Algorithmic Robotics and Motion Planning

Spring 2011

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
- arbitrary polytopes/objects, bounding volume hierarchies
- objects on the move, exploiting temporal coherence

Connection to other lessons

- past: Minkowski sums, which constitute a central tool for collision detection and related queries under translation
- future: (i) sampling-based planning, and (ii) self-collision detection for large kinematic structures

Collision detection, the basic query

- given two objects P and Q (typically in R^2 or 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

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

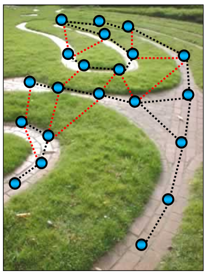
Motivation

- 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]
- …
- sampling-based motion planning

Sampling-based motion planning, reminder

basic PRM (Probabilistic Road-Map) algorithm in a nutshell

- randomly sample *n* valid robot configurations ("milestones")
- connect close-by configurations by dense sampling ("local-planning")
  - discard invalid edges



Note:
For simplicity, configuration space and workspace are identical, in this example

THE CASE OF CONVEX POLYTOPES

Linear programming
a typical formulation

- find $x_1, x_2, \ldots x_d$ to minimize
  - $c_1x_1+c_2x_2+\ldots+c_dx_d$
- subject to the constraints
  - $a_{11}x_1+a_{12}x_2+\ldots a_{1d}x_d \le b_1$
  - $a_{21}x_1+a_{22}x_2+\ldots a_{2d}x_d \le b_2$
  - …
  - $a_{n1}x_1+a_{n2}x_2+\ldots a_{nd}x_d \le b_n$
- a_{ij} s, b_i s, c_is, real numbers

Linear programming, cont'd

- major role in optimization, numerous applications
- efficient solutions and good implementations
- geometric interpretation

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

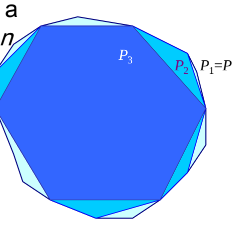
$$\begin{array}{ll} ap_{1x} + bp_{1y} + cp_{1z} + d > 0 & aq_{1x} + bq_{1y} + cq_{1z} + d < 0 \\ ap_{2x} + bp_{2y} + cp_{2z} + d > 0 & aq_{2x} + bq_{2y} + cq_{2z} + d < 0 \\ \vdots & \vdots \\ ap_{mx} + bp_{my} + cp_{mz} + d > 0 & aq_{nx} + bq_{ny} + cq_{nz} + d < 0 \end{array}$$

THE CASE OF CONVEX POLYTOPES

The Dobkin-Kirkpatrick hierarchy

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, …, 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

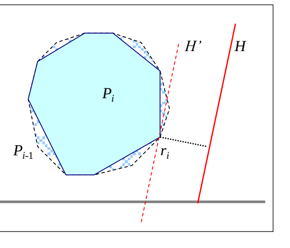
Polytope–hyperplane separation

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:

$$\sigma(P_{i-1},S) = \min\begin{Bmatrix}\sigma(P_{i-1} \cap H'^{(+)},S) \\ \sigma(P_{i-1} \cap H'^{(-)},S)\end{Bmatrix}$$

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



DK hierarchy, more applications

- given two polytopes in 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

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

BOUNDING VOLUME HIERARCHIES (BVH)

BVH, cost

- total cost of interference detection

  N_v x C_v + N_p x 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

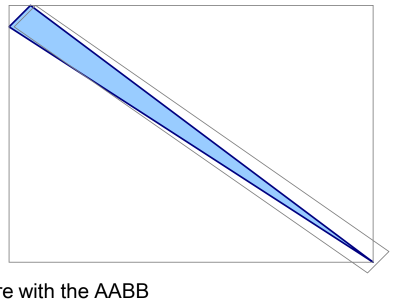
- total cost of interference detection

  N_v x C_v + N_p x 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

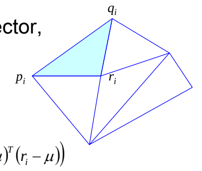
Oriented Bounding Boxes (OBBs)



compare with the AABB

Tight-fitting OBBs, bounding a set of triangles

▪ compute the 3-dimensional mean vector, and the 3×3 covariance matrix:



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

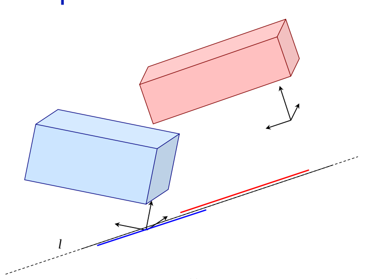
$$\Sigma = \frac{1}{3n}\sum_{i=1}^{n}\left((p_i-\mu)^T(p_i-\mu)+(q_i-\mu)^T(q_i-\mu)+(r_i-\mu)^T(r_i-\mu)\right)$$

▪ the matrix Σ 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 π 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 π 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

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 and Manocha,
  Chapter 35 of the Handbook of Discrete and Computational Geometry,
  Goodman and O’Rourke, eds

high-level comprehensive survey with many references

Assignment no. 3

- all the questions refer to a rod (segment) translating and rotating among polygons in the plane
- 3.1 show that the actual complexity of the free space is $O(n^2)$
- 3.2 (**p2**) devise and implement a simple and effective structure for collision detection between a fixed length rod and a set of polygons in the plane
- 3.3 (**p2**) solve this motion planning problem with PRM, using your solution of 3.2 for collision detection
- 3.4 (optional, difficult) devise an efficient algorithm to solve the problem

THE END