Continuous Collision Detection for Motion Planning

Liang-Jun Zhang

Motion Planning Course Lecture
Comp 290-58,
Oct 10th, 2005, UNC
• Mainly based on F. Schwarzer, M. Saha, J. Latombe’s work

“Exact Collision Checking of Robot Paths” WAFR 2004
Motivation (1)

One tenet of PRM planning is that sampled configurations and connections can be efficiently tested for collision.
Motivation (2) – “Clear Query”

Static collision tests (for sampled configurations) are done efficiently using pre-computed bounding volumes (BV) hierarchies

[Quinlan, 94; Gottschalk, Lin, Manocha, 96]
Bounding Volume Hierarchies

- **Model Hierarchy:**
  - each node has a simple volume that bounds a set of triangles
  - children contain volumes that each bound a different portion of the parent’s triangles
  - The leaves of the hierarchy usually contain individual triangles

- **A binary bounding volume hierarchy:**
Trade-off in Choosing BV’s

Sphere  AABB  OBB  Convex Hull
Collision Test: Tree Traversal

Hierarchy of tests
Trade-off in Choosing BV’s

Sphere  AABB  OBB  Convex Hull

increasing complexity & tightness of fit

decreasing cost of (overlap tests + BV update)
But *dynamic* collision tests (for connections) using BV hierarchies are usually approximate.

- $\varepsilon$ too large $\rightarrow$ collisions are missed
- $\varepsilon$ too small $\rightarrow$ slow test of local paths
Previous Approaches to Dynamic Collision Testing

- Bounding-volume (BV) hierarchies
  - Discretization issue
Previous Approaches to Dynamic Collision Testing

Feature-tracking methods

[Lin, Canny, 91]
[Mirtich, 98] V-Clip
[Cohen, Lin, Manocha, Ponamgi, 95] I-Collide
[Basch, Guibas, Hershberger, 97] KDS

→ Geometric complexity issue with highly non-convex objects
Previous Approaches to Dynamic Collision Testing

- Swept-volume intersection
  
  [Cameron, 85] [Foisy, Hayward, 93]
Previous Approaches to Dynamic Collision Testing

Swept-volume intersection

→ Swept-volumes are expensive to compute.

→ a sufficient but not necessary condition for two moving objects
Schwarzer et al's approach

Problem:
Given two configurations, test if the straight path between them is collision-free, or not.

Ideas:

a) Relate configuration changes to path lengths in workspace

b) Use distance computation rather than pure collision checking
For any $q$ and $q'$ no robot point traces a path longer than:

$$
\lambda(q, q') = L|\delta q_1|
$$

**Ideas**

a) Relate configuration changes to path lengths in workspace

b) Use distance computation rather than pure collision checking

$$
q = (q_1) \\
qu' = (q'_1) \\
\delta q_1 = q'_1 - q_1
$$

For any $q$ and $q'$ no robot point traces a path longer than:

$$
\lambda(q, q') = L|\delta q_1|
$$
For any \( q \) and \( q' \) no robot point traces a path longer than:

\[
\lambda(q,q') = ???
\]

(These is our homework)
\[ \eta(q) = \text{Euclidean distance between robot and obstacles (or lower bound)} \]
If $\lambda(q,q') < \eta(q) + \eta(q')$ then the straight path between $q$ and $q'$ is collision-free.

$\eta(q)$ = Euclidean distance between robot and obstacles (or lower bound)

**Ideas:**

a) Relate configuration changes to path lengths in workspace
b) Use distance computation rather than pure collision checking
Ideas:

a) Relate configuration changes to path lengths in workspace
b) Use distance computation rather than pure collision checking

\[ \lambda(q, q') < \eta(q) + \eta(q') \]

\{ q'' \mid \lambda(q', q'') < \eta(q') \}

\{ q'' \mid \lambda(q, q'') < \eta(q) \}
Ideas:

a) Relate configuration changes to path lengths in workspace
b) Use distance computation rather than pure collision checking

\[ \lambda(q,q') > \eta(q) + \eta(q') \]
Adaptive Checking

If \( qq' \) fail

\( q \rightarrow \text{check } q_{\text{mid}} \text{ and } q_{\text{mid}} q' \)

\[ \lambda(q,q') > \eta(q) + \eta(q') \]

Bisection

\( \{ q'' | \lambda(q',q'') < \eta(q') \} \)

\( \{ q'' | \lambda(q,q'') < \eta(q) \} \)
Greedy Distance Computation

- Use BV hierarchy + same recursion as for pure collision checking
- But compute distance between BVs instead of testing BV overlap
- $\rightarrow$ BVs are RSSs
$\lambda(q,q')$: Traced Length Computation
$\lambda(q,q')$: Traced Length Computation

$q = (q_1,q_2,q_3)$
$q' = (q'_1,q'_2,q'_3)$
$\delta q_i = q'_i - q_i$
\( \lambda(q,q') \): Traced Length Computation

\[ \lambda(q,q') = 3|\delta q_1| + 2|\delta q_2| + |\delta q_3| \]

For any \( q \) and \( q' \) no robot point traces a path longer than:

\[ q = (q_1, q_2, q_3) \]
\[ q' = (q'_1, q'_2, q'_3) \]
\[ \delta q_i = q'_i - q_i \]
Generalization

Robot(s) and static obstacles treated as collection of rigid bodies A1, ..., An.

$\lambda_i(q,q')$: upper bound on length of curve segment traced by any point on Ai when robot system is linearly interpolated between $q$ and $q'$

\[
\begin{align*}
\lambda_1(q,q') &= L|\delta q_1| \\
\lambda_2(q,q') &= 2L|\delta q_1| + L|\delta q_2| \\
\lambda_3(q,q') &= 3L|\delta q_1| + 2L|\delta q_2| + L|\delta q_3|
\end{align*}
\]
Generalization

Robot(s) and static obstacles treated as collection of rigid bodies $A_1, \ldots, A_n$.

$\lambda_i(q,q')$: upper bound on length of curve segment traced by any point on $A_i$ when robot system is linearly interpolated between $q$ and $q'$

If $\lambda_i(q,q') + \lambda_j(q,q') < \eta_{ij}(q) + \eta_{ij}(q')$ then $A_i$ and $A_j$ do not collide between $q$ and $q'$
Generalized Bisection Method

- Each pair of bodies is checked independently of the others → priority queue Q of elements \([q_a, q_b]_{ij}\).
- Initially, Q consists of \([q, q']_{ij}\) for all pairs of bodies \(A_i\) and \(A_j\) that need to be tested.

I. Until Q is not empty do:
   1. \([q_a, q_b]_{ij} \leftarrow \text{remove-first}(Q)\)
   2. If \(\lambda_i(q_a, q_b) + \lambda_j(q_a, q_b) \geq \eta_{ij}(q_a) + \eta_{ij}(q_b)\) then
      a. \(q_{\text{mid}} \leftarrow (q_a + q_b)/2\)
      b. If \(\eta_{ij}(q_{\text{mid}}) = 0\) then return collision
      c. Else insert \([q_a, q_{\text{mid}}]_{ij}\) and \([q_{\text{mid}}, q_b]_{ij}\) into Q

II. Return no collision
Heuristic Ordering Q

Goal: Discover collision quicker if there is one.

Sort Q by decreasing values of:
\[
\left[ \lambda_i(q_a,q_b) + \lambda_j(q_a,q_b) \right] - \left[ \eta_{ij}(q_a) + \eta_{ij}(q_b) \right]
\]
Two Arms and Three Rings

Two 20-dof linkages, with 320 triangles each
Three rings with 6,300 triangles each
Robot in a Cage

Robot: 2,991 triangles
Cage: 432 triangles

Demo
Spot Welding

Robot: 2,991 triangles
Obstacles: 74,681 triangles
Comparative Experiment

**SBL:** PRM planner (single-query, bi-directional, lazy in cc) with fixed-discretization collision checker

**A-SBL:** Same planner, with new collision checker

**Experiment:**
- Run SBL 10 times on same planning problem with some resolution $\varepsilon$
- If a collision has been missed, reduce $\varepsilon$ and repeat
- If no collision has been missed, return average planning time
- Run A-SBL 10 times and return average planning time

\[ \text{SBL} \rightarrow 17 \text{ sec} \]
\[ \text{A-SBL} \rightarrow 4.8 \text{ sec} \]
Some Results

Robot: 2,502 triangles
Obstacles: 432 Triangles

SBL   $\rightarrow$ 17 sec  
A-SBL $\rightarrow$ 4.8 sec

Robot: 2,991 triangles
Obstacles: 432 Triangles

SBL   $\rightarrow$ 83 sec  
A-SBL $\rightarrow$ 44 sec

Robot: 2,991 triangles
Obstacles: 74,681 triangles

SBL   $\rightarrow$ 1.20 sec  
A-SBL $\rightarrow$ 0.81 sec

Robot: 2,502 triangles
Obstacles: 34,171 triangles

SBL   $\rightarrow$ 3.2 sec  
A-SBL $\rightarrow$ 2.1 sec

Robots: 6 $\times$ 2,991 triangles
Obstacles: 19,668 triangles

SBL   $\rightarrow$ 85 sec  
A-SBL $\rightarrow$ 52 sec
Conclusion

- New collision checker suited for PRM planners:
  - Faster than fixed-resolution checkers
  - Fully reliable

- Future work:
  - Automatic computation of tight upper bounds on path lengths in $w$-space from robot kinematics
  - Better treatment of pairs of moving bodies (e.g., bodies moving along parallel paths)
Acknowledgements

Most slides from F. Schwarzer, M. Saha, J. Latombe (WAFR 2004) presentation
Greedy Computation of $\eta_{ij}(q)$

**Idea:** BV hierarchy + same recursion as for pure collision checking, but compute distance between boxes ($\rightarrow$ BVs are RSSs)

Algorithm GREEDY-DIST($B_a,B_b$)
1. $h \leftarrow \text{distance}(B_a,B_b)$
2. If $B_a$ and $B_b$ are both triangles then return $h$
3. If $h > 0$ then return $h$
4. If $B_a$ is “bigger” than $B_b$ then switch $B_a$ and $B_b$
5. $\alpha \leftarrow \text{GREEDY-DIST}(B_a,B_{b_1})$
6. If $\alpha > 0$ then
   1. $\beta \leftarrow \text{GREEDY-DIST}(B_a,B_{b_2})$
   2. If $\beta > 0$ then return $\min\{\alpha,\beta\}$
7. Return 0

**GREEDY-DIST**
- is small factor slower than a pure collision checker
- is much faster than BV-based exact or approximate distance computation
- returns lower-bounds that are often much larger than $\frac{1}{2}$ actual distances
RSS: Rectangle Swept Sphere

Fast Distance Queries with Rectangular Swept Sphere Volumes
Eric Larsen Stefan Gottschalk Ming C. Lin Dinesh Manocha, ICAR 2000