Computing and Software Department, McMaster University

Introduction to GJK

John McCutchan

November 9, 2006

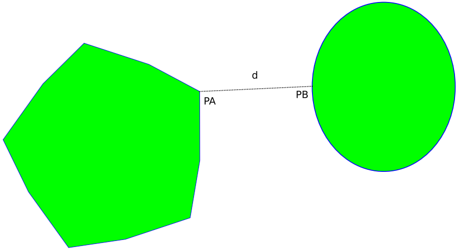
Introduction

- Introduction to GJK
- Terminology
- Algorithm in detail
- Example
- Minkowski Difference (GJK with two convex objects)
- Polyhedra Support Function
- Sphere Support Function
- Cylinder Support Function
- Transformation Support Function
- Computing P of minimum norm in CH(Q) and reducing Q
- References

Introduction to GJK

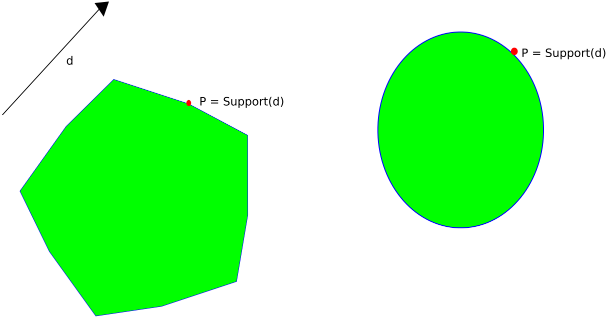
Given two convex shapes

- Computes distance d
- Can also compute closest pair of points P_A and P_B



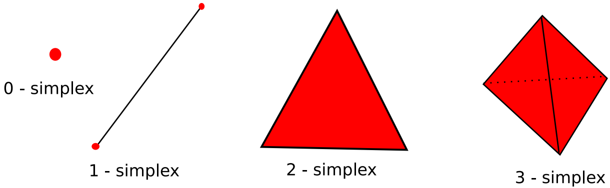
Terminology – Support Point

Supporting (extreme) point P for direction d returned by support mapping function Support(d)

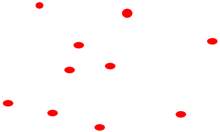


Terminology – Simplex

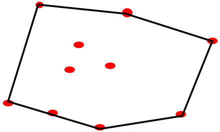
Simplex



Terminology – Convex Hull



Convex Set C



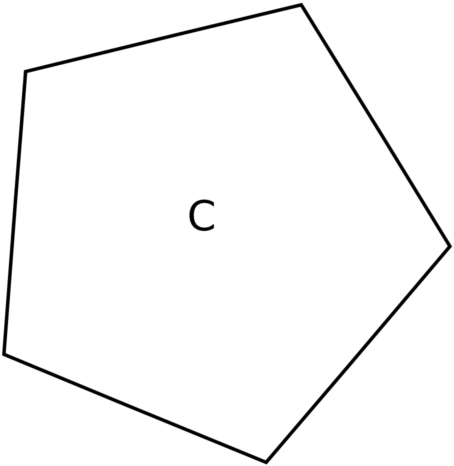
Convex Hull CH(C)

Algorithm in detail

- 1. Initialize simplex set Q with up to d+1 points from C (in d dimensions)
- 2. Compute point P of minimum norm in CH(Q)
- 3. If P is the origin, exit; return 0.0;
- 4. Reduce Q to the smallest subset Q' of Q, such that P in CH(Q')
- 5. Let V = Support(-P)
- 6. If V no more extreme in direction -P than P itself, exit; return length(P)
- 7. Add V to Q. Go to step 2

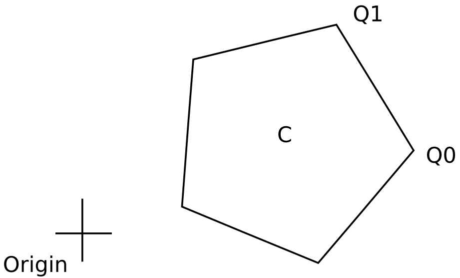
Example 1/10

Input: Convex shape C



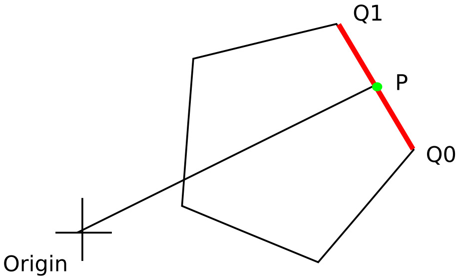
Example 2/10

1. Initialize simplex set Q with up to d+1 points from C (in d dimensions)
$Q = [Q1, Q0]$



Example 3/10

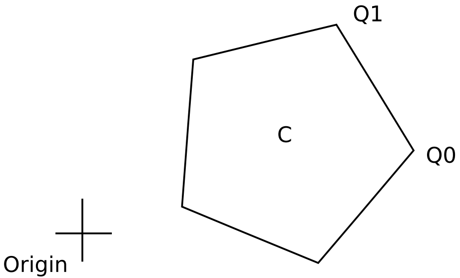
2. Compute point P of minimum norm in CH(Q)



Example 4/10

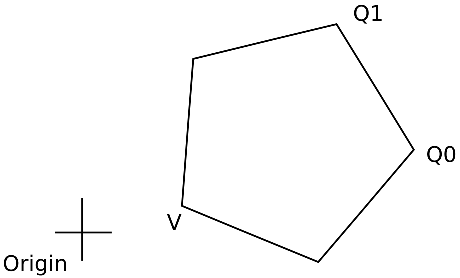
3. If P is the origin, exit; return 0.0;
4. Reduce Q to the smallest subset Q' of Q, such that P in CH(Q')

$Q = [Q1, Q0]$



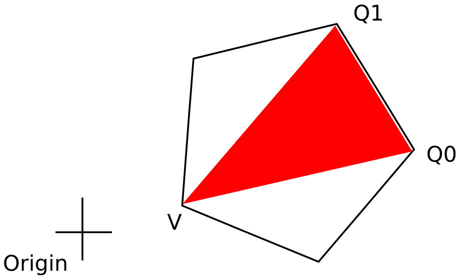
Example 5/10

5. Let V = Support(-P)



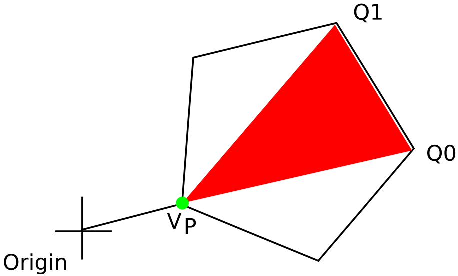
Example 6/10

6. If V no more extreme in direction -P than P itself, exit; return length(P)
7. Add V to Q, go to step 2
$Q = [Q1, Q0, V]$



Example 7/10

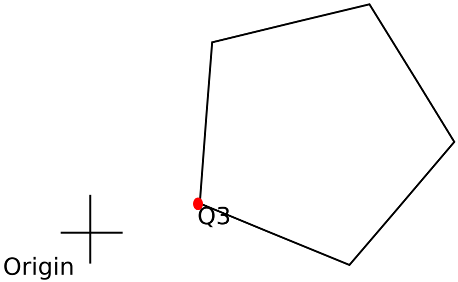
2. Compute point P of minimum norm in CH(Q)
$Q = [Q1, Q0, V]$



Example 8/10

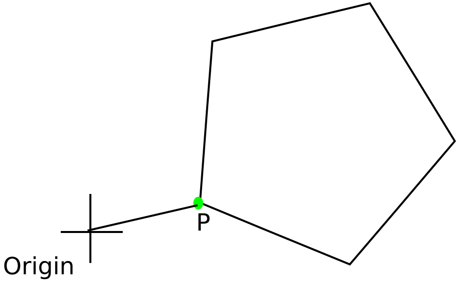
3. If P is the origin, exit; return 0.0;
4. Reduce Q to the smallest subset Q' of Q, such that P in CH(Q')

$Q = [Q3]$



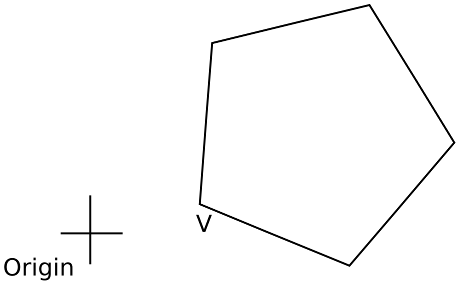
Example 9/10

5. Let V = Support(-P)



Example 10/10

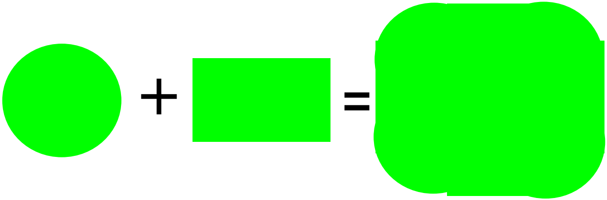
6. If V no more extreme in direction -P than P itself, exit; return length(P)



Minkowski Difference (GJK with two convex objects)

- Problem: How do we handle two convex objects, A and B?
- Solution: Use Minkowski Difference of A and B.

Minkowski Sum

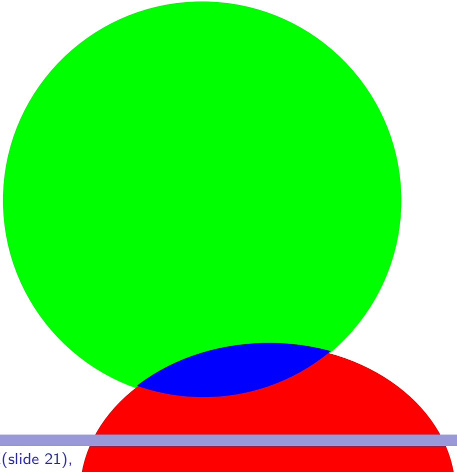


Minkowski Sum

$$MinkowskiSum(A, B) = a + b : a \in A, b \in B$$
$$MinkowskiDifference(A, B) = MinkowskiSum(A, -B)$$
$$MinkowskiDifference(A, B) = a - b : a \in A, b \in B$$

Minkowski Difference

- What happens to points in A and B that are overlapping when you take the minkowski difference?
- They are mapped to the origin.



Minkowski Difference

- So, A and B are intersecting iff MinkowskiDifference(A,B) contains the origin!
- The algorithm can stay the same, we just need to change the support point function to compute the Minkowski Difference
- MinkowskiDiffSupport(A,B,d) = A.Support(d) - B.Support(-d)

Support Point Functions – Polyhedra

Given: C – A convex hull of points

$$Support(d) = max(d \cdot p : p \in C)$$

Support Point Functions – Sphere

Given: Sphere centered at c with radius r

$$Support(d) = c + r\frac{d}{||d||}$$

Support Point Functions – Cylinder

Given: Cylinder centered at c and whose central axis is spanned by the unit vector u. Let the radius of the cylinder be r and the half height be n. As well, Let $w = d - (u \cdot d)u$ be the component of d orthogonal to u.
If $w \neq 0$:

$$Support(d) = c + sign(u \cdot d)nu + r\frac{w}{||w||}$$

else:

$$Support(d) = c + sign(u \cdot d)nu$$

Support Point Functions – Transformation

Given: $T(x) = Bx + c$ Where B is the rotation matrix's basis and c is the translation. SupportC is the support function of the untransformed convex object.

$$Support(SupportC, d) = T(SupportC(B^T d))$$

Computing P of minimum norm in CH(Q') and reducing Q to Q'

- Overview of affine hulls and convex hulls
- Equivelance of affine and convex hulls
- Finding closest point on affine hull to origin
- Finding smallest Q' where P is in CH(Q')

Affine and Convex Hulls

- Affine Hull: $AH(S) = \lambda_1 x_1 + \lambda_2 x_2 + ... + \lambda_k x_k | x_i \in S, \lambda_i \in R$
- $i = 1, ..., k, \lambda_1 + \lambda_2 + ...\lambda_k = 1, k = 1, 2, ...$
- Convex Hull: $CH(S) = \lambda_1 x_1 + \lambda_2 x_2 + ... + \lambda_k x_k | x_i \in S, \lambda_i \in R$
- $i = 1, ..., k, \lambda_1 + \lambda_2 + ...\lambda_k = 1, k = 1, 2, ..., \lambda_i >= 0$

The point P closest to the origin is defined as a convex combination of the points in Q. $P = \sum_{i=1}^{n} \lambda_i x_i$ where $\sum_{i=1}^{n} \lambda_i = 1.0$ and $\lambda_i \geq 0$ Since we are looking for the smallest Q' that contains P we can add another restriction: $\lambda > 0$ Now we are looking for $Q' = x_i : \lambda_i > 0$

Equivelance of affine and convex hulls

If we can find a set $Q' = x_i : i \in Y$ for which $i \in Y, \lambda_i > 0.0$ in

$$AH(Q') = \sum_{i \in Y} \lambda_i x_i, \sum_{i \in Y} \lambda_i = 1.0$$

and for all $j \notin Y, \lambda_j <= 0.0$ in

$$AH(Q' \cup X_j) = \sum_{i \in Y \cup j} \lambda_i x_i, \sum_{i \in Y \cup j} \lambda_i = 1.0$$

For such a set Q' we have $P(AH(Q')) = P(CH(Q'))$

Finding Q'

- You find Q' by iterating over all subsets of Q and checking if they fit the two previous conditions.
- Need to compute all of the λ_i terms for all subsets

Closest point to origin on affine hull of triangle (2-simplex)

- Affine hull of a triangle is plane containing the triangle vertices
- We need to find a point $P = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3$
- P will be closest to the origin if the vector from the origin to P is perpendicular to the plane
- In other words when P is perpendicular to the triangles edges
- Two arbitrary edges are x_1x_2 and x_1x_3 we want $x_1x_2 \cdot P = 0$ and $x_1x_3 \cdot P = 0$
- If we substitute $\lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3$ for P with both edges we get:

Closest point to origin on affine hull of triangle (2-simplex)

$$A = \begin{bmatrix} 1 & 1 & 1 \\ (x_2 - x_1) \cdot x_1 + (x_2 - x_1) \cdot x_2 + (x_2 - x_1) \cdot x_3 \\ (x_3 - x_1) \cdot x_1 + (x_3 - x_1) \cdot x_2 + (x_3 - x_1) \cdot x_3 \end{bmatrix}$$

$$b = [1, 0, 0]$$

$$x = [\lambda_1, \lambda_2, \lambda_3]$$

Johnson's Distance sub algorithm

In general:

$$A = \begin{bmatrix} 1 & ... & 1 \\ (x_2 - x_1) \cdot x_1 & ... & (x_2 - x_1) \cdot x_m \\ ... & & ... \\ (x_m - x_1) \cdot x_1 & ... & (x_m - x_1) \cdot x_m \end{bmatrix}$$

$$b = [1, 0, ..., 0]$$

$$x = [\lambda_1, ..., \lambda_m]$$

Johnson's Distance sub algorithm

- 1. Need to solve $Ax = b$ for every subset of Q
- 2. Search for smallest Q' which satisfies above two conditions
- 3. $Q = Q'$ and $P = \sum_{i \in Q'} \lambda_i x_i$

Johnson's Distance sub algorithm

- How can all of the $Ax = b$ systems be solved efficiently?
- Gino: Cramers Rule
- McCutchan: Use Maple/Matlab to precompute generic $\lambda = A^{-1}b$ for simplex size of 2,3,4
- Gino's is faster but McCutchan's is simple and obvious.
- GJK is already so fast and O(1) that speed difference not noticeable on modern machines

Alternative way to find Q'

- Look at the problem geometrically
- Use voronoi region checks to find which part of simplex the origin is in.
- Solve single set of equations once proper sub simplex has been found.
- Pros: Most efficient and Intuitive way of working with GJK.
- Cons: May be floating point issues in using two different mathematical formulations. One for determining the sub simplex and the other for solving for the lambda values.

References

- Christer Ericson's Real-time Collision Detection
- Christer Ericson's Sigraph slides on GJK
- Gino Van Den Bergen's Collision Detection in interactive 3D environments