Efficient Dynamic Collision Detection using Expanded Geometry Models

Boris Baginski
Robotics and Real–Time Systems Group, Fakultät für Informatik
Technische Universität München, Orleansstr. 34, 81667 Munich, Germany
e-mail: baginski@informatik.tu-muenchen.de

Abstract

Collision detection for articulated robots along possible trajectories is a basic requirement for path and motion planning systems. We present an approach using static collision tests at selected locations merely to check complete trajectories. This is done by using slightly expanded geometry models. We develop a simple formula for estimating the distance between two positions of a link within a kinematic chain. In the case of straight lines in configuration space, an efficient approximation algorithm is introduced. It calculates the next step in joint space to be close to the maximum possible length. The presented scheme allows the effort required for model–based path planning with line graphs in c–space to be reduced by at least a factor of 4 in the case of six degrees of freedom, compared to a typically used 'classical' method.

Keywords: *Dynamic Collision Detection, Geometric Modeling, Path Planning*

1 Introduction

A major aspect of an intelligent, autonomous robot system is its ability to plan its own motion without colliding with the environment. There are several paradigms to deal with motion planning in robotics. A rough differentiation can be made between model–based and non–model–based approaches. If there is no model (at least no model of the environment), the robot reacts on sensor input and motion planning is actually a controlled motion execution.

The model–based approaches can be sub-classified again, and again it is a very rough differentiation. One possibility is to develop a planning algorithm based on a suitable model. A powerful model is the configuration space (c–space), the parameter space of the robot. Each possible state or pose of the robot is a point, and a path is a curve in this space, connecting start and goal. The problem with this kind of approach is obtaining a sufficient representation of the c–space that has to be created from the information that is available. In general, this requires expensive precomputations to build up a cell decomposition (e.g. [9]) or a connectivity graph (e.g. [6]).

Another possibility is to see what kind of geometric, kinematic and dynamic information is available and to develop a path planning approach based upon it. This allows spontaneous planning when the data is available and is our paradigm for path planning [2, 1]. We want to operate a known manipulator in a changing environment, for example a manipulator on a mobile platform for flexible production (see fig. 1). Our environment model consists of two major parts:

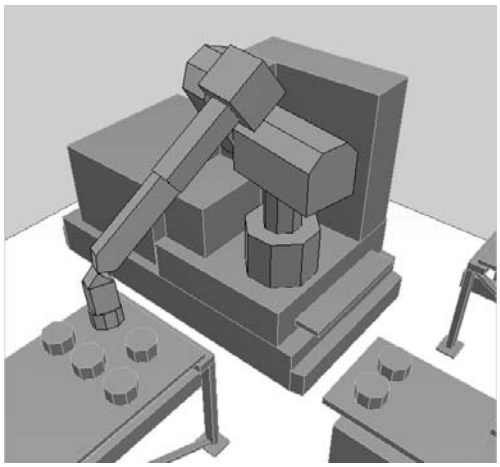

Figure 1: *An example of a mobile manipulator in our simulation environment.*

- The robot. Its geometry and its kinematic constraints are known prior to any planning. This allows us to do precomputations of any kind that simplify the planning process.
- The environment. It is not known a priori. Its geometry will be, at least partially, read by sensors, resulting in an incomplete and uncertain model.

In an industrial environment most of the data will be known from CAD, and will typically consist of complex shaped surfaces without structure that can be used to convert or group them.

In the following, we will focus on one aspect of path planning, the aspect of dynamic collision detection. Most of the path planning efforts will be spent in testing segments of candidate paths for possible collisions with the environment. The model available is not sufficiently structured to use a sophisticated distance computation algorithm efficiently, as these require an environment made up of convex polytopes [7] or other complex data structures [8, 5].

The requirements for an efficient static collision test are generally less demanding. A static collision test ascertains if a particular configuration of the robot collides or not. It is achieved efficiently with a hierarchy of increasingly simplified hull bodies [8, 4]. The output of this test is only binary, no minimal distance is computed.

A static collision test can not directly be used for dynamic collision detection. Theoretically, it requires an infinite number of tests along a trajectory to test it for collisions. It should be marked, that a lot of commercial robot simulation packages use a static collision test only. The discretization is often a parameter set by the user, resulting in a more or less random result. In the following, we expand the geometric model of the robot to be able to correctly use a collision test.

The rest of the paper is organized as follows: Section 2 introduces a 'classic' approach for the distance calculation between multiple collision tests. This approach is analyzed and an improved concept is developed. Section 3 describes this new concept in detail. Section 4 describes the dynamic collision scheme based on step length approximation. The efficiency of this scheme is improved in section 5, when multiple expanded models are used. Section 6 gives some results of our implementation. Section 7 ends the paper with concluding remarks and some words on ongoing work.

2 The Weightened 1-Norm Method

To be able to use a static collision test for dynamic collision detection, the robot geometry model is expanded. A 'protective shield' is constructed, covering the robot with a particular thickness d. This allows static collision detections in distances of $2d$ each. More precisely, a condition for correct collision detection can be put into words: *If the robot is expanded by d, no point of the robot is allowed to move further then $2d$ between two collision tests.* This idea is illustrated in fig. 2.

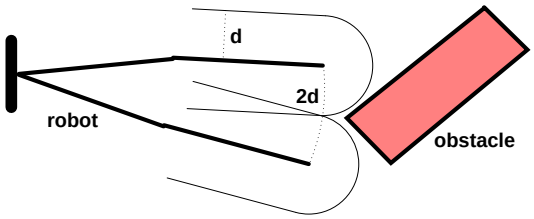

Figure 2: *An expansion of d allows steps of $2d$.*

Discretizing Algorithm

To calculate steps, the condition has to be formalized. Configurations are given as vectors of joint angles in the n–dimensional c–space, $\Phi = (\phi_1 \dots \phi_n)^T$. The relevant motion of the links takes place in the workspace. It has to be estimated how far a certain step $\Delta\Phi = \Phi_1 - \Phi_0$ in c–space moves a point on the robot.

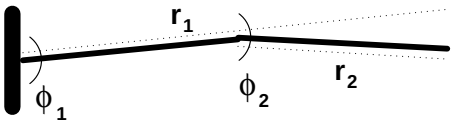

Figure 3: *An example of the maximum possible radii for a two link manipulator in 2 dimensional space. r_2 is the length of the outer link, r_1 is the length of the inner link plus the length of the outer link in its maximum extension with respect to the first joint.*

A worst–case estimation for rotational joints can be done by using the largest possible radius r_i for each joint (see fig. 3). Multiplying it with the current rotation angle ϕ_i results in the maximum arc length. In the worst case, all these rotations add up and the condition for collision test distances can be formulated as follows:

$$\sum_{i=1}^{dof} r_i \mid \Delta\phi_i \mid \leq 2d \qquad (1)$$

For arbitrary shaped robot links with arbitrary oriented axes the radii are difficult to calculate analytically. But they can be estimated by iteratively constructing swept volumes of the links. The outermost radius r_n is just the maximum of the distances of all points of the outermost link to the rotation axis. Then the swept volume (or, easier, the convex hull of the swept volume) for this link is constructed and fixed at the previous link. Now the radius r_{n-1} can be found, and the procedure is repeated for all joints.

Translational joints can be handled with this approach as well. If the joint parameter equals the distance moved by the joint, its 'radius' equals 1. This

just means, that its motion contributes linear to the worst–case maximum motion.

All radii are positive, so (1) can be seen as a weightened 1–norm in c–space. For collision detection along straight lines in c–space, the weightened 1–norm of the line can be calculated and maximum steps between tests can be done with respect to this condition.

Analysis

If we base our dynamic collision detection on (1), the collision tests executed much more frequently than necessary. This has two main reasons:

- The condition is based on a worst–case assumption. It is assumed, that the robot is stretched to its full length and that all rotational motions add up in the same direction.
- At each step, the whole robot is tested for collision. But the worst case can only occur for the outermost link, the inner links are generally tested too often. Figure 4 illustrates this problem.

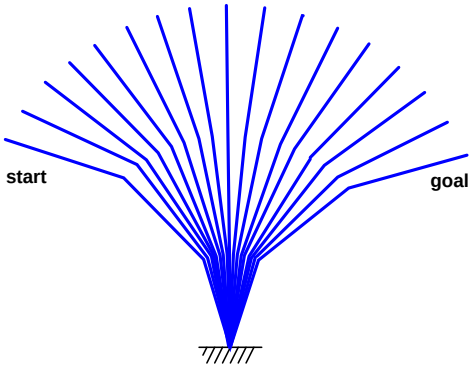

Figure 4: *Dynamic collision detection with a condition for the whole kinematic chain. The innermost link is checked much more often than necessary.*

New Approach

Based on this results, a new condition is put into words for collision detection: *No point of the robot is allowed to move further then* $2d$ *between two collision tests. But all links should be moved as close to* $2d$ *as possible for maximum efficiency.* To do this, we will look at the motion of all the links in a kinematic chain individually and test them independently at the maximum possible distance.

3 Motion of Bodies in a Kinematic Chain

A link in the kinematic chain moves within the six dimensional cartesian space along a trajectory that is defined by the motions of all previous joints. It is not obvious which 'distance' is traveled by a particular link. The motion is not straight, thus the cartesian distance is not sufficient. The maximum arc length of a point is expensive to calculate. Thus we will overestimate the distance between two link positions with a small error, but with little computational effort.

Estimation of Translational Distance

The translational distance is overestimated by the maximum distance of bounding boxes covering the link at each position. As the bounding box is convex, no point within the box can have a larger translational distance than the box. The maximum distance of all points of the bounding box is the maximum distance moved by one of its extreme points (see fig. 5).

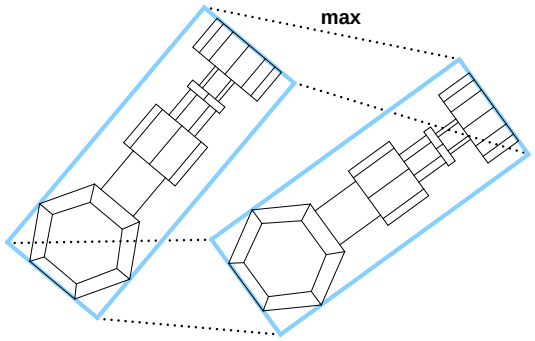

Figure 5: *Overestimating the translational distance by the distance of bounding boxes.*

To estimate the distance, eight precomputed points p_i have to be transformed in each configuration and the maximum distance between the pairs of points is chosen:

$$\mathrm{est}(\Phi_0, \Phi_1) \quad = \quad \max_{i=1,\ldots,8} \|p_i(\Phi_1) - p_i(\Phi_0)\|_2 \qquad (2)$$

Estimation of Rotational Divergence

The motion of a link is not straight, if there are rotational joints in the kinematic chain *before* the link. The maximum divergence from the straight line can be easily calculated for one rotational joint if the angle and the maximum radius are known, see fig. 6.

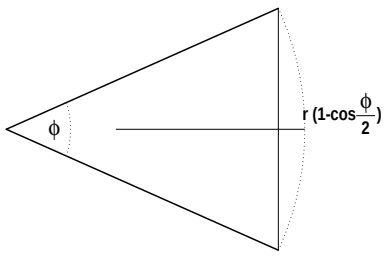

Figure 6: *The maximum divergence from the straight line for rotations is very small for small angles.*

We can overestimate the maximum divergence of link j if we sum up all divergences from the previous joints:

$$\mathrm{div}(\Delta\Phi) \;=\; \sum_{i=1}^{j} r_{i,j} \left(1 - \cos\left(\frac{\Delta\phi_i}{2}\right)\right) \qquad (3)$$

The radii $r_{i,j}$ in this equation are the maximum radii similar to the radii described in section 2, but the links $j+1,\ldots,n$ are not taken into account. This requires the precomputation of a triangular $n \times n$ matrix of weights. The divergence calculated with (3) is a worst–case estimation, as it is assumed that all divergences sum up. But the divergence is small for small angles, as the cosine stays very close to 1. The calculation is only valid for $\Delta\phi_i \in [-\pi, \pi]$, but we will generally use much smaller steps.

The distance between positions can now be carefully estimated by adding the translational estimation and the maximum rotational divergence. This is again a worst–case assumption: the rotational divergence resulted in maximum disturbance of the translational estimation.

Maximum Distance between Tests

For the collision test distance, equation (4) has to hold. No point of the robot can pass free space without being covered by a collision test.

$$\mathrm{div}(\Delta\Phi) \;+\; \mathrm{est}(\Phi_0, \Phi_1) \;\leq\; 2d \qquad (4)$$

There are several possibilities to use the above defined estimation for the distance to assure large distance between collision tests. If the trajectory is of arbitrary shape, it can be discretized and the distance is estimated between all steps. Whenever the distance to the last test is above the allowed threshold $2d$, a collision test is performed for the *previous* discretization point. This will require a lot of distance estimation. Even if this is much cheaper than a collision test, a forward computation is better.

4 Straight Motions in C–Space

For the case of straight lines, forward computation is possible, if we have already made one step $\Delta\Phi = \Phi_0 - \Phi_1$. If there were only translational joints, the new step $\Delta\hat{\Phi}_{\mathrm{opt}}$ of maximum length $2d$ would be directly proportional to the cartesian distance *dist* of the previous step:

$$\Delta\hat{\Phi}_{\mathrm{opt}} \;=\; \frac{\Delta\Phi}{\mathrm{dist}(\Delta\Phi)}\, 2d \qquad (5)$$

But there are rotations that complicate the calculation. If we assume maximum divergence for the distance estimation and maximum divergence within the new step $\Delta\Phi_{\mathrm{opt}}$, the following equation is a careful estimation:

$$\Delta\Phi_{\mathrm{opt}} \;=\; \frac{\Delta\Phi}{\mathrm{est}(\Phi_0, \Phi_1) \;+\; \mathrm{div}(\Delta\Phi)} \left(2d \;-\; \mathrm{div}(\Delta\Phi_{\mathrm{opt}})\right) \qquad (6)$$

This is difficult to calculate, because $\Delta\Phi_{\mathrm{opt}}$ is the parameter of the non-linear function *div* on the right side. But *div* is super-linear, and a two-step approximation to this optimal value can be derived:

$$\overline{\Delta\Phi_{\mathrm{new}}} \;=\; \frac{\Delta\Phi}{\mathrm{est}(\Phi_0, \Phi_1) \;+\; \mathrm{div}(\Delta\Phi)}\, 2d \qquad (7)$$

$$\Delta\Phi_{\mathrm{new}} \;=\; \overline{\Delta\Phi_{\mathrm{new}}}\, \frac{2d}{2d \;+\; \mathrm{div}(\overline{\Delta\Phi_{\mathrm{new}}})} \qquad (8)$$

This approximation results in a step $\Delta\Phi_{\mathrm{new}} < \Delta\Phi_{\mathrm{opt}}$, and $\Delta\Phi_{\mathrm{new}}$ was based on worst–case assumptions and overestimated divergences. Thus $\Delta\Phi_{\mathrm{opt}}$ is a safe step, i.e. equation (4) holds. But for relatively small angles it will be very close to the optimal step. The results obtained were in the range of 90% to 99% of $2d$.

Collision Detection Algorithm

An efficient collision detection algorithm can be developed with this approximation, as the current and the previous test positions can be used to estimate the next one. The algorithm requires Φ_{start} and Φ_{goal} as input and runs as follows (see fig. 7):

- Test the configuration Φ_{start}.
- Calculate a small linear step towards Φ_{goal}. The position is not tested, but used to approximate the first step for each link using (7), (8).
- Test the link with its next untested position closest to Φ_{start} and calculate its next step. Repeat until terminated.
- Terminate, if a collision occurs, or if the distance between the last tested position and Φ_{goal} falls below d for all links.

5 Using Multiple Expanded Models

The only parameter required for this algorithm is the expansion d. This is difficult to select, if the robot shall efficiently move through free space (large expansion, large steps) but as well shall move close to obstacles (small expansion). If we use multiple layers of models with different thicknesses, it is possible to adapt dynamically to different situations. We suggest to use a small expansion d_0 as minimal tolerance and, covering this, increasingly larger models, each with a doubled expansion, $d_{i+1} = 2 * d_i$. This leads to an effective reduction of steps if a larger model can be

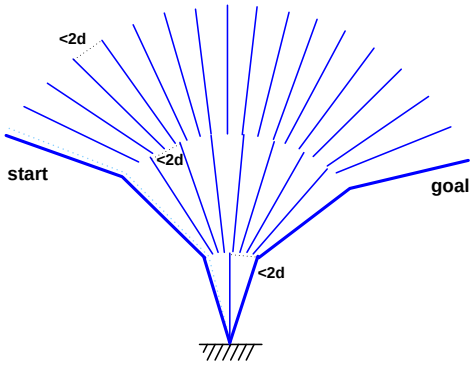

Figure 7: *By estimating individual steps for each link, the distances between all collision tests are maximized.*

used, each increase will almost half the number of required tests. The collision detection algorithm treats all links separately, so the 'protective shields' for the links can be different. Figure 8 illustrates the possible different thicknesses in one pose.

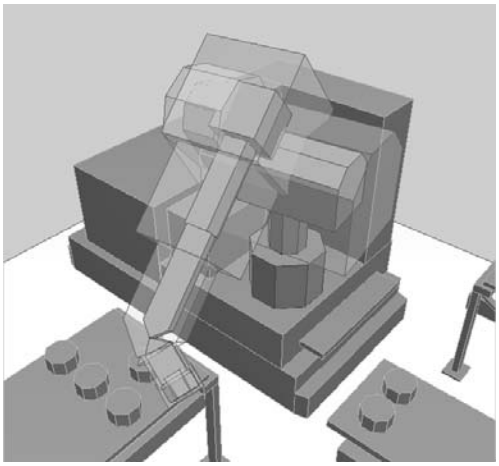

Figure 8: *An example for different model levels. The tool and the load are close to the table and can only be covered by a thin 'protective shield'. The other links have more free space, thus less collision tests are required when they move.*

Advanced Possibilities

Larger models will in general yield more collisions than smaller models, and the collision test will be more expensive. But this can possibly be compensated by the use of *simplified* models at higher levels. The 'protective shield' has to have a thickness of *at least* d_i, but a slightly larger model with a lot of details removed will be much more convenient to test. All this can be done in a preprocessing phase for the robot, without any knowledge of the environment.

Switching between the Models

We modified our algorithm to be able to handle a fixed number of layers with the thicknesses $d_i = 2^i\, d_0$. The factors can easily be integrated in the approximation formulae. We start at Φ_{start} with the smallest model (if the possible size is not known from previous computations) and try to switch to the next larger model in each step. If a collision occurs for a particular level, the model is shrunken and the test position is moved backwards until the obstacle can be passed or the smallest model collides (see fig. 9).

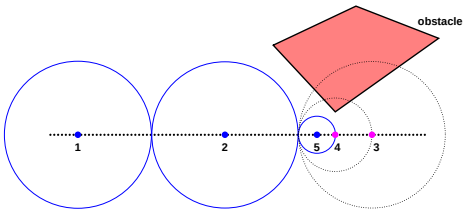

Figure 9: *The robot is illustrated as one point in this figure. If it gets too close to an obstacle, the thickness of the protective shield is halved and the test position is adjusted to ensure a collision free line segment.*

For each level, a delay value is stored that counts how many steps have to be done with the level below before the algorithm attempts to switch to that level. The delay is one at the beginning for all levels, and is doubled for the respective level if a collision occurs. This assures, that the number of unsuccessful attempts decreases if, for example, the robot passes an obstacle with constant distance (see fig. 10).

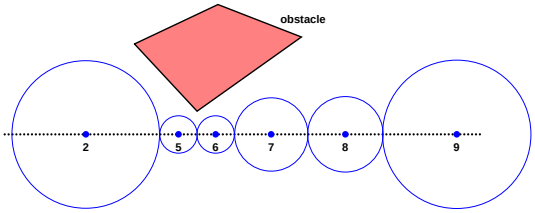

Figure 10: *If there were collisions, the attempt to switch to thicker protective layers is delayed.*

6 Results

We implemented the dynamic collision detection based on expanded models as part of our path planning and simulation system. We were not able to compare the achieved efficiency with other dynamic collision detection systems, but we were able to efficiently increase the performance with respect to the 'global' discretization described in section 2. Table 1 shows a comparison of the number of collision tests executed

robot type	work-space	1–norm	approx. Steps	multiple models
6 DOF	6 dim.	100 %	24,5 %	2,0 %
8 DOF	3 dim.	100 %	7,8 %	4,9 %
16 DOF	6 dim.	100 %	7,4 %	0,6 %

Table 1: *The relative number of collision tests performed with the different collision detection algorithms.*

to check 1,000 random straight lines. We use several manipulators from 2 to 16 DOF in environments with several obstacles. The smallest models used are expanded by $d_0 = 2.5mm$. The multiple model algorithm works with 8 models of equal complexity resulting in a maximum expansion of $32cm$. The radii of the robots' workspaces were about $100cm$ to $200cm$.

All algorithms returned the same results, i.e. the same percentage of paths were classified as colliding. The computational overhead for the step approximation is quite low, especially if the geometry is complex and the collision test gets more expensive. The reduction of necessary collision tests is enormous, especially if multiple models are used. Robots with more degrees of freedom will in general show much better performance as the discretization formula (1) results in worse steps compared to robots with few DOF.

7 Future Work and Conclusion

We have presented an efficient approach to collision detection along straight lines in c–space. Our method only requires a static collision test and does not rely on any special data structure or preprocessing of the environment representation required by other dynamic collision detection algorithms. This is achieved by using several expanded geometry models of the robot instead of calculating distances to the obstacles. The number of collision tests is close to the possible minimum.

In our opinion, any kind of geometric preprocessing on the robot can be done, as its geometry and kinematics are known well before planning. The geometry of the environment does not allow such operations, as it may be incomplete or sensor–read and often only becomes available when immediate planning is already required. Our dynamic collision detection scheme was developed under these assumptions and handles them efficiently. In its current form it can be easily integrated in all path planning systems that explore straight lines in c–space (e.g. [6, 3]).

We will continue our work to minimize the number of collision tests within our local planning algorithms [2]. Especially in combination with the local planning based on shrinking and growing geometry models [1], the use of several sets of grown models is very promising. Another important step is the development of a simple approximation scheme for arbitrary trajectories.

The simulated ('virtual') environment can supply other information besides that of the real world and it provides the opportunity to change size and shape of the 'virtual' objects to gain information. It is not necessary to limit oneself in the 'virtuality' to the features of reality. We think our work efficiently gains maximum information out of the simulated model with minimum costs.

References

[1] Boris Baginski. Local motion planning for manipulators based on shrinking and growing geometry models. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 3303–3308, Minneapolis, April 1996.

[2] Boris Baginski. The Z^3–method for fast path planning in dynamic environments. In *Proceedings of IASTED Conference Aplications of Control and Robotics*, pages 47–52, Orlando, Florida, January 1996.

[3] Martin Eldracher. Neural subgoal generation with subgoal graph: An approach. In *Proceedings of World Conference on Neural Networks WCNN '94*, pages II–142 – II–146, 1994.

[4] S. Gottschalk, M.C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. In *Proceedings of ACM Siggraph '96*, August 1996.

[5] P. Jiminez and C. Torras. Speeding up interference detection between polyhedra. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1485–1492, Minneapolis, April 1996.

[6] Lydia Kavraki and Jean-Claude Latombe. Randomized preprocessing of configuration space for fast motion planning. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 2138–2145, San Diego, California, May 1994.

[7] Ming C. Lin and John F. Canny. A fast algorithm for incremental distance calculation. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1008–1014, Sacramento, April 1991.

[8] Sean Quinlan. Efficient distance computation between non-convex objects. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 3324–3329, San Diego, California, May 1994.

[9] E. Ralli and G. Hirzinger. A global and resolution complete path planner for up to 6DOF robot manipulators. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 3295–3302, Minneapolis, April 1996.