Robust Algorithm for Intersections of Triangles

Conor McCoid & Martin J. Gander
University of Geneva

Problem discussion and definitions

Suppose we have two triangles: a 'subject' triangle U, and a 'clipping' triangle V. We want to see how much of U is covered by V.

Pick a vertex of V. The vector pointing to this vertex is labelled v_0. The vectors between this vertex and the other two vertices of V are labelled v_1 and v_2, so that the positions of the three vertices of V may be summarized as the following matrix: $v_0 + \begin{bmatrix} 0 & v_1 & v_2 \end{bmatrix}$. The same may be done for U, creating u_0, u_1 and u_2.

Step 1: Change of Coordinates

Use an affine transformation that turns the clipping triangle into a right-angled triangle with perpendicular sides of length 1, called the reference triangle Y. The vertices of the transformation of U (called X) may be found by solving the system

$$\begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} = u_0 + \begin{bmatrix} 0 & u_1 & u_2 \end{bmatrix} - v_0.$$

Step 2: Edge Parametrization and Intersection

Pick an edge of Y. Find parametrizations p and q of the coordinates so that the edge lies at $p = 0$ and $0 \leq q \leq 1$ (see table below). For each vertex (x_i, y_i) of X calculate the parameters $(p_i, q_i) = (p(x_i, y_i), q(x_i, y_i))$.

If $\text{sign}(p_i) \neq \text{sign}(p_j)$ then there is an intersection between the edge connecting the i–th and j–th vertices of X and the line $p = 0$. Calculate q_0^k, the q–coordinate of this intersection.

There will be at most two intersections per edge of Y: q_0^1 and q_0^2. WLOG, $q_0^1 < q_0^2$. If $0, 1 \in [q_0^1, q_0^2]$ then the corresponding vertices of Y lie inside the triangle X. If $q_0^1, q_0^2 \in [0, 1]$ then the corresponding intersections lie on the edge of Y. These points are vertices of the polygon of intersection (see figure).

Convert these vertices back into (x, y) coordinates (see table below). Undo the affine transformation on these points by multiplying by the matrix $\begin{bmatrix} v_1 & v_2 \end{bmatrix}$.

Parameters	$y = 0$	$x = 0$	$x + y = 1$
$p(x, y)$	y	x	$1 - x - y$
$q(x, y)$	x	y	$(1 - x + y)/2$
Intersection	$(q_0, 0)$	$(0, q_0)$	$(1 - q_0, q_0)$

Repeat this step for each of the three edges.

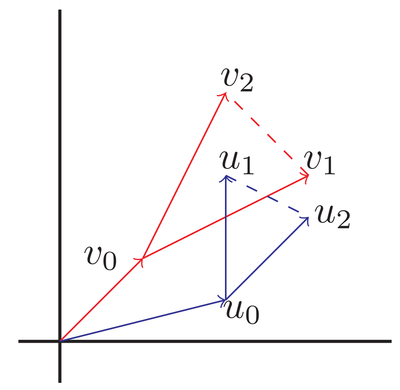
Step 3: Vertices of X in Y and Polygon of Intersection

If p_i is positive for every edge, the i–th vertex of X lies inside Y.

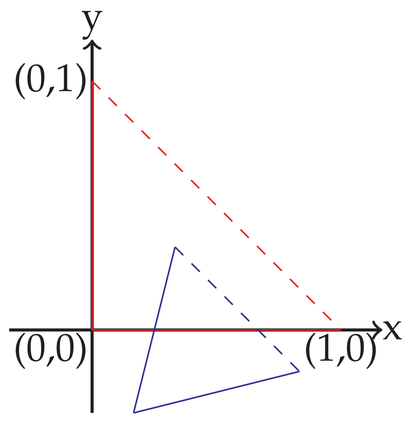
Take the convex hull of the points listed to the right to get the polygon of intersection.

- Vertices of X in Y ($\text{sign}(p_i) = 1$)
- Vertices of Y in X ($0, 1 \in [q_0^1, q_0^2]$)
- Intersections on Y ($q_0^1, q_0^2 \in [0, 1]$)

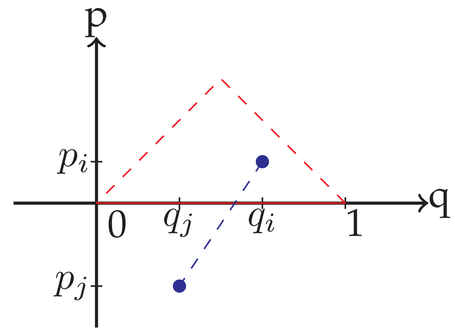
Original triangles



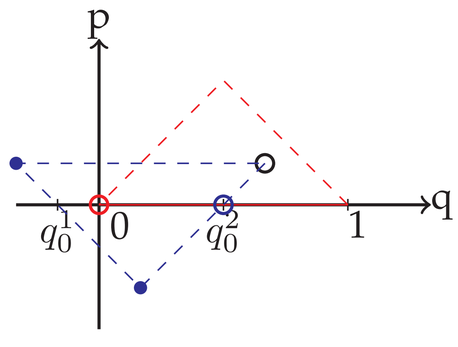
After Coord. Transf.



Edge Parameters



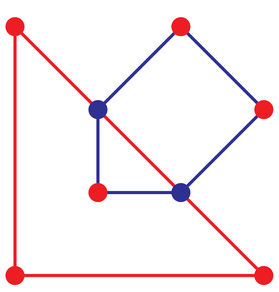
Intersection Vertices



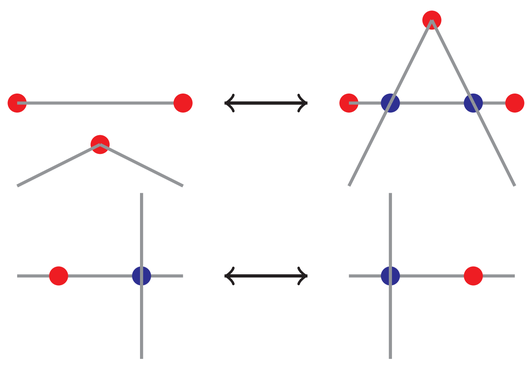
Sketch of Robustness Proof

All points of Step 3 are found using the same configuration: the transformed coordinates created in Step 1. This makes the algorithm self-consistent but asymmetric (it treats U differently than V). A symmetric algorithm may be more accurate but inconsistent.

Any possible intersection may be written as one of ten graphs. An example is provided below.



Any error caused by the algorithm may be expressed as a graph rewrite. The algorithm can have errors related to all three types of points listed under Step 3, but, because the algorithm is self-consistent, these result in only two kinds of graph rewrites: first type (top) and second type (bottom).



If the graph rewrites map the set of graphs to itself then the algorithm always gives a valid intersection. In the figure below, the ten graphs are connected by coloured edges to indicate which rewrites can be used to go from one to the other: blue for the first type and red for the second type.