Algorithmic Robotics and Motion Planning

Spring 2018

Collision detection and proximity queries

Dan Halperin
School of Computer Science
Tel Aviv University
Today’s lesson

• terminology, motivation, and variants
• the case of convex polytopes: the Dobkin-Kirkpatrick hierarchy and recent log-time query algorithms
• arbitrary polytopes/objects, bounding volume hierarchies
• objects on the move, exploiting temporal coherence
Collision detection, the basic query

• given two objects $P$ and $Q$ (typically in $\mathbb{R}^2$ or $\mathbb{R}^3$) decide whether $P \cap Q \neq \emptyset$

sometimes referred to as interference detection or intersection detection, whereas the term collision detection is reserved for predicting collision while in motion, or continuous collision detection
Variants

• minimum distance between P and Q
• penetration depth
• dynamic (one or both are moving)
• determine first intersection along a trajectory
• 2-body, N-body
• more
Motivation

• sampling-based motion planning
• dynamic simulation
• walkthroughs, virtual environments
• computer games
• molecular modeling
• haptic rendering [displaying computer controlled forces on the user to make them sense the tactual feel of virtual objects]
• ...
THE CASE OF CONVEX POLYTOPES
Linear programming
a typical formulation

• find $x_1, x_2, \ldots, x_d$ to minimize
  • $c_1 x_1 + c_2 x_2 + \ldots + c_d x_d$

• subject to the constraints
  • $a_{11} x_1 + a_{12} x_2 + \ldots + a_{1d} x_d \leq b_1$
  • $a_{21} x_1 + a_{22} x_2 + \ldots + a_{2d} x_d \leq b_2$
  • $\ldots$
  • $a_{n1} x_1 + a_{n2} x_2 + \ldots + a_{nd} x_d \leq b_n$

• $a_{ij}$s, $b_i$s, $c_i$s, real numbers
Linear programming for collision detection

• to decide if two polytopes intersect: throw in all half-spaces (supporting the faces and containing the respective polytope) and look for a feasible solution under arbitrary objective function

• to find a separating hypreplane between the two polytopes, define an LP such that the vertices of the two polytopes are on distinct sides of the (unknown) hyperplane
Finding a separating plane using LP

if \( P = \{p_1, \ldots, p_m\} \) and \( Q = \{q_1, \ldots, q_n\} \), find a hyperplane \( H: ax + by + cz + d = 0 \), such that:

\[
\begin{align*}
    ap_{1x} + bp_{1y} + cp_{1z} + d &> 0 \\
    ap_{2x} + bp_{2y} + cp_{2z} + d &> 0 \\
    \vdots &\quad \vdots &\quad \vdots \\
    ap_{mx} + bp_{my} + cp_{mz} + d &> 0 \\
\end{align*}
\]

\[
\begin{align*}
    aq_{1x} + bq_{1y} + cq_{1z} + d &< 0 \\
    aq_{2x} + bq_{2y} + cq_{2z} + d &< 0 \\
    \vdots &\quad \vdots &\quad \vdots \\
    aq_{nx} + bq_{ny} + cq_{nz} + d &< 0 \\
\end{align*}
\]
Collision detection between two convex polygons

in $O(\log |P| + \log |Q|)$, assuming the vertices of each polygon are given in (CCW) order

[Barba-Langerman ‘15]
• TP: the CH of three vertices of P
• TQ: the edge hull of three edges of Q

• V*(P): the set of vertices of P after pruning
• E*(Q): the set of edges of Q after pruning
• Correctness invariant: P ∩ Q ≠ ∅ iff CH(V*(P)) intersects an edge of E*(Q)
• Separation invariant or intersection invariant holds after each pruning step
More generally, in any fixed dimension

[Barba-Langreman ‘15]

• Given two convex polytopes P and Q in $\mathbb{R}^d$ for a fixed d, they can each be preprocessed separately in time proportional to its size ($|P|, |Q|$) such that collision detection between P and Q can be determined in time $O(\log |P| + \log |Q|)$

• Relying on (a variant of) the Dobkin-Kirkpatrick hierarchy

• An algorithm with such query time was previously known only in the plane
The Dobkin-Kirkpatrick hierarchy

Let $P$ be a convex polytope with a vertex set $V(P)$, where $|V(P)| = n$

define the hierarchy $P_1, P_2, \ldots, P_k$ where:

• $P_1 = P$ and $P_k$ is a simplex
• $P_{i+1} \subset P_i$ and $V(P_{i+1}) \subset V(P_i)$.
• the vertices of $V(P_i) - V(P_{i+1})$ form an independent set in $P_i$. 
DK hierarchy, cont’d

• each face $F$ of $P_{i+1}$ that is not a face of $P_i$ can be associated with a unique vertex $v$ of $P_i$, that lies in the half-space opposite to $P_{i+1}$ with respect to the hyperplane supporting $F$

• the hierarchy has $O(\log n)$ height, $O(n)$ size, and the constant max degree over all vertices of all polytopes in the hierarchy
Polytope–hyperplane separation

let $\sigma(P_i, H)$ be the separating distance of $P_i$ and a hyperplane $H$, obtained at some point $r_i \in V(P_i)$

let $H'$ be a hyperplane parallel to $H$ that touches $r_i$. then:

$$\sigma(P_{i-1}, S) = \min \left\{ \sigma(P_{i-1} \cap H'^{+}, S), \sigma(P_{i-1} \cap H'^{-}, S) \right\}$$
Polytope–hyperplane separation, cont’d

Q: how do we find the intersection of $P_{i-1}$ and $H^-$? Namely the single vertex $v$ of $P_{i-1}$ in $H^-$

A: we maintain a projection of the $P_i$’s onto a hyperplane orthogonal to $H$; this sequence of projections is a DK hierarchy in one dimension less and the $v$ will grow out of the two edges incident to $r_i$ in the projection.

thus $\sigma(P,H)$ can be computed in $O(\log n)$ time

\[
\sigma(P_{i-1}, S) = \min \left\{ \sigma(P_{i-1} \cap H^{(+)}, S), \sigma(P_{i-1} \cap H^{(-)}, S) \right\}
\]
DK hierarchy, more applications

- given two polytopes in $\mathbb{R}^3$, after linear time preprocessing using linear space, the DK hierarchy of the two polytopes (or some variants of it) can be used to answer a variety of proximity queries in (poly)logarithmic time: minimum separation, directional penetration depth
BOUNDING VOLUME HIERARCHIES (BVH)
BVH, basics

• a recursive partitioning of objects that allows for quick pruning of irrelevant intersection tests, represented as a tree
• the root bounds the entire object/ambient space and the leaves bound a small number of features
• construction: bottom-up or top-down
• queries answered by traversing two trees from the root to the leaves
BVH, variants

• partitioning the objects vs. space (e.g., octrees)
• the type of bounding volume: spheres, AABBs, OBBs, spherical shells, ellipsoids, and more
• the type of underlying objects: convex polytopes, polygon soups, spheres, and more
• we will describe a BVH partitioning the object, which are represented as polygon soups, using OBBs
BVH, cost

• total cost of interference detection
  \[ N_v \times C_v + N_p \times C_p \]

• \( v \) stands for volume and \( p \) for primitive, \( N \) for number and \( C \) for cost
  • \( N_v \) : number of bounding volumes pair overlap test
  • \( C_v \) : cost of one overlap test
  • \( N_p \) : number of primitives pairs tested for intersection
  • \( C_p \) : cost of primitive intersection test
BVH, tradeoffs

• total cost of interference detection
  \[ N_v \times C_v + N_p \times C_p \]

• tight-fit bounding volumes vs. simple loose fit BV

• simple BVs have low \( C_v \) but may incur large \( N_v \) and \( N_p \); tight-fit BVs have higher \( C_v \)

• no single hierarchy gives the best solution in all scenarios, even not for the same models in different placements
OBBTrees

[Gottschalk-Lin-Manocha ’96]

• BVH partitioning the objects, which are represented as polygon soups, using OBBs
• tight-fitting OBBs using principal component analysis (PCA)
• improved BV overlap test using a so-called separating-axis theorem
• the tree is constructed top down, splitting the soup by a plane cutting through the major axis of an OBB
Oriented Bounding Boxes (OBBs) compare with the AABB
Tight-fitting OBBs, bounding a set of triangles

- compute the 3-dimensional mean vector, and the $3 \times 3$ covariance matrix:

$$\mu = \frac{1}{3n} \sum_{i=1}^{n} (p_i + q_i + r_i)$$

$$\Sigma = \frac{1}{3n} \sum_{i=1}^{n} ((p_i - \mu)^T (p_i - \mu) + (q_i - \mu)^T (q_i - \mu) + (r_i - \mu)^T (r_i - \mu))$$

- the matrix $\Sigma$ is symmetric, therefore its eigenvectors are mutually orthogonal
- use the normalized eigenvectors as the axes of the bounding box
Tight-fitting OBBs, improvements

• [GLM] use the convex hulls of the triangle vertices to reduce the influence of “buried” vertices
• they use the area of convex hull faces as a continuous version of densely sampling the convex hull boundary
BB overlap test

• two boxes do not intersect iff there exists a line $l$ such that their projections onto this line do not overlap; it is then called the separating axis
The separating-axis theorem

• the separating axis of two oriented boxes is either perpendicular to one of the faces, or can be obtained as the vector product of two box axes

• this means that 15 axes has to be tried: one for each face normal for a total of 6, and one for each pair of axes, one from each box, for a total of 9
The recursive partitioning

• top-down
• find the OBB of the entire soup
• take a plane $\pi$ orthogonal to the longest axis at the mean of the projection of the vertices along the axis
• partition the polygons according to the side of $\pi$ where their center lies
• if one axis does not yield a subdivision proceed to other axes, or determine the set indivisible
Implementation

• RAPID, OBBs [GLM `96]
• PQP (Proximity Query Package) [LGLM `99], uses OBBs for collision detection and rectangular swept-sphere volumes for distance queries
• FCL
OBJECTS ON THE MOVE
Tracking the minimum separation distance

- finding the closest features between two polytopes $P$ and $Q$ is equivalent to finding the closest feature of $P \oplus -Q$ to the origin
- static case:
  - compute the distance of all features of the Minkowski sum from the origin [GJK]
  - use “hill climbing” [Cameron’s improvement]
- dynamic case using temporal coherence:
  - start with the feature found in the previous query
Tracking the minimum separation distance, cont’d

• ample experimental evidence of effectiveness
• alternative approach by Lin and Canny
Reference

• Collision and Proximity Queries, by Lin, Manocha, and Kim
Chapter 39 of the Handbook of Discrete and Computational Geometry, Goodman, O’Rourke, Toth editors, 3rd Edition
high-level comprehensive survey with many references including the paper referred to from the presentation

• see also the website:  
  http://gamma.cs.unc.edu/research/collision/

• The handbook in full: