Input: SA: HalfEdgeMesh

Output SB: HalfEdgeMesh

(Besides **vertex**, **halfedge** and **cell**, HalfEdgeMesh contains an **edge** class, whose related function would be used in the algorithm)

Procedure FacePoints(SA, SB)

For all h\_idx∈[0, SA.NumHalfEdge()) do

h := SA.HalfEdge(h\_idx)

m := h.Cell().NumVertices()

v := h.FromVertex() // half edge from vertex

i := SA.NumVertices() + h.cellIndex() // new face point vertexID

SB.Vertex(i) += v/m

end for

end procedure

Procedure EdgePoints(SA, SB)

For all h\_idx∈[0, SA.NumHalfEdge()) do

h := SA.HalfEdge(h\_idx)

j := SA.NumVertices() + SA.NumCells() + h.EdgeIndex() // new edge point VertexID

if h.IsBoundary()

SB.Vertex(j) += (h.FromVertex() + h.ToVertex()) / 2.0

else

v := h.FromVertex() // half edge from vertex

i := SA.NumVertices() + h.cellIndex() // new face point vertexID

SB.Vertex(j) += (v + SB.Vertex(i)) / 4.0

end if

end for

end procedure

Procedure VertexPoints(SA, SB)

For all h\_idx∈[0, SA.NumHalfEdge()) do

h := SA.HalfEdge(h\_idx)

v\_idx := h.FromVertexIndex() // half edge from vertex ID

if h.IsBoundary()

SB.Vertex(v\_idx):= SA.Vertex(v\_idx) // old vertex point

else

n := VA[v\_idx].NumHalfEdgeFrom() // valence of the old vertex point

i := SA.NumVertices() + h.cellIndex() // new face point vertexID

j := SA.NumVertices() + SA.NumCells() + h.EdgeIndex() // new edge point VertexID

SB.Vertex(j) += (4 \* SB.Vertex(j) – SB.Vertex(i) + (n-3) \* SA.Vertex(v\_idx)) / (n^2)

end if

end for

end procedure

Procedure RefineHalfEdges(SA, SB)

For all h\_idx∈[0, SA.NumHalfEdge()) do

h := SA.HalfEdge(h\_idx)

// apply halfedge’s twin rule

SB.HalfEdge(4\*h\_idx + 0).NextHalfEdgeAroundEdge()

:= SB.HalfEdge(4 \* h.NextHalfEdgeAroundEdge().NextHalfEdgeInCell() + 3))

SB.HalfEdge(4\*h\_idx + 1).NextHalfEdgeAroundEdge()

:= SB.HalfEdge(4 \* h.NextHalfEdgeInCell() + 2)

SB.HalfEdge(4\*h\_idx + 2).NextHalfEdgeAroundEdge()

:= SB.HalfEdge(4 \* h.PrevHalfEdgeInCell() + 1)

SB.HalfEdge(4\*h\_idx + 3).NextHalfEdgeAroundEdge()

:= SB.HalfEdge(4 \* h.PrevHalfEdgeInCell().NextHalfEdgeAroundEdge() + 0)

// apply halfedge’s next rule

SB.HalfEdge(4\*h\_idx + 0).NextHalfEdgeInCell() := SB.HalfEdge(4 \* h\_idx + 1)

SB.HalfEdge(4\*h\_idx + 1).NextHalfEdgeInCell() := SB.HalfEdge(4 \* h\_idx + 2)

SB.HalfEdge(4\*h\_idx + 2).NextHalfEdgeInCell() := SB.HalfEdge(4 \* h\_idx + 3)

SB.HalfEdge(4\*h\_idx + 3).NextHalfEdgeInCell() := SB.HalfEdge(4 \* h\_idx + 0)

// apply halfedge’s previous rule

SB.HalfEdge(4\*h\_idx + 0).PrevHalfEdgeInCell() := SB.HalfEdge(4 \* h\_idx + 3)

SB.HalfEdge(4\*h\_idx + 1).PrevHalfEdgeInCell() := SB.HalfEdge(4 \* h\_idx + 0)

SB.HalfEdge(4\*h\_idx + 2).PrevHalfEdgeInCell() := SB.HalfEdge(4 \* h\_idx + 1)

SB.HalfEdge(4\*h\_idx + 3).PrevHalfEdgeInCell() := SB.HalfEdge(4 \* h\_idx + 2)

// apply halfedge’s vertex rule

h\_prev = h.PrevHalfEdgeInCell()

SB.HalfEdge(4\*h\_idx + 0).FromVertex() := SB.vertex(h.FromVertex())

SB.HalfEdge(4\*h\_idx + 1).FromVertex() :=

SB.vertex(SB.Vertices(SA.NumVertices() + SA.NumCells() + h.EdgeIndex())

SB.HalfEdge(4\*h\_idx + 2).FromVertex() :=

SB.vertex(SB.Vertices(SA.NumVertices() + h.CellIndex())

SB.HalfEdge(4\*h\_idx + 2).FromVertex() :=

SB.vertex(SB.Vertices(SA.NumVertices() + SA.NumCells() + h\_prev.EdgeIndex())

// apply halfedge’s edge rule

h\_prev\_idx = h\_prev.Index()

SB.HalfEdge(4\*h\_idx + 0).EdgeIndex() :=

2 \* h.EdgeIdx() if h\_idx > h.NextHalfEdgeAroundEdge().Index()

2 \* h.EdgeIdx() + 1 otherwise

SB.HalfEdge(4\*h\_idx + 1).EdgeIndex() := 2 \* SA.NumEdges() + h\_idx

SB.HalfEdge(4\*h\_idx + 2).EdgeIndex() := 2 \* SA.NumEdges() + h\_prev\_idx

SB.HalfEdge(4\*h\_idx + 0).EdgeIndex() :=

2 \* h\_prev.EdgeIdx()

if h\_prev\_idx > h\_prev.NextHalfEdgeAroundEdge().Index()

2 \* h\_prev.EdgeIdx() + 1 otherwise

// apply halfedge’s face rule

SB.HalfEdge(4\*h\_idx + 0).CellIndex() := h.CellIndex()

SB.HalfEdge(4\*h\_idx + 1).CellIndex() := h.CellIndex()

SB.HalfEdge(4\*h\_idx + 2).CellIndex() := h.CellIndex()

SB.HalfEdge(4\*h\_idx + 3).CellIndex() := h.CellIndex()

end for

end procedure

Procedure Main(SA, SB)

FacePoints(SA, SB)

EdgePoints(SA, SB)

VertexPoints(SA, SB)

RefineHalfEdges(SA, SB)

return SB

end procedure