A provably robust algorithm for triangle intersections

Conor McCoid

University of Geneva

12.09.20

Intersecting triangles

Goal

Transfer information between two nonmatching grids (AKA the grid transfer problem).

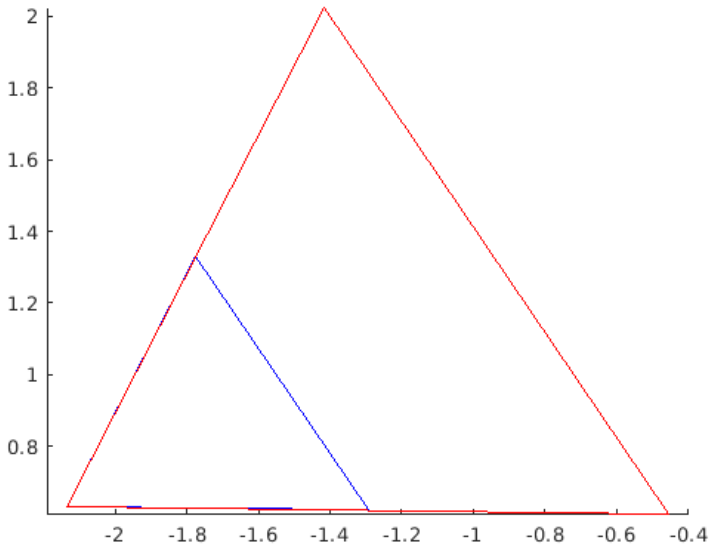
Method

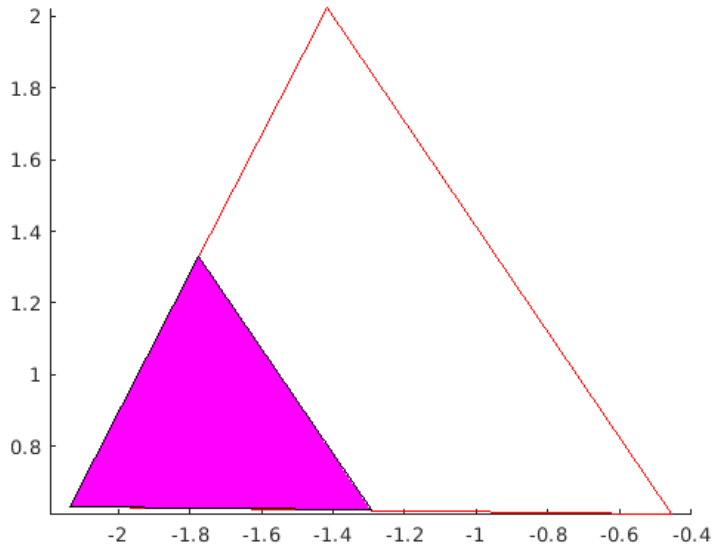
Calculate the overlap of the triangulations.

New goal

Calculate the intersection of 2D triangles robustly in floating point arithmetic.

The trouble clipping polygons





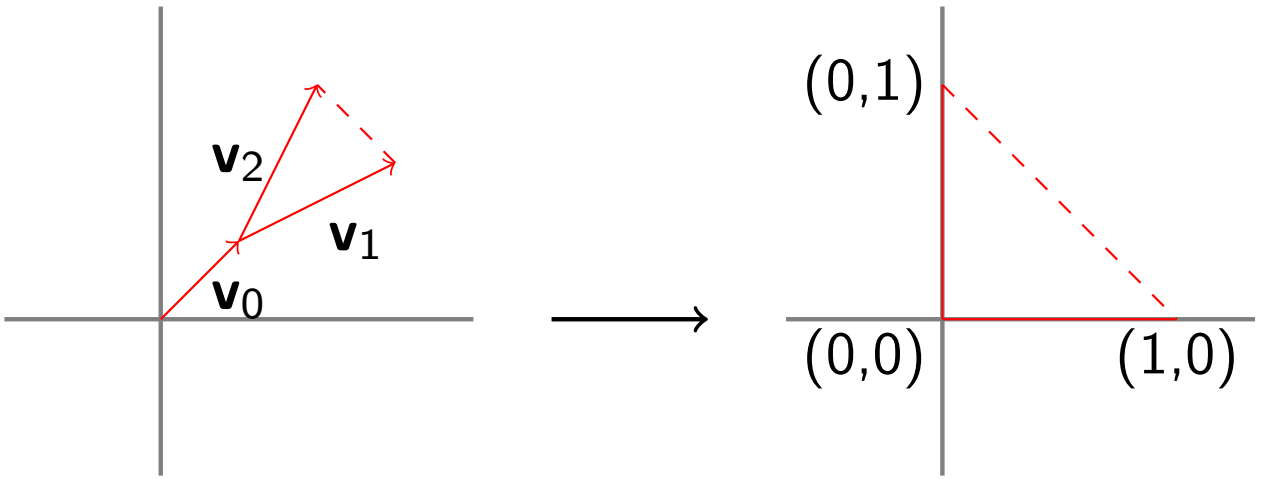
The algorithm

1. Change of coordinates $\rightarrow$ reference triangle
2. Calculate intersections between edges
3. Determine if vertices are inside or outside

1. Change of coordinates

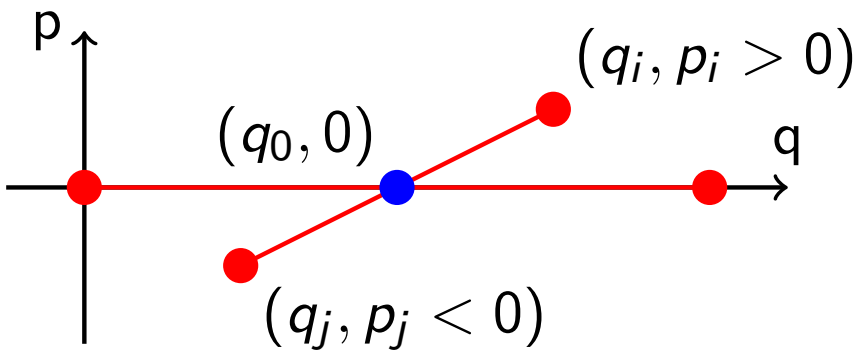
1. Choose one of the two triangles
2. Transform this triangle into a reference triangle, ie. $\{(0,0),(0,1),(1,0)\}$
3. Use the same transform on the second triangle

Call the reference triangle Y and the other triangle X



2. Edge parametrization

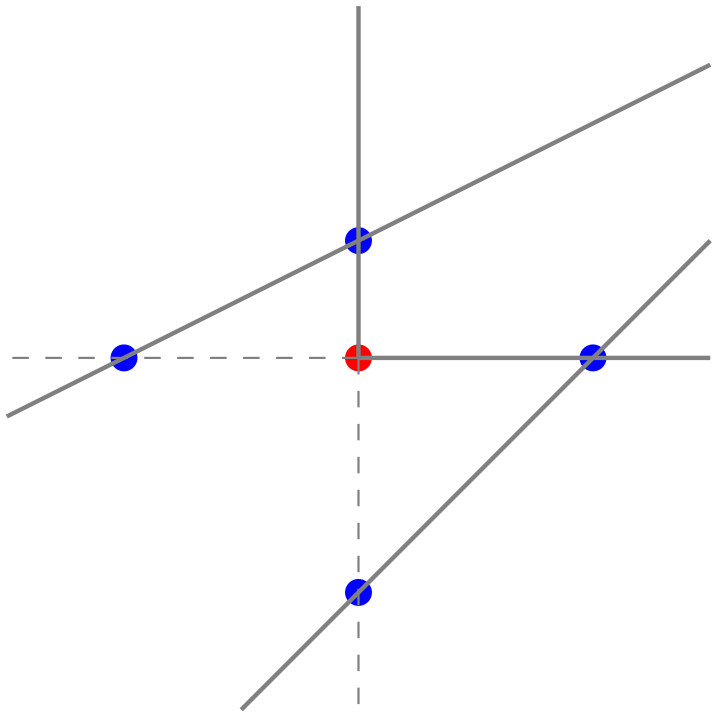
1. Choose an edge of Y, the reference triangle
2. Parametrize the coordinates (q, p) such that this edge lies on $p = 0$, $q \in [0, 1]$ and Y lies on the positive side of $p = 0$; let (q_i, p_i) be the coordinates of the i–th vertex of X
3. For the three edges of X, check if p_i and p_j have different signs; calculate the value of q_0 where this edge crosses $p = 0$
4. If $q_0 \in [0, 1]$ then $(q_0, 0)$ is an intersection between an edge of X and an edge of Y; transform $(q_0, 0)$ into the original coordinates



3. Vertices

Vertices of Y in X:

1. Choose an edge of Y and consider the two values of q_0 for this edge (call them q_0^1 and q_0^2)
2. If $0, 1 \in [q_0^1, q_0^2]$ then the corresponding vertices of Y lie in X



Vertices of X in Y:

1. Choose a vertex of X and consider the three values of p_i for this vertex (call them p_i^j where $j = 1, 2, 3$)
2. If p_i^j is positive for all j then this vertex lies in Y

Proof of robustness

1. There are only a limited number of errors that can occur with this algorithm
2. These errors can change the size and shape of the polygon of intersection, but only on the order of machine epsilon

Types of errors

X–in–Y errors: The algorithm incorrectly places a vertex of X outside/inside Y

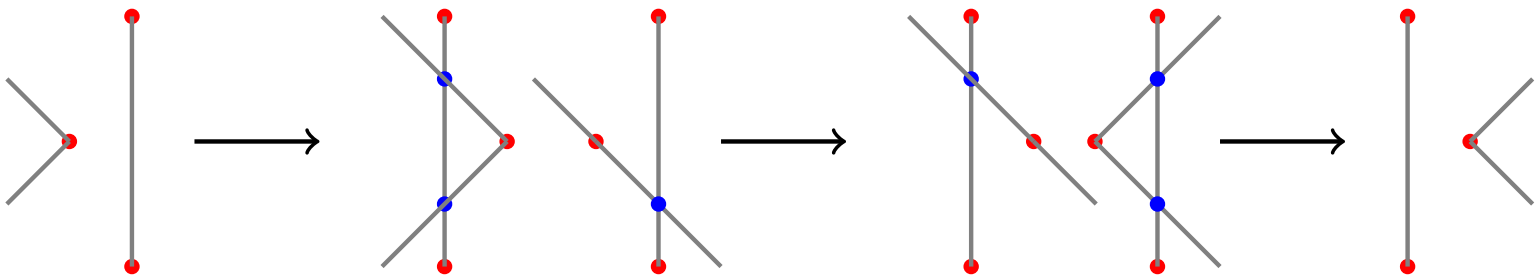
Intersection errors: The algorithm miscalculates an intersection to lie on/off an edge of Y

Y–in–X errors: The algorithm incorrectly determines a vertex of Y to be outside/inside X

X–in–Y errors

This error is caused by a sign flip of a value of p_i^j, for some i and j. As a result, one of the following occurs:

- two intersections have been created;
- an intersection has been moved;
- two intersections have been deleted.

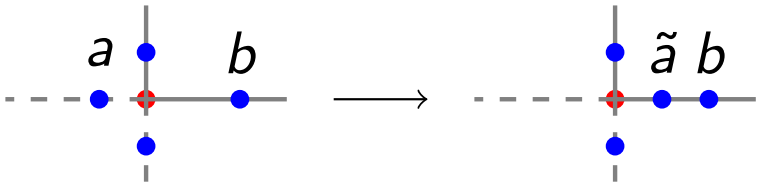


Intersection errors and Y–in–X errors

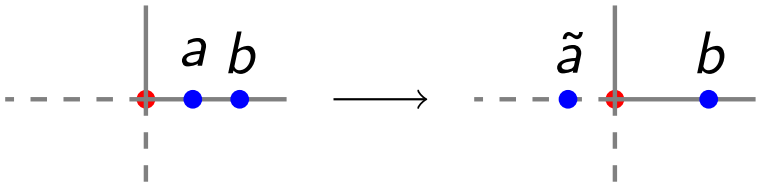
If an intersection moves over the line $q = 0$ or $q = 1$ then either:

- there is no change in the shape or size of the polygon of intersection or;
- the change is equivalent to an X–in–Y error where an intersection has been moved.

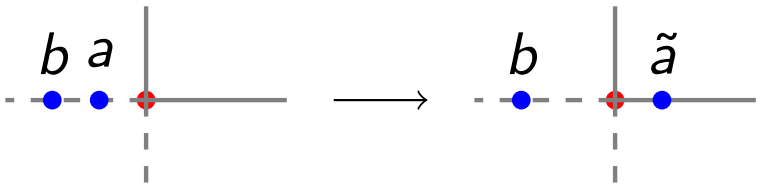
Y–in–X errors occur only as a result of the other types of errors. They help ensure consistency.



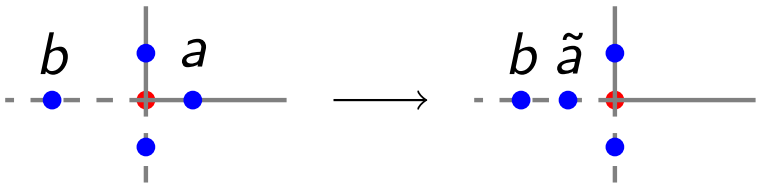
(a) New point of intersection.



(b) Loss of intersection, new Y-in-X point.



(c) New intersection and Y-in-X point.



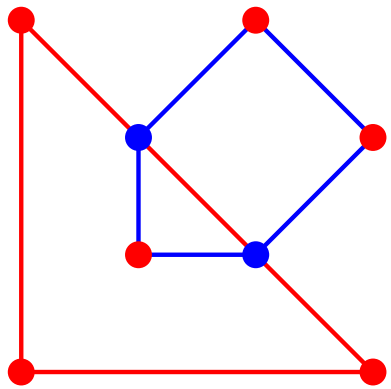
(d) Loss of intersection.

Triangle graphs

Triangle-triangle intersections can be represented by graphs with the following conditions:

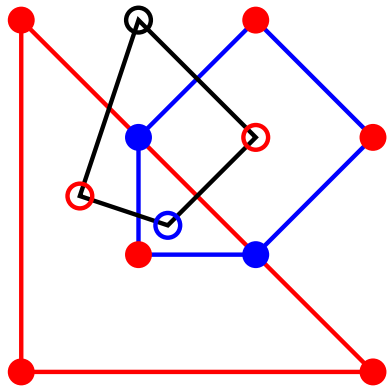
Vertex condition: the graph has exactly 6 red vertices with two neighbours;

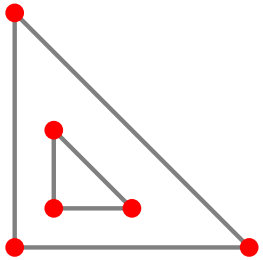
Intersection condition: the graph has 0, 2, 4 or 6 blue vertices with four neighbours;



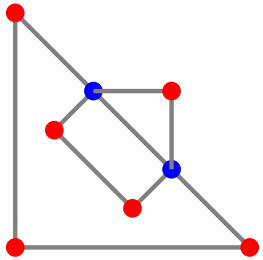
Triangle condition: there are exactly two cycles each containing three red vertices not shared between these cycles such that the cycles meet at the blue vertices;

Sheltered polygon property: no edge between two blue vertices touches the exterior of the graph.

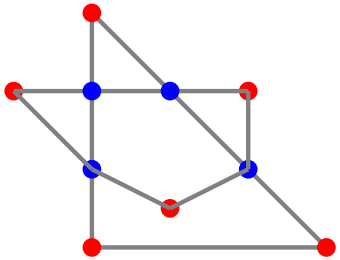




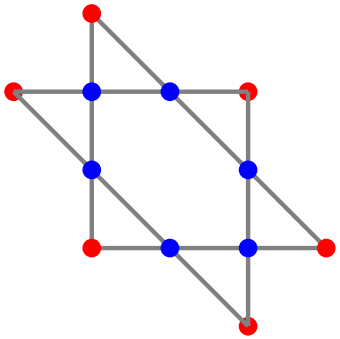
(a) A



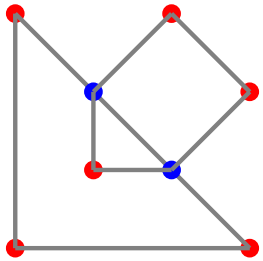
(b) B



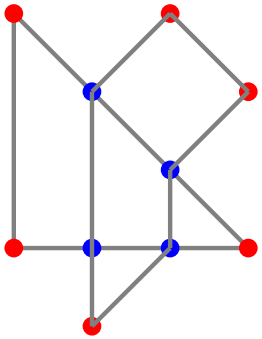
(c) C



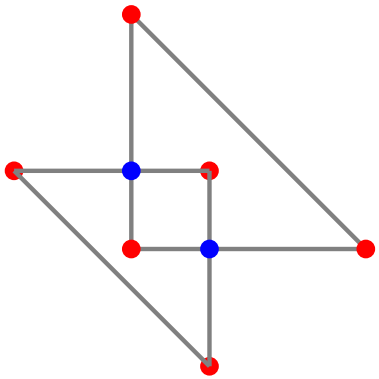
(d) D



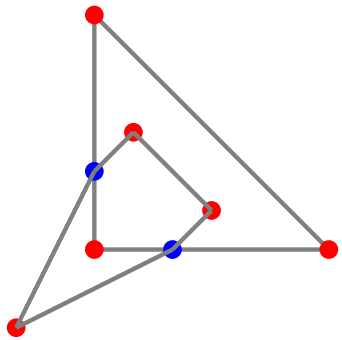
(e) E



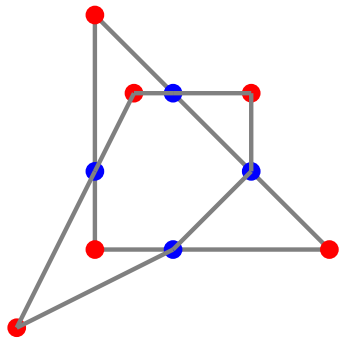
(f) F



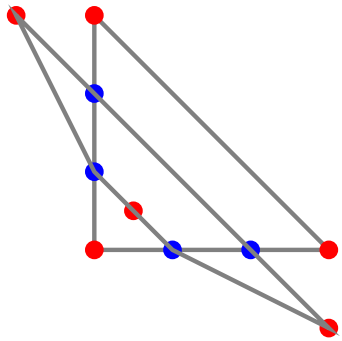
(a) B*



(b) G



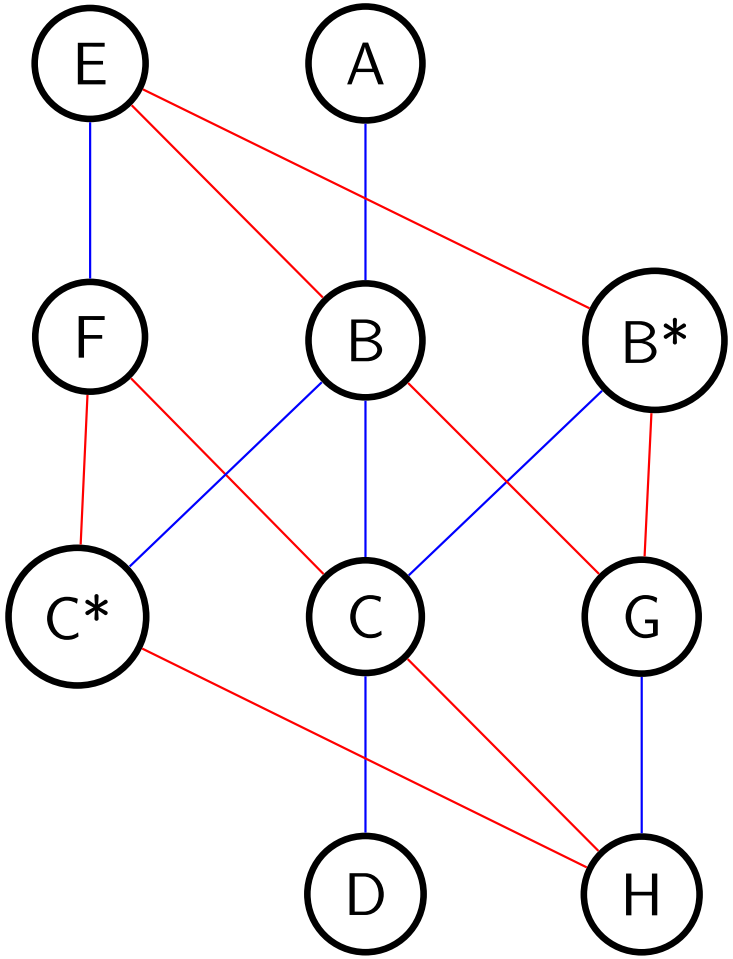
(c) H



(d) C*

Figure: The graphs arising from all possible triangle-triangle intersections, degenerate cases removed.

Connectivity



The proof

- An error can only transform a triangle-triangle intersection into another triangle-triangle intersection. There is no deletion or creation of large, unexpected areas.
- Since all steps of the algorithm can only introduce errors on the order of machine epsilon, the resulting polygon of intersection will have an error at most of the same order.

Degenerate cases

Vertex of X coincident with edge of Y

Vertex of Y coincident with edge of X

Vertices coincident