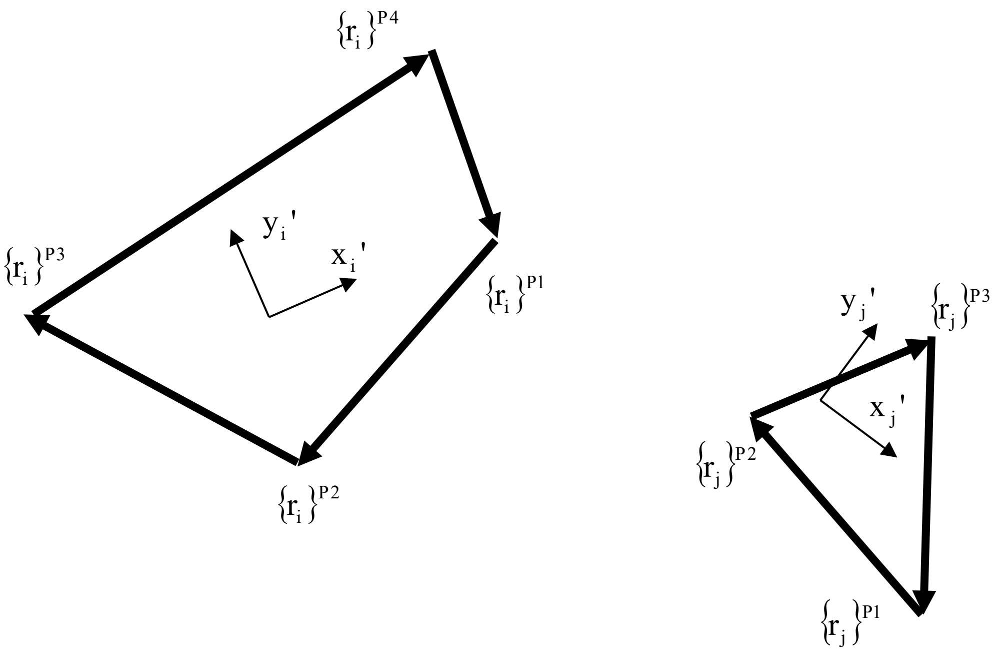
Collision Detection for Polygonal Objects

given global location $\{r_i\}$ and attitude ϕ_i for all bodies at time t

given $\{s_i\}'^{P}$ for all vertices on all bodies (constants)

polygons must not be self-crossing

polygons must not have holes



Bounding Circle - quick check

before the simulation, find maximum radius $\rho_i = \max\left(\text{norm}\{s_i\}'^{\,P}\right)$ over all vertices for each body (one constant scalar value per body that may be computed a priori)

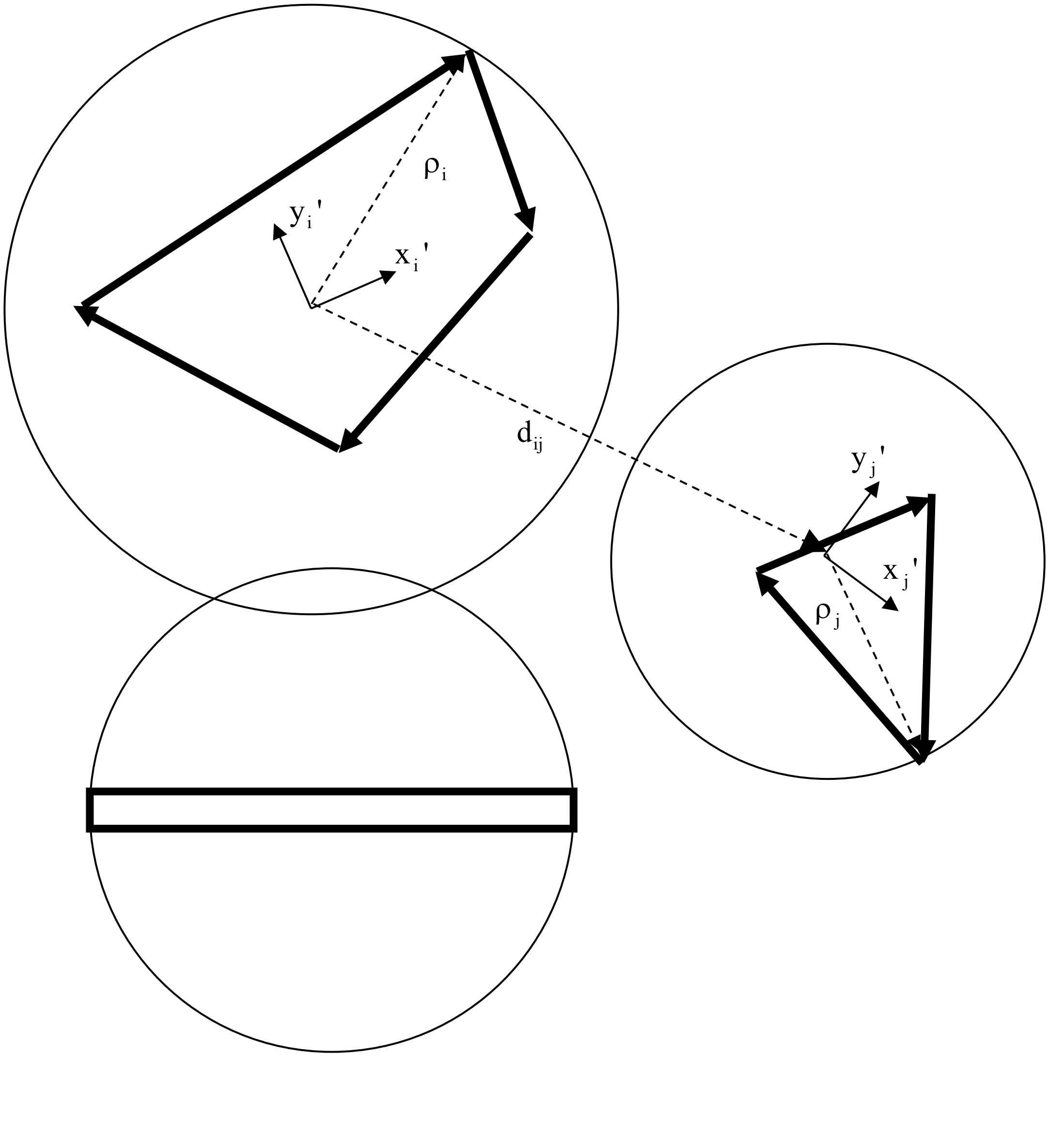
at each time t, compute center distance $d_{ij} = \text{norm}\left(\{r_j\} - \{r_i\}\right)$ between all pairs of bodies

if $d_{ij} > \rho_i + \rho_j$ then no collision can occur between bodies i and j

if $d_{ij} \le \rho_i + \rho_j$ then bodies i and j are candidates for collision detection

does not require calculation of $\{r_i\}^P$ for any vertices on any bodies

works best for centroidal origins and objects with low aspect ratios



Axis Aligned Bounding Box (AABB) - quick check

at each time t, calculate $\{r_i\}^P$ for all vertices on all bodies

find AABB for each body x_i^{MAX} x_i^{MIN} y_i^{MAX} y_i^{MIN}

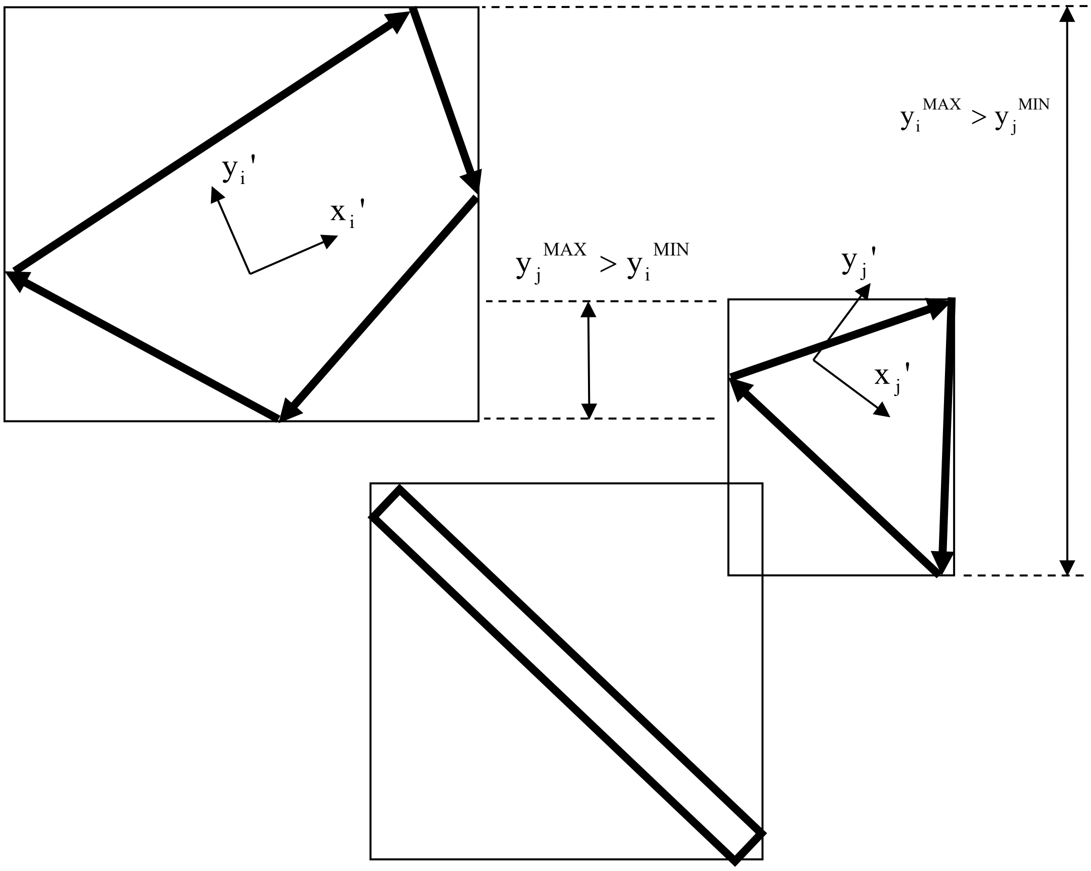
test between all pairs of bodies

$$\left(x_i^{MAX} \geq x_j^{MIN}\right) \text{AND} \left(x_j^{MAX} \geq x_i^{MIN}\right) \text{AND} \left(y_i^{MAX} \geq y_j^{MIN}\right) \text{AND} \left(y_j^{MAX} \geq y_i^{MIN}\right)$$

if true, then AABBs intersect and bodies i and j are candidates for collision detection

if false, then AABBs do not intersect and collision cannot occur between bodies i and j

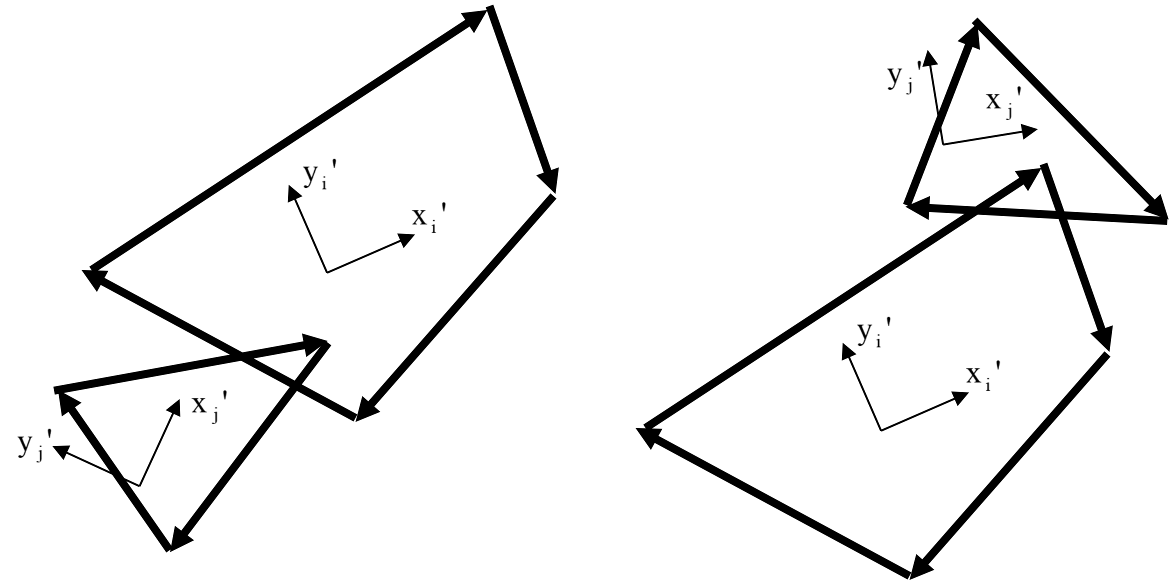
works best for objects with low aspect ratios



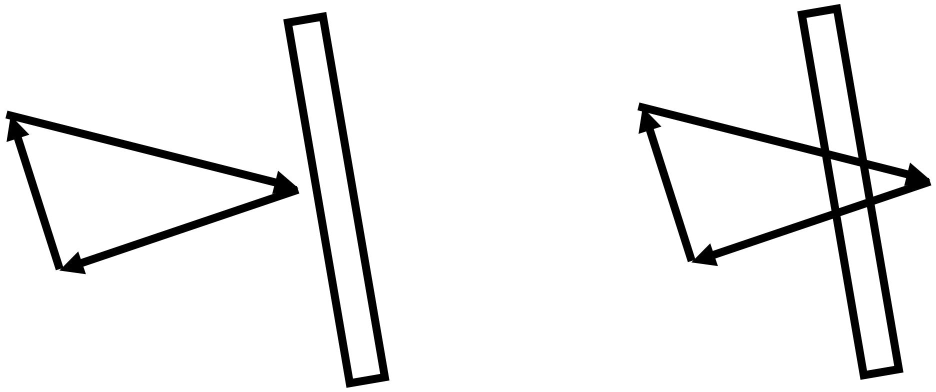
Point in Polygon

must check if any vertices on body j are inside body i AND check if any vertices on body i are inside body j

computational complexity O(n_i x n_j) for convex objects and O(2 n_i x n_j) for ray casting



point in polygon does not always work for thin bodies (may need edge intersection)



Point in Polygon – Convex Objects

vertices must be ordered CW around polygon

at each time t, calculate $\{r_i\}^P$ for all points on all bodies that are candidates for collision

select one point $\{r_j\}^P$ on body j and test if it is to the right or to the left of edge $\{r_i\}^Q - \{r_i\}^P$ on body i

external normal $\{\hat{n}\} = \text{unit}([R](\{r_i\}^Q - \{r_i\}^P))$ to the left of current edge

if $(\{r_j\}^P - \{r_i\}^P)^T \{\hat{n}\} > 0$, the point is to the left of the edge, is outside the polygon and the search across edges may be terminated for that point

if $(\{r_j\}^P - \{r_i\}^P)^T \{\hat{n}\} = 0$, the point is on the edge and may be inside the polygon

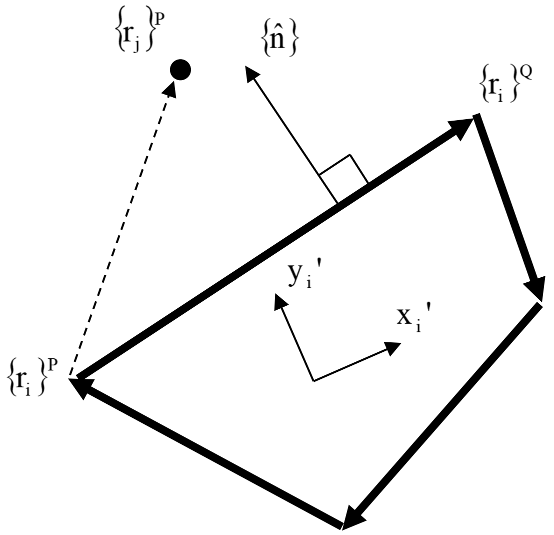
if $(\{r_j\}^P - \{r_i\}^P)^T \{\hat{n}\} < 0$, the point is to the right of the edge and may be inside the polygon

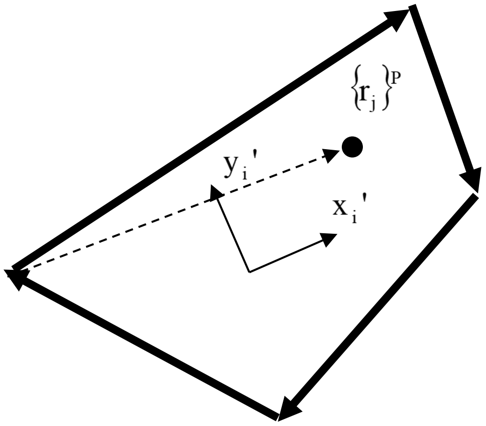
expand $(\{r_j\}^P - \{r_i\}^P)^T [R](\{r_i\}^Q - \{r_i\}^P) = (y_j^P - y_i^P)(x_i^Q - x_i^P) - (x_j^P - x_i^P)(y_i^Q - y_i^P)$

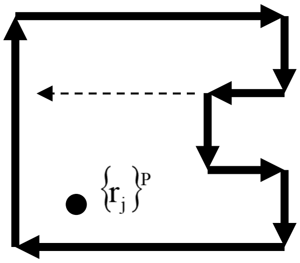
point $\{r_j\}^P$ must be to the right of ALL edges to be inside body i

computational complexity O(n_i x n_j /2) because may terminate search early

does not work for bodies with concavities or holes







Point in Polygon – Ray Casting

at each time t, calculate $\{r_i\}^P$ for all points on all bodies that are candidates for collision

select one point $\{r_j\}^P$ on body j and test if it is inside body i

cast an infinite ray in any direction from $\{r_j\}^P$ and compute intersections with all edges on body i

an infinite ray to the right from $\{r_j\}^P$ intersects edge $\{r_i\}^Q - \{r_i\}^P$ if $(0 \le d_i < 1)$ AND $(d_j \ge 0)$

$$d_i = \frac{y_j^{\,P} - y_i^{\,P}}{y_i^{\,Q} - y_i^{\,P}} \qquad d_j = (x_i^{\,Q} - x_i^{\,P})d_i - (x_j^{\,P} - x_i^{\,P})$$

if there are an even number of intersections, $\{r_j\}^P$ is not inside body i

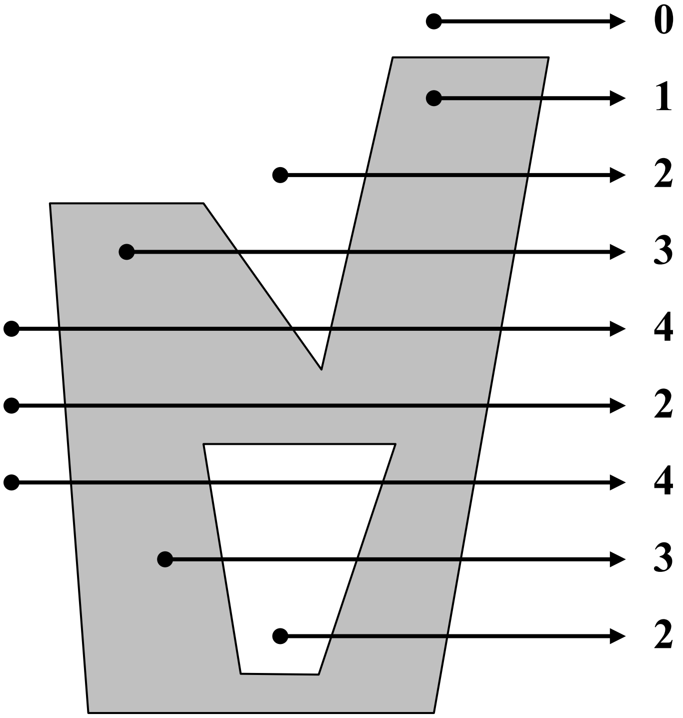
if there are an odd number of intersections, $\{r_j\}^P$ is inside body i

computational complexity O(n_i x n_j)

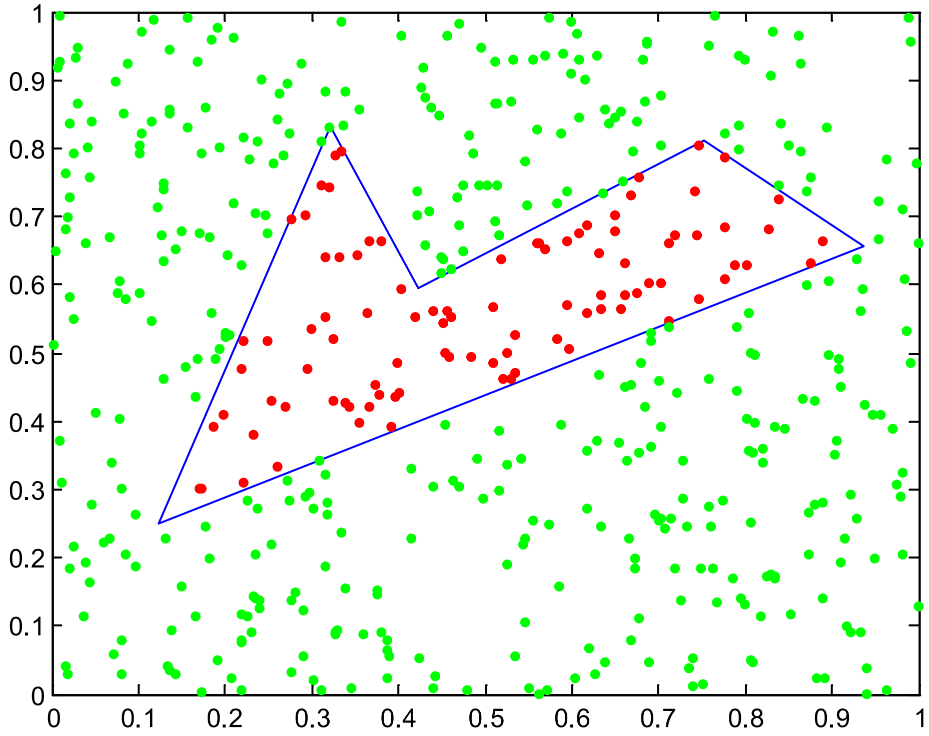
works for concavities, holes and self-crossing and is independent of CW versus CCW boundary

special case when $\{r_j\}^P$ is on an edge or vertex of body i

algorithm used by MATLAB “inpolygon”

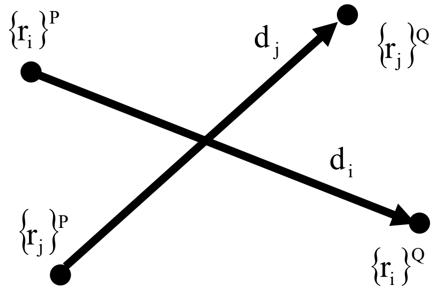


```
% t_inpolygon.m - test inpolygon
% HJSIII, 11.4.13

clear

% vertices for body i - bounded 0 to 1
ni = 5;
ri = rand(2,ni);
ri = [ ri  ri(:,1) ];  % close the boundary

% points for j
nj = 500;
rj = rand(2,nj);

% which are inside?
%               points j             vertices i
in = inpolygon( rj(1,:),rj(2,:),    ri(1,:),ri(2,:) );
figure( 1 )
  clf
  plot( ri(1,:),ri(2,:),'b', rj(1,in),rj(2,in),'.r', rj(1,~in),rj(2,~in),'.g' )

% bottom of t_inpolygon
```



Edge Intersection

must check every edge on body i for intersection with every edge on body j



$\{r_i\}^P + d_i\left(\{r_i\}^Q - \{r_i\}^P\right) = \{r_j\}^P + d_j\left(\{r_j\}^Q - \{r_j\}^P\right)$ intersect if $0 \le d_i \le 1$ and $0 \le d_j \le 1$

$$\left[\left(\{r_i\}^Q - \{r_i\}^P\right) \quad -\left(\{r_j\}^Q - \{r_j\}^P\right)\right]\begin{Bmatrix} d_i \\ d_j \end{Bmatrix} = \left(\{r_j\}^P - \{r_i\}^P\right)$$

$$d_i = \frac{\left(x_j^{\,Q} - x_j^{\,P}\right)\left(y_j^{\,P} - y_i^{\,P}\right) - \left(y_j^{\,Q} - y_j^{\,P}\right)\left(x_j^{\,P} - x_i^{\,P}\right)}{\left(x_j^{\,Q} - x_j^{\,P}\right)\left(y_i^{\,Q} - y_i^{\,P}\right) - \left(x_i^{\,Q} - x_i^{\,P}\right)\left(y_j^{\,Q} - y_j^{\,P}\right)}$$

$$d_j = \frac{\left(x_i^{\,Q} - x_i^{\,P}\right)\left(y_j^{\,P} - y_i^{\,P}\right) - \left(y_i^{\,Q} - y_i^{\,P}\right)\left(x_j^{\,P} - x_i^{\,P}\right)}{\left(x_j^{\,Q} - x_j^{\,P}\right)\left(y_i^{\,Q} - y_i^{\,P}\right) - \left(x_i^{\,Q} - x_i^{\,P}\right)\left(y_j^{\,Q} - y_j^{\,P}\right)}$$

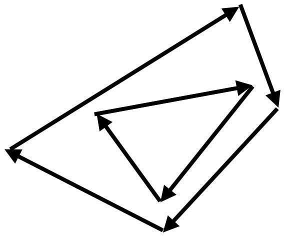
if denominator = 0, segments are parallel

must check if they are coincident $\left(\{r_j\}^P - \{r_i\}^P\right)^T [R]\left(\{r_i\}^Q - \{r_i\}^P\right) = 0$

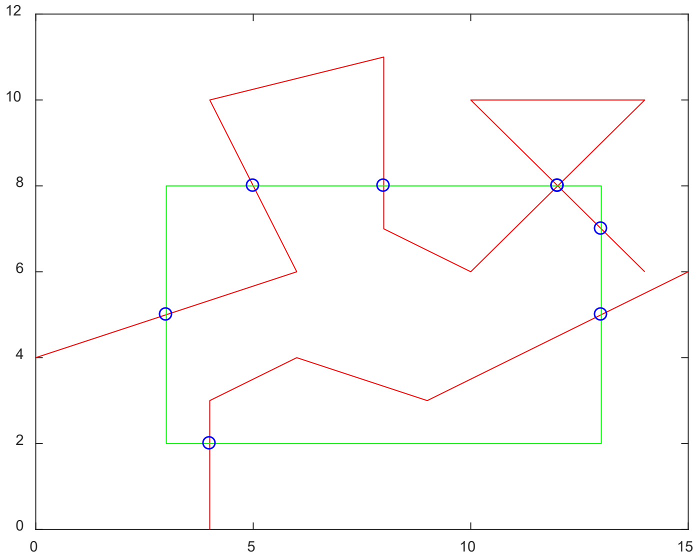
expand $\left(y_j^{\,P} - y_i^{\,P}\right)\left(x_i^{\,Q} - x_i^{\,P}\right) - \left(x_j^{\,P} - x_i^{\,P}\right)\left(y_i^{\,Q} - y_i^{\,P}\right) = 0$

computational complexity O(n_i x n_j)

edge intersection does not always work if one body is completely inside the other (may need point in polygon)



```
% t_polyxpoly -  test polyxpoly
% HJSIII, 18.04.30

clear

% rectangle - CW closed curve
xbox = [ 3  3  13  13  3 ];
ybox = [ 2  8   8   2  2 ];

% two-part polyline - open curve with gap
xline = [ 0  6   4   8  8  10  14  10  14  NaN  4  4  6  9  15 ];
yline = [ 4  6  10  11  7   6  10  10   6  NaN  0  3  4  3   6 ];

% find edge intersections
[ xi, yi, ii ] = polyxpoly( xline, yline, xbox, ybox );

% list results
fprintf( 'line edge  box edge\n' )
for i = 1 : length(ii),
  fprintf( '%10d', ii(i,1) )
  fprintf( '%10d\n', ii(i,2) )
end

% show results
figure( 1 )
  clf
  plot( xline,yline,'r', xbox,ybox,'g', xi,yi,'bo' )

% bottom of t_polyxpoly

****************************************************

>> t_polyxpoly
 line edge  box edge
         1         1
         2         2
         4         2
         6         2
         8         2
         8         3
        14         3
        11         4
```



Separating Axis Theorem (SAT)

vertices must be ordered CW around polygon

before the simulation, select one edge $\{s_i\}'^{Q}-\{s_i\}'^{P}$ on body i and compute external normal $\{\hat{n}\}=\text{unit}\left([R]\left(\{s_i\}'^{Q}-\{s_i\}'^{P}\right)\right)$

compute the projection p_i^{R} for all points $\{s_i\}'^{R}$ onto the normal and store the minimum value p_i^{MIN} (one scalar constant per edge that may be computed a priori)

$$p_i^{R}=\left(\{s_i\}'^{R}-\{s_i\}'^{P}\right)^{T}\text{unit}\left([R]\left(\{s_i\}'^{Q}-\{s_i\}'^{P}\right)\right)$$

at each time t, calculate $\{r_i\}^{P}$ for all points on all bodies that are candidates for collision

select edge $\{r_i\}^{Q}-\{r_i\}^{P}$ on body i and compute external normal $\{\hat{n}\}=\text{unit}\left([R]\left(\{r_i\}^{Q}-\{r_i\}^{P}\right)\right)$ called the separating axis

select point $\{r_j\}^{P}$ on body j and compute its projection p_j^{P} onto the separating axis

$$p_j^{P}=\left(\{r_j\}^{P}-\{r_i\}^{P}\right)^{T}\{\hat{n}\}$$

compute the projection p_j^{P} onto the separating axis for all other points on body j and store the maximum p_j^{MAX} and minimum p_j^{MIN} values

check for overlap of projections onto the separating axis

$$\left(0\geq p_j^{MIN}\right)\text{AND}\left(p_j^{MAX}\geq p_i^{MIN}\right)$$

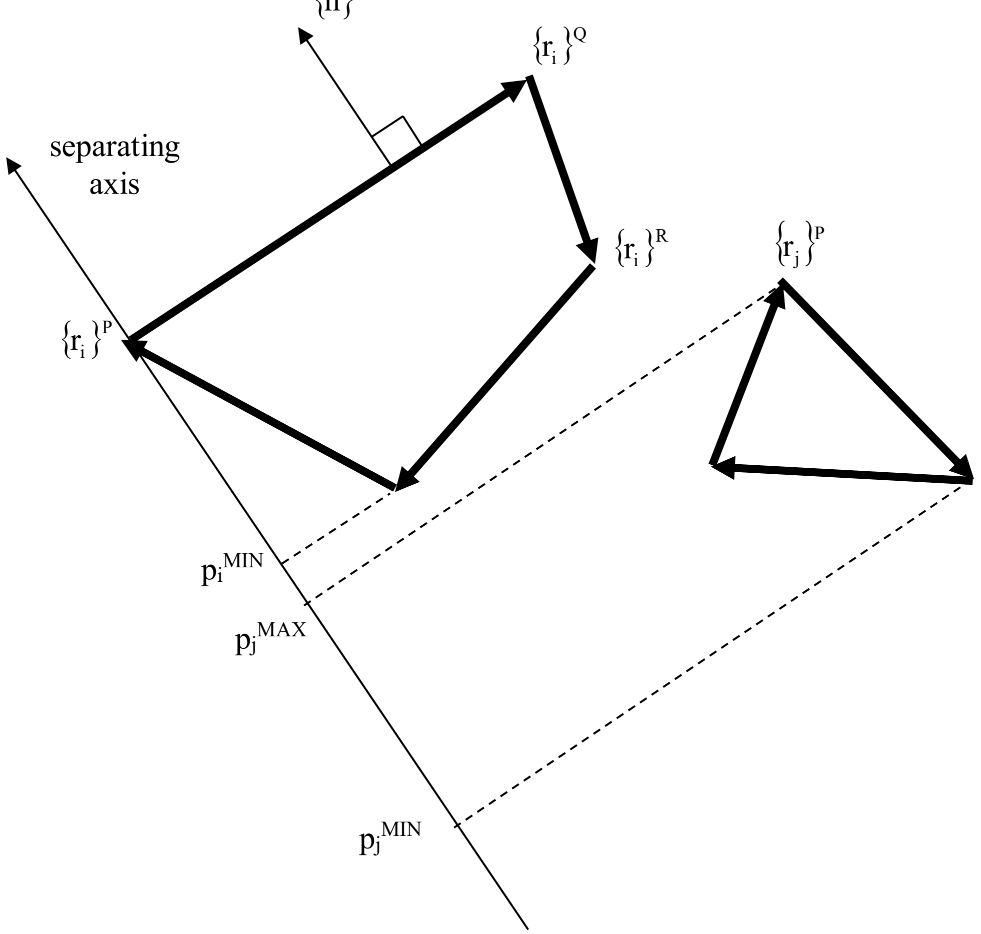
if false, the projections do not overlap, there can be no collision and the test may be terminated

if true, the projections overlap and must continue checking all other edges $\{r_i\}^{Q}-\{r_i\}^{P}$

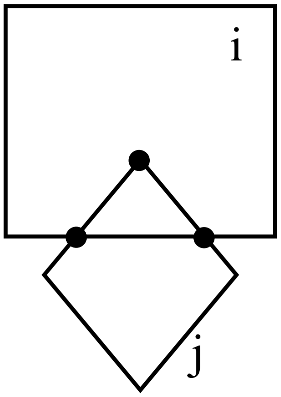
does not work for bodies with concavities or holes

very similar to convex body point in polygon however do not need to check (i in j) and (j in i)

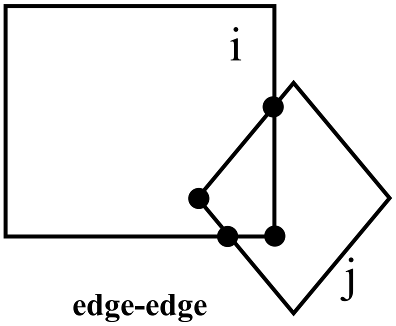
computational complexity O(n_i x n_j / 2) because may terminate search early AND do not need to check body i against body j


$\{\hat{n}\}$
$\{r_i\}^Q$
separating axis
$\{r_i\}^R$
$\{r_j\}^P$
$\{r_i\}^P$
p_i^{MIN}
p_j^{MAX}
p_j^{MIN}

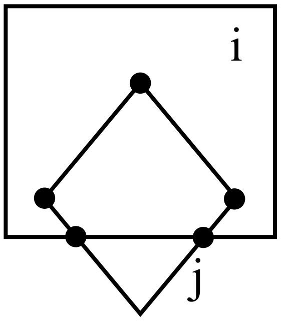
Classification of Contacts



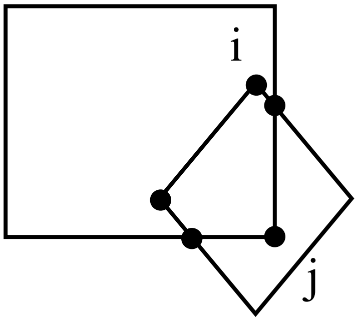
vertex-edge



edge-edge



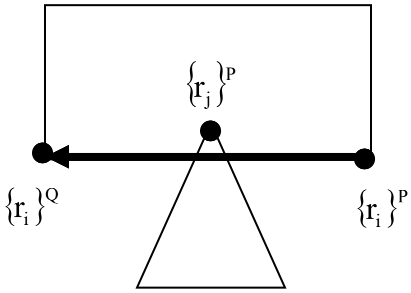
multiple vertex-edge



multiple edge-edge

	Body i vertices in j	Body i edge intersections	Body j vertices in i	Body j edge intersections	Contact points
vertex-edge	0	1	1	2	3
edge-edge	1	2	1	2	4
multiple vertex-edge	0	1	≥2	2	5
multiple edge-edge	1	2	≥2	2	5

Interpolating Time of Collision (vertex-edge)



given candidate vertex $\{r_j\}_t^P$ on body j that collides with body i at time t

given candidate edge $\{r_i\}_t^Q - \{r_i\}_t^P$ on body i that has two edge intersections with body j at time t

must know position and velocity for all three points at time t-h (time step h)

Constant velocity predictor

predict position of all three points at local time $0 \leq \Delta t \leq h$ using vales from t-h

$$\{r_i\}^P = \{r_i\}_{t-h}^P + \{\dot{r}_i\}_{t-h}^P(\Delta t)$$

$$\{r_i\}^Q = \{r_i\}_{t-h}^Q + \{\dot{r}_i\}_{t-h}^Q(\Delta t)$$

$$\{r_j\}^P = \{r_j\}_{t-h}^P + \{\dot{r}_j\}_{t-h}^P(\Delta t)$$

when vertex $\{r_j\}^P$ intersects edge $\{r_i\}^Q - \{r_i\}^P$ $\left(\{r_j\}^P - \{r_i\}^P\right)^T[R]\left(\{r_i\}^Q - \{r_i\}^P\right) = 0$

substituting position equations and collecting similar terms

$$a(\Delta t)^2 + b(\Delta t) + c = 0$$

$$a = \left(\{\dot{r}_j\}_{t-h}^P - \{\dot{r}_i\}_{t-h}^P\right)^T[R]\left(\{\dot{r}_i\}_{t-h}^Q - \{\dot{r}_i\}_{t-h}^P\right)$$

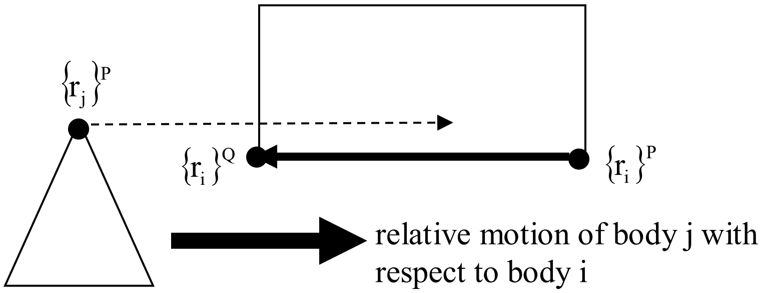
$$b = \left(\{r_j\}_{t-h}^P - \{r_i\}_{t-h}^P\right)^T[R]\left(\{\dot{r}_i\}_{t-h}^Q - \{\dot{r}_i\}_{t-h}^P\right) + \left(\{\dot{r}_j\}_{t-h}^P - \{\dot{r}_i\}_{t-h}^P\right)^T[R]\left(\{r_i\}_{t-h}^Q - \{r_i\}_{t-h}^P\right)$$

$$c = \left(\{r_j\}_{t-h}^P - \{r_i\}_{t-h}^P\right)^T[R]\left(\{r_i\}_{t-h}^Q - \{r_i\}_{t-h}^P\right)$$

$$\Delta t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

must check $0 \le \Delta t \le h$ and it must not be imaginary

if not true, then that vertex did not actually collide with that edge



Constant acceleration predictor

predict position of all three points at local time $0 \le \Delta t \le h$ measured from t-h

$$\{r_i\}^P = \{r_i\}^P_{t-h} + \{\dot{r}_i\}^P_{t-h}(\Delta t) + \tfrac{1}{2}\{\ddot{r}_i\}^P_{t-h}(\Delta t)^2$$

$$\{r_i\}^Q = \{r_i\}^Q_{t-h} + \{\dot{r}_i\}^Q_{t-h}(\Delta t) + \tfrac{1}{2}\{\ddot{r}_i\}^Q_{t-h}(\Delta t)^2$$

$$\{r_j\}^P = \{r_j\}^P_{t-h} + \{\dot{r}_j\}^P_{t-h}(\Delta t) + \tfrac{1}{2}\{\ddot{r}_j\}^P_{t-h}(\Delta t)^2$$

substituting position equations and collecting similar terms

$$c_4(\Delta t)^4 + c_3(\Delta t)^3 + c_2(\Delta t)^2 + c_1(\Delta t) + c_0 = 0$$

$$c_4 = \tfrac{1}{4}\left(\{\ddot{r}_j\}^P_{t-h} - \{\ddot{r}_i\}^P_{t-h}\right)^T [R]\left(\{\ddot{r}_i\}^P_{t-h} - \{\ddot{r}_i\}^P_{t-h}\right)$$

$$c_3 = \tfrac{1}{2}\left(\{\dot{r}_j\}^P_{t-h} - \{\dot{r}_i\}^P_{t-h}\right)^T [R]\left(\{\ddot{r}_i\}^Q_{t-h} - \{\ddot{r}_i\}^P_{t-h}\right) + \tfrac{1}{2}\left(\{\ddot{r}_j\}^P_{t-h} - \{\ddot{r}_i\}^P_{t-h}\right)^T [R]\left(\{\dot{r}_i\}^Q_{t-h} - \{\dot{r}_i\}^P_{t-h}\right)$$

$$\begin{aligned} c_2 = {} & \tfrac{1}{2}\left(\{r_j\}^P_{t-h} - \{r_i\}^P_{t-h}\right)^T [R]\left(\{\ddot{r}_i\}^Q_{t-h} - \{\ddot{r}_i\}^P_{t-h}\right) + \left(\{\dot{r}_j\}^P_{t-h} - \{\dot{r}_i\}^P_{t-h}\right)^T [R]\left(\{\dot{r}_i\}^Q_{t-h} - \{\dot{r}_i\}^P_{t-h}\right) \\ & + \tfrac{1}{2}\left(\{\ddot{r}_j\}^P_{t-h} - \{\ddot{r}_i\}^P_{t-h}\right)^T [R]\left(\{r_i\}^Q_{t-h} - \{r_i\}^P_{t-h}\right) \end{aligned}$$

$$c_1 = \left(\{r_j\}^P_{t-h} - \{r_i\}^P_{t-h}\right)^T [R]\left(\{\dot{r}_i\}^Q_{t-h} - \{\dot{r}_i\}^P_{t-h}\right) + \left(\{\dot{r}_j\}^P_{t-h} - \{\dot{r}_i\}^P_{t-h}\right)^T [R]\left(\{r_i\}^Q_{t-h} - \{r_i\}^P_{t-h}\right) \quad \text{(same as b above)}$$

$$c_0 = \left(\left\{r_j\right\}_{t-h}^{P} - \left\{r_i\right\}_{t-h}^{P}\right)^T \left[R\right]\left(\left\{r_i\right\}_{t-h}^{Q} - \left\{r_i\right\}_{t-h}^{P}\right) \qquad \text{(same as c above)}$$

use root finding algorithm for Δt (e.g. MATLAB “roots”)

must check $0 \leq \Delta t \leq h$ and it must not be imaginary

Three-dimensional Collision Detection

Quick checks

Bounding spheres and AABB both work well

Point in polyhedron – convex bodies

at each time t, calculate $\{r_i\}^P$ for all points on all bodies that are candidates for collision

select one point $\{r_j\}^P$ on body j and test if it is outside or inside a facet on body i defined by external normal $\{\hat{n}\}$ and any vertex $\{r_i\}^P$ on the facet

if $\left(\{r_j\}^P - \{r_i\}^P\right)^T \{\hat{n}\} > 0$, the point is outside the polyhedron and the search across facets may be terminated for that point

if $\left(\{r_j\}^P - \{r_i\}^P\right)^T \{\hat{n}\} = 0$, the point is on the facet and may be inside the polyhedron

if $\left(\{r_j\}^P - \{r_i\}^P\right)^T \{\hat{n}\} < 0$, the point is inside the facet and may be inside the polyhedron

point $\{r_j\}^P$ must be inside of ALL facets to be inside body i

point in polygon does not always work for thin bodies (may need edge intersection)

Point in polyhedron – ray casting

at each time t, calculate $\{r_i\}^P$ for all points on all bodies that are candidates for collision

select one point $\{r_j\}^P$ on body j and test if it is inside body i

cast an infinite ray in any direction from $\{r_j\}^P$ and compute intersections with all facets on body i

if there are an even number of intersections, $\{r_j\}^P$ is not inside body i

if there are an odd number of intersections, $\{r_j\}^P$ is inside body i

works for concavities, holes and self-crossing and is independent of CW versus CCW boundary ray-facet intersection is similar to edge-facet intersection described below

point in polygon does not always work for thin bodies (may need edge intersection)

Triangular facet intersection (MATLAB code available)

vertices for facet on body i $\{r_i\}^P \quad \{r_i\}^Q \quad \{r_i\}^R$

vertices for facet on body j $\{r_j\}^P \quad \{r_j\}^Q \quad \{r_j\}^R$

edges on body i $\{s_i\}^{Q-P} = \{r_i\}^Q - \{r_i\}^P \qquad \{s_i\}^{R-Q} = \{r_i\}^R - \{r_i\}^Q \qquad \{s_i\}^{P-R} = \{r_i\}^P - \{r_i\}^R$

edges on body j $\{s_j\}^{Q-P} = \{r_j\}^Q - \{r_j\}^P \qquad \{s_j\}^{R-Q} = \{r_j\}^R - \{r_j\}^Q \qquad \{s_j\}^{P-R} = \{r_j\}^P - \{r_j\}^R$

unit normal for facet i $\{\widehat{n}_i\} = \text{unit}\left(\{s_i\}^{R-Q} \times \{s_i\}^{Q-P}\right)$

unit normal for facet j $\{\widehat{n}_j\} = \text{unit}\left(\{s_j\}^{R-Q} \times \{s_j\}^{Q-P}\right)$

unit vector for line of intersection of planes through facets $\{\widehat{u}\} = \text{unit}\left(\{\widehat{n}_i\} \times \{\widehat{n}_j\}\right)$

equation for plane through facet i $\{\widehat{n}_i\}^T \{x\} = \{\widehat{n}_i\}^T \{r_i\}^P$

equation for ray on facet j along edge Q-P from P $\{x\} = \{r_j\}^P + d_j^{Q-P} \{s_j\}^{Q-P}$

directed distance from P on body j to plane for facet i $d_j^{Q-P} = \dfrac{\{\widehat{n}_i\}^T \left(\{r_i\}^P - \{r_j\}^P\right)}{\{\widehat{n}_i\}^T \{s_j\}^{Q-P}}$

three directed distances on facet i that pierce plane for facet j

$$d_i^{Q-P} = \frac{\{\widehat{n}_j\}^T \left(\{r_j\}^P - \{r_i\}^P\right)}{\{\widehat{n}_j\}^T \{s_i\}^{Q-P}} \qquad d_i^{R-Q} = \frac{\{\widehat{n}_j\}^T \left(\{r_j\}^P - \{r_i\}^Q\right)}{\{\widehat{n}_j\}^T \{s_i\}^{R-Q}} \qquad d_i^{P-R} = \frac{\{\widehat{n}_j\}^T \left(\{r_j\}^P - \{r_i\}^R\right)}{\{\widehat{n}_j\}^T \{s_i\}^{P-R}}$$

three directed distances on facet j that pierce plane for facet i

$$d_j^{Q-P} = \frac{\{\hat{n}_i\}^T\left(\{r_i\}^P - \{r_j\}^P\right)}{\{\hat{n}_i\}^T\{s_j\}^{Q-P}} \qquad d_j^{R-Q} = \frac{\{\hat{n}_i\}^T\left(\{r_i\}^P - \{r_j\}^Q\right)}{\{\hat{n}_i\}^T\{s_j\}^{R-Q}} \qquad d_j^{P-R} = \frac{\{\hat{n}_i\}^T\left(\{r_i\}^P - \{r_j\}^R\right)}{\{\hat{n}_i\}^T\{s_j\}^{P-R}}$$

only use directed distances that are bounded $0 \le d < 1$

there must be two valid directed distances on body i and two on body j for intersection

otherwise there is no intersection

use the ray equation to find the two piercing points for facet i and the two for body j

$$\{r_i\}^A \quad \{r_i\}^B \quad \{r_j\}^A \quad \{r_j\}^B$$

directed distances from first piercing point along line of intersection

$$h_i^A = \{\hat{u}\}^T\left(\{r_i\}^A - \{r_i\}^A\right) = 0 \qquad h_i^B = \{\hat{u}\}^T\left(\{r_i\}^B - \{r_i\}^A\right)$$
$$h_j^A = \{\hat{u}\}^T\left(\{r_j\}^A - \{r_i\}^1\right) \qquad h_j^B = \{\hat{u}\}^T\left(\{r_j\}^B - \{r_i\}^A\right)$$

sort piercing points by directed distances $h_i^A \quad h_i^B \quad h_j^A \quad h_j^B$ along line of intersection

if first two piercing points belong to the same body, there is no intersection

otherwise the intersection segment runs from the second to third sorted piercing points

General facet intersection

check if edges for a given facet on body j intersect facet on body i

establish local coordinate frame with origin at the first vertex for facet on i, local x axis along first edge and local y defined by second edge (CCW vertex sequence around facet i)

local z axis will be parallel to external normal

transform vertices for facet i and facet j into local coordinates and perform all following computations in local coordinates

test where edge $\{r_j\}^Q - \{r_j\}^P$ intersects facet i

use $d_j^P = -z_j^P / \left(z_j^Q - z_j^P\right)$ and check $0 \le d_j^P \le 1$

if false, that edge does not intersect the facet and proceed to check next edge on j for same facet i

if true that edge intersects the plane of the facet at $\{r\} = \{r_j\}^P + d_j^{\ P}\left(\{r_j\}^Q - \{r_j\}^P\right)$

use planar point in polygon test if x,y components of $\{r\}$ are inside x,y components of facet i

if false, the edge intersects the plane of the facet outside its boundary

if true, the edge intersects the facet inside its boundary

must also check if edges for same facet on body i intersect same facet on body j

facet intersection does not always work if one body is completely inside the other (may need point in polygon)

Handling Collision

Physical simulators differ in the way they react to a collision. Some use the softness of the material to calculate a force, which will resolve the collision in the following time steps like it is in reality. Due to the low softness of some materials this is very CPU intensive.

Other simulators estimate the time of collision by interpolation, roll back the simulation, and calculate the exact time of collision. This may require several iterations.

After an inelastic collision, special states of sliding are often imposed. For example, an open pin-in-slot constraint is added for a vertex-edge intersection to force that vertex to slide on the edge rather than penetrate it. The Open Dynamics Engine uses this approach.