Unsigned distance function

# Git repos

<https://github.com/paragpathak2006/unsigned_distance_function>

<https://github.com/paragpathak2006/CudaRuntime1>

# Mesh

Define a Mesh that has vertex Points and Triangular faces as

# Pointwise distance

Q is query point. is maximum truncated distance.

# Facewise distance

Q is query point. is maximum truncated distance.

Ref: [Distance Between Point and Triangle in 3D (geometrictools.com)](https://www.geometrictools.com/Documentation/DistancePoint3Triangle3.pdf)

(<https://www.geometrictools.com/Documentation/DistancePoint3Triangle3.pdf>)

Let Face Triangle be defined as

Face Triangle to Point distance can be found using the formula

# Brute force approach

Go over all the points and faces to find the minimum possible distance between target point and mesh points.

# Spatial indexing approach

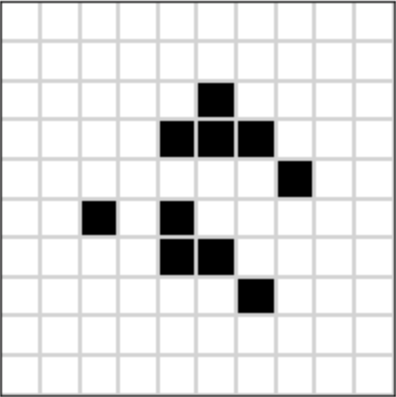


Figure 1 Space map showing spatial indexing of points

Recommended approach of indexing is using octree, but in our case were going to implement a simple space map to spatially Index the mesh points. After indexing, find all the points and faces in a sphere to minimize list of candidate Points to search.

# Thrust library function

double min\_dist\_calculation2(

const HPoint& Px,

const Point& target,

const double& beta2

){

DPoint points = Px;

Dvec distances(Px.size());

// apply the transformation

thrust::transform(\_ITER\_(points), distances.begin(), dist2\_point(target));

return thrust::reduce(\_ITER\_(distances), beta2, min\_dist());

}

Where,

typedef thrust::host\_vector<double> Hvec;

typedef thrust::host\_vector<Point> HPoint;

typedef thrust::device\_vector<double> Dvec;

typedef thrust::device\_vector<Point> DPoint;