FAST TRAJECTORY GENERATION USING BEZIER CURVES

Melissa Fixter [1]

[1] School of Computer Science and Software Engineering,
The University of Western Australia

ABSTRACT: Many real world applications require the coexistence of robots with humans and machinery, including other robots. The unpredictable nature of such environments necessitates that path planning and obstacle avoidance be performed in real-time. This paper presents Bezier curves as the basis of a new method for the fast generation of a continuous, time-parameterised trajectory without the need for calculating complex configuration spaces. A Bezier curve of degree n is specified by $n + 1$ control points that have weighting functions associated with them. The curve passes through the first and last control points and is pulled towards the intermediate control points. By assigning the first and last control points to the initial and goal end effector positions, and positioning the intermediate control points relative to the location of obstacles within the robot's workspace, it is possible to produce rapidly a collision-free trajectory that curves around obstacles. This technique thus offers a viable solution to path planning in dynamic and unpredictable environments, as illustrated in this paper with a multi-link robot manipulator.

INTRODUCTION

Path planning is the generation of a complete, time parameterised trajectory of a robot, based on a snapshot of its workspace at a given time. It ensures a robot or manipulator will reach the required position and orientation at a particular time. The main shortcoming of path planning algorithms is their high computational complexity; such an overhead cannot be afforded in systems requiring real-time generation of a collision-free path.

Fundamental to computer graphics, a Bezier curve is a curve that results from the weighted sum of a number of control points. The shape of this curve is easily deformed by modifying the relative locations of the control points.

Operating purely in the workspace of the robot, my implementation of Bezier curves as a fast off-line path planner results in a rapidly generated collision-free trajectory in many situations. In the event that no trajectory could be produced via the Bezier curve method, traditional global path planning methods can be employed.

BACKGROUND

The one essential requirement of a robot trajectory is that it be collision free. In order to satisfy this, a map of the obstacles within the environment must be generated. In terms of the workspace, an obstacle is simply a three-dimensional 'thing' that must be avoided; with respect to the environmental map however, it is every position and orientation of the robot such that *any* part of the robot is in contact with *any* part of the obstacle. For a robot, a given position and orientation is known as a *configuration*. It is a vector whose dimensionality is equal to the number of degrees of freedom of the robot, where each component is a measure of the state of the respective joint with regard to some zero position. The environmental map, therefore, is also of the same dimension as a configuration, and is known as the *configuration space*, or *C-space*, of the robot. Any part of the configuration space not containing an obstacle is known as *free space*.

Once C-space has been generated, it remains to find a path entirely in free space that connects the initial configuration to the goal configuration. While selecting a trajectory is simple in theory, it is far more complicated in practice. There are infinitely many paths between any two configurations, and for a path to be free, every configuration along the path must be a free configuration. Then there is the additional complexity of choosing the path that best satisfies the goal metrics. A number of methods have been used for the generation of a trajectory, including visibility graphs, Voronoi diagrams, potential

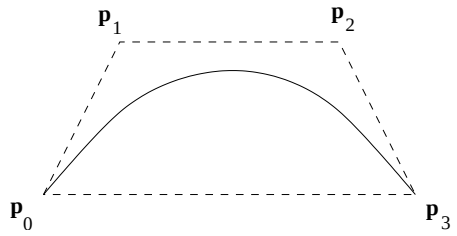

Figure 1: A 3rd degree Bezier curve generated from control points $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$. The characteristic polygon of the curve is shown as a dashed line.

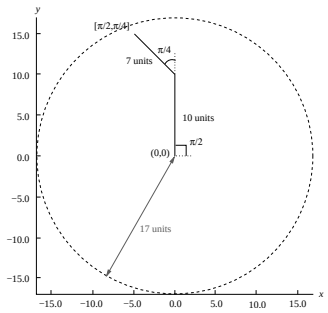

Figure 2: Robot in park position. The first and second joints are set at $\frac{\pi}{2}$ and $\frac{\pi}{4}$ radians respectively.

fields and cell decomposition [Lat91]. All this comes at a very high computational cost; the generation of configuration space alone can take in the order of minutes, or in the best case scenario, seconds, even for relatively simple robots. And increasing the number of degrees of freedom of the robot results in exponential time increases!

In dynamic environments, where the motion of obstacles is not known a priori, global path planning algorithms cannot provide a trajectory that is guaranteed to remain collision-free during execution. Additionally, the high computational complexity of such algorithms renders them unsuitable for real-time replanning. Identification of this deficiency has resulted in the generation of algorithms that integrate the planning and execution of a robot trajectory. Brock and Khatib's Elastic Strips framework [BK00] is one such method. An initial collision-free trajectory is deformed in real-time in response to repulsive forces exerted by obstacles within the workspace. Its success as a real-time execution algorithm, however, hinges on the ability to generate the initial trajectory very quickly. As such, I propose the use of Bezier curves as a fast alternative to configuration space methods.

Bezier curves have been considered as a means for satisfying Cartesian curve descriptions [KOS01]. They are more accurately described as a method of trajectory design rather than trajectory fitting, as the resultant trajectory exhibits desired curve characteristics without fitting a curve exactly.

Developed independently by de Casteljau and Bezier in 1959 and 1962 respectively, Bezier curves were the result of demand by the automobile industry to develop more modern curved shapes for cars.

A Bezier curve of degree n is specified by $n+1$ control points, $\mathbf{p}_0$, $\mathbf{p}_1$, $\ldots$, $\mathbf{p}_n$, that have weighting functions associated with them. The curve passes through the first and last control points, and is pulled towards the intermediate control points according to the Bernstein-Bezier equation

$$\mathbf{P}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{p}_i,$$

where t is a parameter that varies from 0 to 1 along the curve. The *characteristic polygon* of the Bezier curve is formed by joining the control points in order, $\mathbf{p}_0$, $\mathbf{p}_1$, $\ldots$, $\mathbf{p}_n$, $\mathbf{p}_0$, as shown in Figure 1.

Lamiraux and Kavarki [LK99] used Bezier curves successfully in modelling the deformation of elastic plates grasped by opposite edges, and Chung and Hahn [CH99] used them in the animation of humans walking over varying terrain in virtual environments. In the latter, Bezier curves were used to plan the trajectory of the pelvis and swing foot during the gait cycle. One of the motivations for this was the ease and low computational requirements with which different walks can be animated simply by modifying the control parameters.

METHODOLOGY

My model of a multi-link robot manipulator consists of a two degree of freedom planar robot within a two dimensional environment, modelled in Matlab. The robot manipulator has a fixed base located at the origin, $x = 0$ and $y = 0$, and limb lengths of 10 and 7 units respectively. Positive x direction is to the

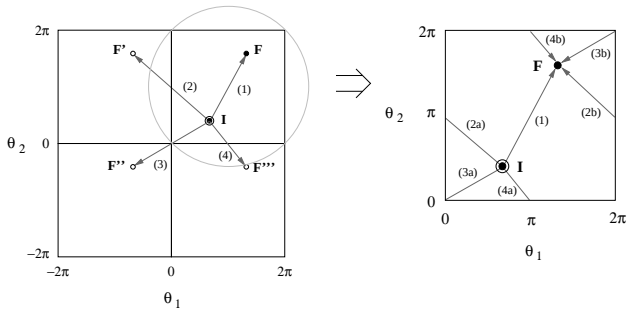

Figure 3: The four potential paths that result from an initial, $\mathbf{I}$, and goal, $\mathbf{F}$, configuration. This is an artifact of the toroidal shape of C-space.

right, and positive y direction is up. The 'park', or initial, position of the robot is fixed at the configuration $[\frac{\pi}{2}, \frac{\pi}{4}]$. Figure 2 shows the robot in park position.

The rationale behind using Bezier curves was that given an initial and final end effector position and some number of obstacles within the robot's workspace, control points could be positioned such that the resultant trajectory bends around the obstacles towards the origin. It therefore made sense to position control points relative to the location of obstacles within the workspace.

The first and last control points are defined to be the initial and final end effector positions, $\mathbf{A}$ and $\mathbf{B}$ respectively. This ensures that the trajectory passes through both points. The intermediate control points are positioned relative to obstacles within the workspace, and the origin, $\mathbf{O}$. This requires two perpendicular unit vectors, $\mathbf{u}$ and $\mathbf{v}$. The vector $\mathbf{u}$ is defined to be parallel to the vector from initial to final end effector positions. The second unit vector, $\mathbf{v}$, is perpendicular to $\mathbf{u}$ and is oriented toward the origin such that $\mathbf{v} \bullet \mathbf{AO} \leq 0$ and $\mathbf{v} \bullet \mathbf{BO} \leq 0$.

An obstacle reduction step limits the number of obstacles within the robot's workspace to at most four, each of which would result in a collision with the robot if the robot were to travel along a straight line path within C-space from initial to goal configurations. The control points are then positioned relative to each obstacle's centre-of-mass, adjusted by $k(\mathbf{u} + \mathbf{v})$, for k an optimally determined scalar [Fix03]. A Bezier curve is then generated from these control points within the robot's workspace. Points along this curve, corresponding to the robot's end-effector position, are then extracted at intervals according to the desired trajectory resolution. The robot's configuration at each point is then calculated via inverse kinematics.

To determine the suitability of my Bezier curve method, a simplistic global path planner was employed for timing comparisons. The results of this algorithm were by no means optimal, nor was it guaranteed that a path would be found if one exists. Successful invocation of the configuration space method returned one of eight straight line paths in configuration space [Fix03]. The goal end-effector position corresponded to two robot configurations: the 'elbow up' and 'elbow down' solutions. The elbow up (elbow down) solution occurs when the second joint angle is less than or equal to (greater than or equal to) π radians. For each pair of initial and goal configurations (initial and elbow up or initial and elbow down) there existed four straight line paths, as shown in Figure 3.

Based on the current implementation, I propose that Bezier curves enable the generation of a trajectory faster than the global (configuration space) method. In order to show this, both methods were employed on a number of scenarios and the time taken to generate the trajectory was recorded. The following tests were all conducted on a Pentium IV 1.7GHz machine with 256MB RAM, running the student version of Matlab under Windows XP. For each scenario, the test was repeated five times, with the reported duration being the average of the five results. This was repeated on trajectory resolutions of 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 18, and 20 degrees.

To demonstrate that the relative performance of the methods was independent of the environment, variants of two scenarios were tested. For all tests, the initial and final configurations were fixed at $[\frac{\pi}{2}, \frac{\pi}{4}]$ and $[\frac{3\pi}{4}, \frac{\pi}{2}]$ radians respectively.

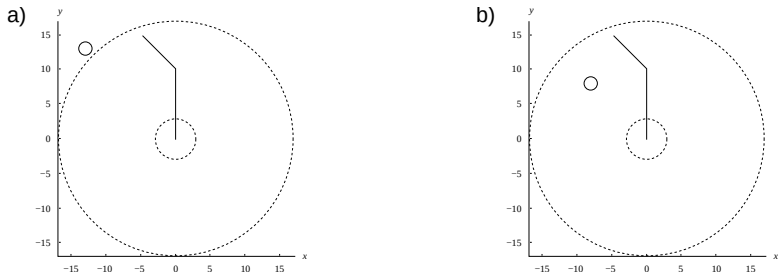

Figure 4: The geometry of the workspace for two tests consisting of a single circular obstacle of radius 1 unit. a) The obstacle centred at $(-13, 13)$. b) The obstacle centred at $(-8, 8)$.

- **Single obstacle**. The model was employed twice in an environment containing a single circular obstacle of radius 1 unit. The centre of the obstacle was positioned at (x, y) = $(-13, 13)$ and $(-8, 8)$ in the first and second tests respectively.

- **Multiple obstacles**. Four circular obstacles were evenly positioned around the workspace, with centres at (x, y) = $(-12, 12)$, $(12, 12)$, $(12, -12)$ and $(-12, -12)$. The models were both invoked on two different scenarios, firstly with all obstacles having a small radius of 0.5 units, then with all obstacles having a large radius of 2 units.

RESULTS

Single obstacle workspace. Figure 4 depicts the robot's environment during the two tests. The results of timings given an obstacle centred at $(-13, 13)$ and $(-8, 8)$ are shown in Table 1a and 1b respectively.

Table 1: Durations for generating a collision-free trajectory using the Bezier curve and global (configuration space) methods with respect to trajectory resolution. The workspace contained one medium sized circular obstacle of radius 1 unit located at a) $x = -13$ and $y = 13$, and b) $x = -8$ and $y = 8$.

a)

Resolution (degrees)	Bezier Timings (seconds)	Global Timings (seconds)
1	0.5062	84.8908
2	0.2312	21.2438
3	0.1564	9.4686
4	0.1312	5.3624
5	0.1064	3.4468
6	0.0936	2.4064
8	0.0782	1.3686
9	0.0626	1.0814
10	0.0656	0.8842
12	0.0624	0.6250
15	0.0408	0.4032
18	0.0468	0.2906
20	0.0406	0.2562

b)

Resolution (degrees)	Bezier Timings (seconds)	Global Timings (seconds)
1	0.7686	80.6000
2	0.4126	20.1312
3	0.2718	11.7906
4	0.2248	5.1094
5	0.1876	3.3250
6	0.1592	2.3250
8	0.1342	1.3062
9	0.1250	1.0530
10	0.1344	0.8438
12	0.1124	0.5938
15	0.1124	0.3938
18	0.1062	0.2688
20	0.1064	0.2280

The general trend of the timing results for a single obstacle workspace indicate that generating a trajectory via the Bezier curve method is considerably faster than the global method, particularly for finer resolution trajectories. It therefore appears the distance an obstacle is positioned from the origin has no bearing on the decision to first attempt to generate a trajectory using the Bezier curve method.

Multiple obstacle workspace. Having concluded that, for a single obstacle workspace, the Bezier curve method of trajectory generation produces results faster than the global method, it remains to show the effect of a multiple obstacle environment.

The tests conducted comprised evenly positioning four circular obstacles within the robot's workspace, as shown in Figure 5. To eliminate the effect of obstacle size from consideration, the tests were executed first with all obstacles having a radius of 0.5 units, then with all obstacles having a radius of 2 units. Tables 2a and 2b show the execution times of the trajectory generation methods according to trajectory resolution for the obstacles of 0.5 units and 2 units respectively.

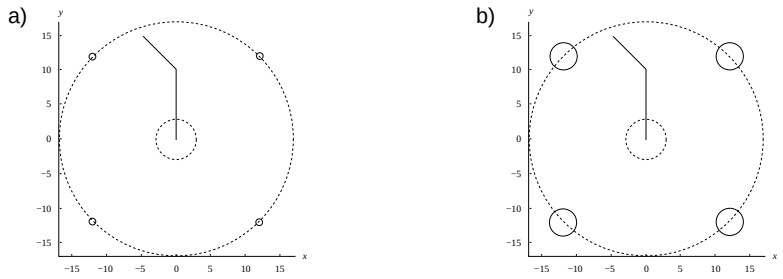

Figure 5: The geometry of the workspace for two tests consisting of four evenly spaced circular obstacles centred at (x, y) = $(-12, 12)$, $(12, 12)$, $(12, -12)$ and $(-12, -12)$. a) The obstacles have a radius of 0.5 units. b) The obstacles have a radius of 2 units.

Table 2: Durations for generating a collision-free trajectory using the Bezier curve and global (configuration space) methods with respect to trajectory resolution. The workspace contained four circular obstacles evenly spread around the workspace at (x, y) = $(-12, 12)$, $(12, 12)$, $(12, -12)$ and $(-12, -12)$, each with a radius of a) 0.5 units, and b) 2 units.

a)

Resolution (degrees)	Bezier Timings (seconds)	Global Timings (seconds)
1	1.8030	84.1220
2	0.9312	21.1532
3	0.7564	9.5280
4	0.4780	5.3408
5	0.3626	3.4406
6	0.3220	2.3970
8	0.2626	1.3718
9	0.2844	1.0784
10	0.2218	0.8906
12	0.2218	0.6250
15	0.1190	0.4062
18	0.1186	0.2906
20	0.1186	0.2344

b)

Resolution (degrees)	Bezier Timings (seconds)	Global Timings (seconds)
1	1.7970	82.2876
2	0.9220	20.7562
3	0.5938	9.2500
4	0.5312	5.2312
5	0.3564	3.3624
6	0.3092	2.3470
8	0.2376	1.3468
9	0.2094	1.0562
10	0.2250	0.8750
12	0.1656	0.6156
15	0.1438	0.4000
18	0.1438	0.2906
20	0.1312	0.2344

As with the single obstacle scenarios, the results from these tests support the use of the Bezier curve method for generating trajectories. Although not to the same magnitude, the same general trends visible in the previous section's results are also evident in the multiple obstacle tests.

RESULTS SUMMARY

As Tables 1a, 1b, 2a and 2b indicate, in the situations tested the relative performance of generating a trajectory using the Bezier curve method against the global (configuration space) method appears to be independent of the layout of the workspace.

For the scenarios examined, it has been shown that the Bezier curve method of trajectory generation is capable of producing a trajectory in a fraction of the time of the global method, regardless of the number, position and size of obstacles within the workspace. Though the magnitude of time difference reduces with increasing trajectory resolution, in all except two of the cases tested, the Bezier curve method returned a trajectory in less than half the time required by the configuration space method.

Although the time taken to execute the Bezier curve method increased for more complex workspaces, more than doubling in fine resolutions ($<$5 degrees) as the number of obstacles was increased, it still represented a vast improvement over configuration space methods.

LIMITATIONS

The comparison of timings between methods relies on the successful generation of a path by both methods. In the event that the Bezier curve generation method fails and the trajectory must be generated by a global planner, the resultant time would be the sum of the two times. As such, in situations where the Bezier curve method fails, the duration would be greater than the duration of the global method

alone. Further research must therefore be carried out to improve the success of the Bezier method.

The global method, while not necessarily returning an optimal path, would be expected to exhibit faster times than other more advanced algorithms. This is due to the simplicity of the implementation. Although the generation of a visibility graph or cell decomposition roadmap would most likely result in a more favourable trajectory, an extra computational cost would be encountered. It therefore appears more appropriate to concentrate efforts into improving the Bezier curve method rather than providing a superior back-up method.

CONCLUSION

Traditional path planning algorithms can successfully generate collision-free trajectories optimal to predefined metrics. Based on the assumption of a static environment, they may be invalidated by unforseen or moving obstacles obstructing the computed path during execution. Furthermore, they are unsuitable for real-time replanning by virtue of their high computational complexity.

The Elastic Strips framework [BK00] integrates the planning and execution of a robot trajectory. An initial collision-free trajectory is deformed in real-time in response to repulsive forces exerted by obstacles within the workspace. The ability of Elastic Strips to execute in real-time is limited by the complexity of the initial path generation. In an attempt to avoid calculating configuration space, I proposed the use of Bezier curves as an efficient off-line path planner.

It has been shown that generation of a trajectory using Bezier curves presents a viable alternative to traditional (configuration space) methods. The high computational complexity of configuration space methods is evident in the timings obtained for execution. These are in contrast to the faster times recorded for the Bezier curve method, as it was shown to perform faster than the global method for all possible resolutions up to and including 20 degrees, independent of the complexity of the workspace.

The speed with which a collision-free robot trajectory can be generated using the Bezier curve method makes it suitable for use in conjunction with Elastic Strips to enable a robot to safely navigate in a dynamic environment.

ACKNOWLEDGEMENTS

I would like to express my most sincere thanks to my supervisor, Professor Robyn Owens. Without your help, the production of this document and my thesis would not have been possible. Thankyou!

REFERENCES

[BK00] Oliver Brock and Oussama Khatib. Integrated planning and execution: Elastic strips. In *Proc. of the 2000 World Automation Congress*, 2000.

[CH99] Shih-kai Chung and James K. Hahn. Animation of human walking in virtual environments. In *Proc. Int. Conf. on Computer Animation*, pages 4–15. IEEE, 1999.

[Fix03] Melissa Fixter. Path planning and obstacle avoidance for dynamic environments, 2003. School of Computer Science and Software Engineering, The University of Western Australia, Honours thesis.

[KOS01] Peter Kovesi, Robyn Owens, and Nick Spadaccini. Cartesian space schemes, 2001. Robotics 315 Lecture Notes. The University of Western Australia.

[Lat91] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[LK99] Florent Lamiraux and Lydia E. Kavarki. Path planning for elastic plates under manipulator constraints. In *Proc. Int. Conf. on Robotics and Automation*, pages 151–156. IEEE, 1999.