 and 11). The black circle

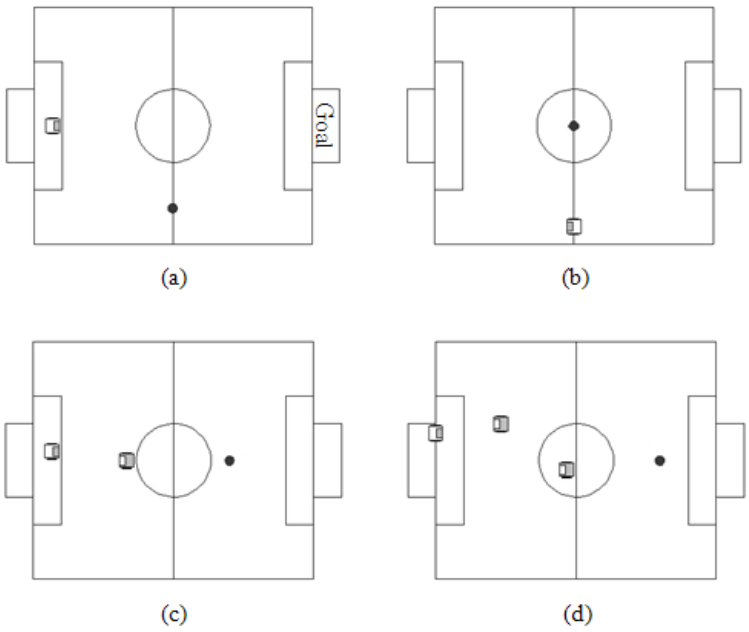

Figure 13. Example of Test Cases Used to Evaluate and Compare the Path-Planning Algorithms

depicts the ball; the light-colored robot is controlled by the path-planning algorithm while the dark-colored robots are stationary obstacles. The objective of the robot is to proceed towards the ball and kick it towards the center of the goal at the right side of the field. The goal is the rectangular area labeled in Figure 13(a).

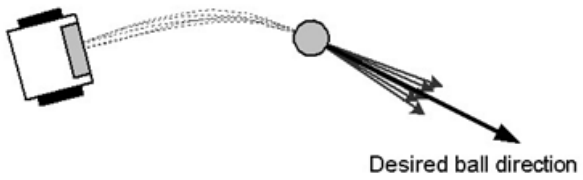

Figure 14. The noise added by the simulator causes the robot to traverse different paths during each execution.

A robot soccer simulator was used to evaluate the path-planning algorithms. The simulator was used to execute each test case using each path planner 2000 times. Each execution would yield a different result because noise was added to the actual coordinates of the robots and ball. That is, the robot would traverse slightly different paths and the ball would be kicked to directions different from the desired direction, as shown in Figure 14. A path planning algorithm is effective if the average direction of the ball after impact is close to the desired ball direction. Furthermore, the standard deviation of the ball direction should be small; implying that the range of direction to which the ball is kicked is small.

The following information were gathered after each set of simulations: (1) number of successful goal attempts, (2) average direction of the ball after impact, (3) standard deviation of the direction of the ball, and (4) number of collisions that occurred. The first three are measures of the accuracy of the path-planner. A goal attempt is successful if the kicked ball enters the rectangular goal area. The number of successful goals is counted because scoring goals is the ultimate objective in a soccer game. The average angle and its standard deviation are measures of the consistency of the robot in kicking the ball towards a desired direction. The path planner is consistent if the average direction is close to the desired direction and the standard deviation of the direction is small. The number of collisions is counted because it is a measure of the effectiveness of the obstacle avoidance scheme.

4. DISCUSSION OF RESULTS

The discussion of experimental results is divided into two parts. The first part describes the effectiveness of the Bezier curvature as a path planning algorithm and the second part compares its performance with the uni-vector and line-circle algorithms.

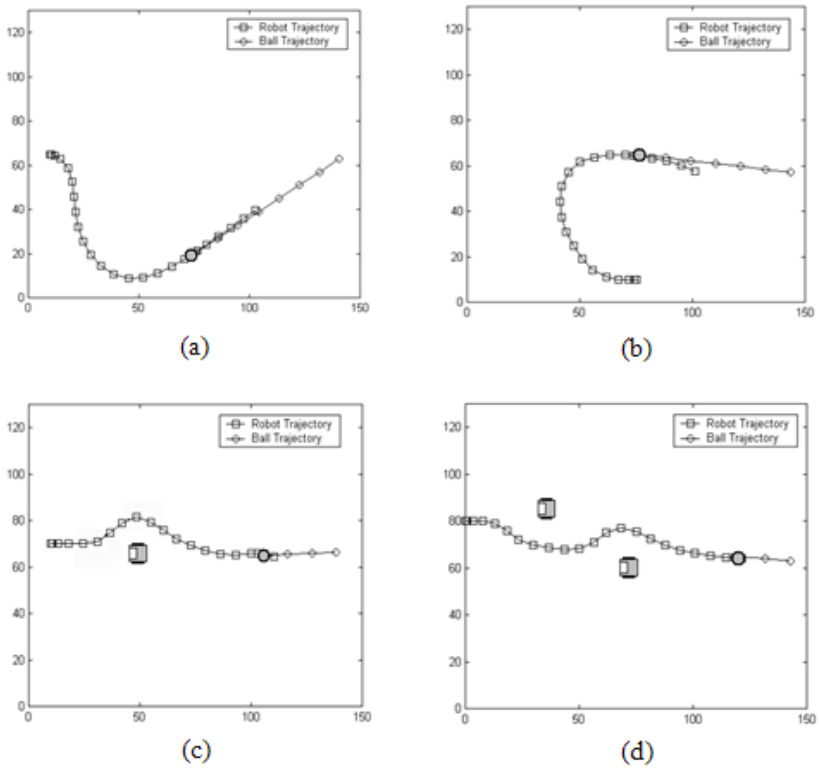

Figure 15. Robot and Ball Trajectories Using the Bezier Curvature Algorithm

4.1. Results of the Bezier Curvature Algorithm

The Bezier curvature algorithm was used to control the robot in all test cases. For the cases shown in Figure 13, a sample trajectory of the robot is shown in Figure 15. A set of coordinate axes is superimposed to the soccer field. The origin of the axes coincides with the lower left corner of the field. The field is 150 units long and 130 units wide, and the center of the target goal is at the coordinates (150,65).

In all cases, the robot followed a smooth trajectory as it moved from its original position towards the ball. The robot was able to achieve the correct orientation before it hits the ball; therefore, the ball was sent off to the desired direction after the impact. In the last two examples, the robot was able to avoid the obstacles. These sample results affirm that the Bezier curvature algorithm is an effective path-planning algorithm for soccer robots.

4.2. Comparison of the Path Planning Algorithms

The succeeding tables and figures show the results of the simulations. The tables show the goal success rate, accuracy and effectiveness of collision avoidance schemes of the path-planning algorithms in all test cases.

Tables II to IV correspond to test cases that evaluate the shooting accuracy of the robot (cases 1 to 8). The tables show that the line-circle algorithm has the best success rate and shooting accuracy and consistency. This result was expected because in this algorithm, the robot is made to traverse a straight line (segment d-e in Figures 11 and 12) before it hits the ball. Unlike in the bezier curvature and uni-vector algorithms, the robot may have a curved trajectory at the moment of impact. The curved trajectory of the robot caused the ball to be sent off to a wider range of direction after collision.

Except test cases 6 and 7, the performance of the bezier curvature algorithm is comparable to the performance of the line-circle algorithm. The goal success rates of the two algorithms are close to each other. The average error and standard direction of ball direction are small, and some results using the Bezier curvature algorithm are better than that of the line-circle algorithm.

The Bezier curvature algorithm performed very well because the two control points placed behind the ball were effective in guiding the robot towards the region behind the ball before impact. As shown in Figure 15, the robot is almost following a straight path before it hits the ball.

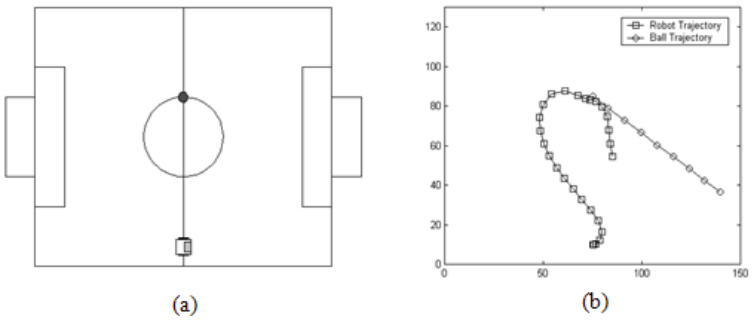

Figure 16. Test Case 7 and a Sample Robot and Ball Trajectory

Test Case	**Goal Percentage (%)**		
	Uni-Vector	Line-Circle	*Bezier Curvature*
1	96.80	100.00	*99.95*
2	90.15	96.15	*99.70*
3	48.35	99.70	*99.45*
4	48.35	99.70	*99.60*
5	49.00	99.90	*99.15*
6	53.35	99.50	*47.65*
7	55.55	99.60	*60.15*
8	62.10	99.65	*98.90*

Table II. Percentage of Successful Goal Attempts for Test Cases 1 to 8

Test Case	**Average Error (in degrees)**		
	Uni-Vector	Line-Circle	*Bezier Curvature*
1	-0.12	-0.01	*-0.03*
2	0.75	1.98	*-2.41*
3	-7.84	3.30	*-1.56*
4	-7.98	2.88	*-1.82*
5	-7.31	2.68	*-1.71*
6	3.70	-2.60	*-13.85*
7	3.88	-2.43	*-11.97*
8	1.83	-1.97	*1.19*

Table III. Average Error of Ball Direction After Impact for Test Cases 1 to 8

Test Case	**Standard Deviation of Ball Direction (in degrees)**		
	Uni-Vector	Line-Circle	*Bezier Curvature*
1	7.24	0.66	*1.40*
2	12.08	6.96	*3.01*
3	20.95	2.12	*3.69*
4	21.93	2.50	*3.18*
5	20.15	1.70	*3.40*
6	22.07	2.75	*25.24*
7	21.75	2.43	*15.57*
8	22.81	2.55	*5.20*

Table IV. Standard Deviation of Ball Direction After Impact for Test Cases 1 to 8

Test cases 6 and 7 show the inadequacy of the Bezier curvature algorithm. In these test cases, the robot is coming from a region that is in front of the ball. The placement of control points was not efficient in guiding the robot from a region in front of the ball to the region behind the ball. Figure 16 shows test case 7 and a sample robot and ball trajectory. Although the robot reached the region behind the ball, it was traversing a curved path at the moment of impact. The figure shows that the ball missed the goal.

Tables II to IV show that although the Bezier curvature algorithm does not perform as well as the line-circle algorithm, it is better compared to the uni-vector algorithm. The uni-vector algorithm yielded the lowest goal percentage and highest average error and standard deviation.

Test Case	**Goal Percentage (%)**		
	Uni-Vector	Line-Circle	*Bezier Curvature*
9	33.35	85.25	*95.85*
10	79.95	97.65	*100.00*
11	25.75	98.50	*99.80*
12	NA	98.95	*91.95*

Table V. Percentage of Successful Goal Attempts for Test Cases 9 to 12

Test Case	**Average Error (in degrees)**		
	Uni-Vector	Line-Circle	*Bezier Curvature*
9	-13.69	-5.19	*-1.96*
10	-19.47	0.88	*1.21*
11	39.33	-2.80	*-1.24*
12	NA	3.38	*-1.93*

Table VI. Average Error of Ball Direction After Impact for Test Cases 9 to 12

Test Case	**Standard Deviation of Ball Direction (in degrees)**		
	Uni-Vector	Line-Circle	*Bezier Curvature*
9	36.58	12.24	*6.60*
10	22.62	7.87	*1.96*
11	27.48	10.97	*3.53*
12	NA	3.35	*2.46*

Table VII. Standard Deviation of Ball Direction After Impact for Test Cases 9 to 12

Test Case	**Percentage of Simulations with Collisions (%)**		
	Uni-Vector	Line-Circle	*Bezier Curvature*
9	12.40	0.00	*2.10*
10	0.00	0.00	*0*
11	0.00	0.00	*0*
12	100	0.00	*7.95*

Table VIII. Percentage of Collision for Test Cases 9 to 12

Tables V to VIII show the results for test cases that have stationary obstacles (cases 9 to 12). Tables V to VII show the percentage successful goal attempts and the shooting accuracy, while Table VIII shows the percentage of attempts where collisions with obstacles occur.

Except in test case 12, the Bezier curvature algorithm is best in terms of shooting percentage. Furthermore, it is most accurate because it has the lowest ball direction deviation and error.

The line-circle algorithm, however, is best in collision avoidance. This result was expected because the robot was made to traverse a circular arc with radius greater that the dimensions of the obstacle, as shown in Figure 12.

In test case 12, the line-circle algorithm performed better than the Bezier curvature algorithm because it did not collide with an obstacle. In 2000 attempts, a goal was made 1979 times giving it a success rate of 98.95%. For the Bezier curvature algorithm, 1839 goals were made and 159 collisions occurred. These translate to 91.95% goal and 7.95% collision percentages. However, the ratio between the number of successful attempts and the number of times the robot was able to avoid the obstacle is $1839 \div (2000 - 159) \times 100\% = 99.89\%$. This means that whenever the robot was able to avoid all obstacles, it has an almost perfect chance of making a successful goal. Figure 17 shows test case 12 and a sample path using the Bezier curvature algorithm.

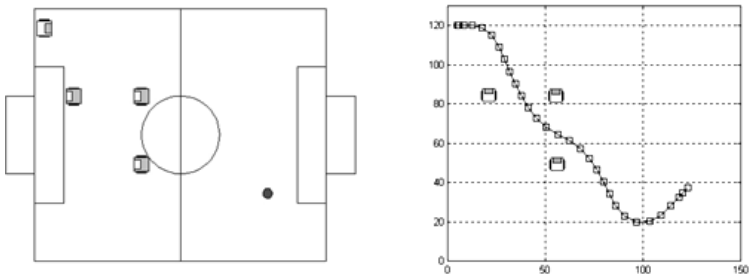

Figure 17. Test Case 12 and Sample Robot and Ball Trajectories Using the Bezier Curvature Algorithm

Test Case	**Goal Percentage (%)**	
	Uni-Vector	*Bezier Curvature*
13	85.55	*99.95*
14	48.40	*55.00*
15	33.45	*91.65*
16	40.15	*89.10*
17	41.30	*73.45*

Table IX. Percentage of Successful Goal Attempts for Test Cases 13 to 17

Test Case	**Average Error (in degrees)**	
	Uni-Vector	*Bezier Curvature*
13	-12.00	*-0.75*
14	1.62	*1.76*
15	-35.00	*0.10*
16	-33.61	*-1.64*
17	-38.50	*-1.48*

Table X. Average Error of Ball Direction After Impact for Test Cases 13 to 17

Test Case	**Standard Deviation of Ball Direction (in degrees)**	
	Uni-Vector	*Bezier Curvature*
13	21.46	*1.85*
14	34.34	*2.84*
15	28.57	*16.17*
16	19.72	*9.76*
17	25.19	*5.40*

Table XI. Standard Deviation of Ball Direction After Impact for Test Cases 13 to 17

Test Case	**Percentage of Simulations with Collisions (%)**	
	Uni-Vector	*Bezier Curvature*
13	0.00	*0.00*
14	28.35	*44.95*
15	11.35	*5.70*
16	19.20	*8.65*
17	26.35	*26.35*

Table XII. Percentage of Collision for Test Cases 13 to 17

Also note that no data has been collected for the uni-vector algorithm because the robot collided with an obstacle in all attempts.

Tables IX to XII show the results for test cases that have moving and stationary obstacles (cases 13 to 17). These tests were not performed in the line-circle algorithm because its implementation in this study cannot handle moving obstacles.

The results show that the Bezier curvature algorithm performs better than the uni-vector algorithm. In all cases, it yielded higher goal success rate and smaller error and deviation. In case 14, however, more collisions occurred in the bezier curvature algorithm.

An explanation to the large standard deviation of ball direction for the uni-vector algorithm is the jittery motion of the robot as it travels towards the ball. Figure 18 shows the variation of the angular velocity of both wheels while the robot is executing test case 1. This is the simplest test case; the robot should only traverse a straight path towards the ball. In Figures 18(a) and 18(b), v_L and v_R are the left and right wheel velocities. The third plot is the distance of the robot from the ball. The lowest point of this plot corresponds to the moment the robot touches the ball.

The plots show that the wheel velocities when using the Bezier curvature algorithm are more stable than when using the uni-vector algorithm. This means that the robot is able to maintain a smooth trajectory as it travels towards the ball. Furthermore, when using the bezier curvature algorithm, the wheel velocities are equal just before the robot reaches the ball. The robot is moving in a straight line immediately before impact so the ball is sent of to a narrow range of direction.

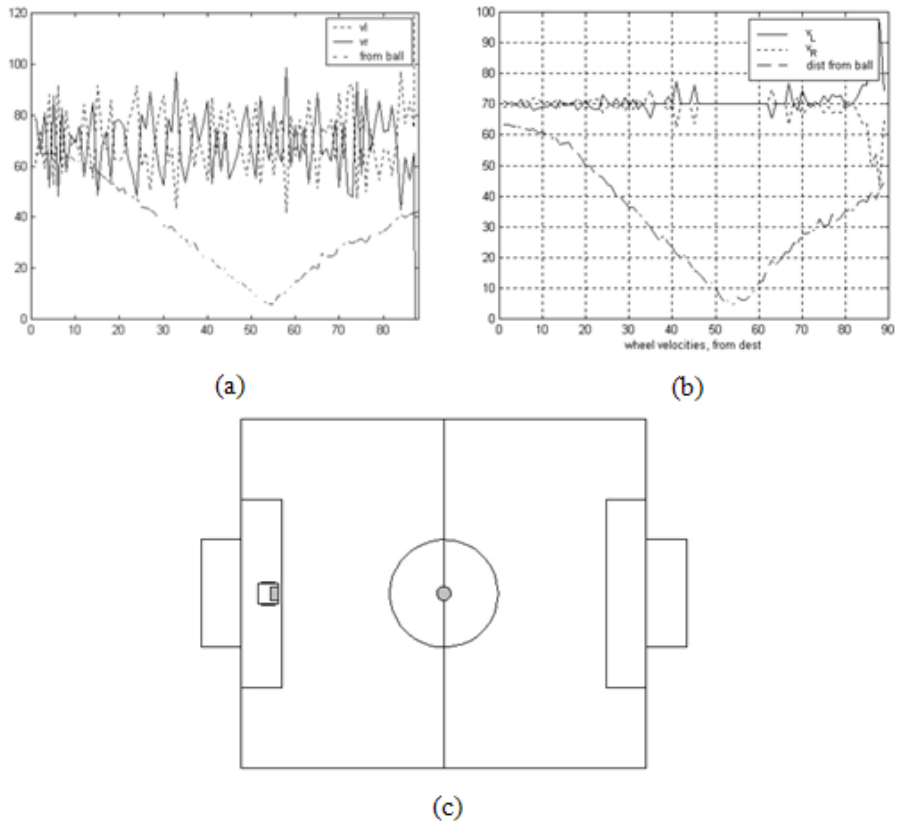

Figure 18. Variation of wheel velocities when using the (a) uni-vector algorithm and (b) bezier curvature algorithm for test case 1. (c) Robot and ball placement for test case 1.

5. CONCLUSIONS AND RECOMMENDATIONS

The results confirmed that the Bezier curvature algorithm is an effective path planning algorithm. The fact that the robot was able to accomplish the tasks verified that the principle behind the algorithm is valid. In particular, it was shown that:

- the repeated path modification approach using Bezier polynomials is valid;
- the numerical calculation of the curvature from the parametric equation describing the curve is valid; and
- the adopted method of selecting the control points for ball shooting and collision avoidance is valid.

The results also confirmed that the Bezier curvature algorithm performs equally with, or if not, exceed the performance of existing path planning algorithms. It was shown in the results that the ball direction is approximate to the desired ball direction and the ball is kicked in this direction at a very high consistency. It is recommended that the Bezier curvature path planning algorithm be implemented in real robot soccer systems.

The results of test cases 6 and 7 uncovered the deficiency of the suggested method of selecting of control points for ball shooting. It is therefore recommended that the method of selecting of the control points be improved so that the algorithm could steer the robot around the ball if the robot is coming from the region in front of the ball.

It was shown that the numerical computation of the curvature from a parametric equation is valid, therefore, this method may be also applied to other parametric equations.

The study also showed the excellent performance of the line-circle path-planning algorithm. However, it was also discovered that this algorithm is difficult to implement. It is recommended that the bezier curvature algorithm be compared to line-circle algorithm in terms of handling moving obstacles.

REFERENCES

1. Ahrikencheikh, Cherif and Seireg, Ali. **Optimized-Motion Planning, Theory and Implementation**. Copyright by John Wiley & Sons. 1994.
2. http://astronomy.swin.edu.au/pbourke/curves/bezier/
3. Leithold, Louis. **The Calculus with Analytic Geometry, 6th ed.** New York: Harper & Row. 1990.
4. Kim, K.-C.; Kim, D.-H.; Kim, Y.-J.; Kim, J.-H. and Vadakkepat, P. "'Uni-vector Field Based Path Planning and Role Selection Mechanism for Soccer Robots."' In FIRA Robot World Cup France '98 Proceedings, pp. 57-65. Edited by Stonier, Russel and Kim, Jong-Hwan. 1998.
5. Kim, Sung Ho; Choi, Jong Suk and Kim, Byung Kook. "'Development of BEST Nano-robot Soccer Team for FIRA '98."' In FIRA Robot World Cup France '98 Proceedings, pp. 7-16. Edited by Stonier, Russel and Kim, Jong-Hwan. 1998.
6. Fernandez, F. P. "'Robot Soccer Simulator Manual."' University of the Philippines, Diliman. August 2000.

