Direct Triangle Meshes Remeshing using Stellar Operators

Aldo Zang, Fabian Prada

VISGRAF - IMPA
Introduction

Remeshing: Mesh quality improvement

- Sampling density.
- Regularity.
- Size.
- Orientation.
- Alignment.
- Shape.

Remeshing Algorithms

- Variational Remeshing.
- **Incremental Remeshing.**
### Problem Statement

Propose a remeshing strategy based on stellar operators to obtain a mesh that satisfactorially meets the following criteria:

- **Uniformity:** Equilateral Aspect Triangles
- **Regularity:** Valence 6 vertices at interior and valence 4 at boundary

### Constraints

- Preserve Geometry and Features.
- Maintain Resolution.
Papers used for our approach

- **A remeshing approach to multiresolution modeling.** Mario Botsch, Leif Kobbelt.

- **Multiresolution shape deformations for meshes with dynamic vertex connectivity.** Leif Kobbelt, Thilo Bareuther, Hans-Peter Seidel.

- **Stellar mesh simplification using probabilistic optimization.** Antônio Wilson Vieira et al.

- **Diffusion tensor weighted harmonic fields for feature classification.** Shengfa Wang et al.

- **Hierarchical feature subspace for structure-preserving deformation.** Submitted to GMP 2012
Remeshing pipeline

Remeshing algorithm

1. Get a edge target length $l$

2. Split all edges that are longer than $3l$ at their midpoint

3. Collapse all edges shorter than $5l$ into their midpoint (or collapse over the vertex with higher valence)

4. Flip edges in order to minimize the deviation from valence 6 (or 4 on boundaries)

5. Relocate vertices on the surface by tangential smoothing

6. Repeat steps (2)-(5) until satisfy the stop criteria (good edges ratio)

7. Apply area based tangential smoothing to equalize triangles areas

Aldo Zang, Fabian Prada Direct Triangle Meshes Remeshing using Stellar Operators
Remeshing pipeline

Remeshing algorithm

1. Get an edge target length $l$
2. Split all edges that are longer than $\frac{4}{3}l$ at their midpoint
Remeshing pipeline

Remeshing algorithm

1. Get a edge target length $l$
2. Split all edges that are longer than $\frac{4}{3}l$ at their midpoint
3. Collapse all edges shorter than $\frac{4}{5}l$ into their midpoint (or collapse over the vertex with higher valence)
4. Flip edges in order to minimize the deviation from valence
5. Relocate vertices on the surface by tangential smoothing
6. Apply area based tangential smoothing to equalize triangles areas

Aldo Zang, Fabian Prada
Direct Triangle Meshes Remeshing using Stellar Operators
Remeshing pipeline

Remeshing algorithm

1. Get a edge target length $l$
2. Split all edges that are longer than $\frac{4}{3}l$ at their midpoint
3. Collapse all edges shorter than $\frac{4}{5}l$ into their midpoint (or collapse over the vertex with higher valence)
4. Flip edges in order to minimize the deviation from valence 6 (or 4 on boundaries)

Aldo Zang, Fabian Prada
Direct Triangle Meshes Remeshing using Stellar Operators
### Remeshing algorithm

1. Get a edge target length $l$
2. Split all edges that are longer than $\frac{4}{3}l$ at their midpoint
3. Collapse all edges shorter than $\frac{4}{5}l$ into their midpoint (or collapse over the vertex with higher valence)
4. Flip edges in order to minimize the deviation from valence 6 (or 4 on boundaries)
5. Relocate vertices on the surface by tangential smoothing

---

Aldo Zang, Fabian Prada

Direct Triangle Meshes Remeshing using Stellar Operators
Remeshing pipeline

Remeshing algorithm

1. Get a edge target length $l$
2. Split all edges that are longer than $\frac{4}{3}l$ at their midpoint
3. Collapse all edges shorter than $\frac{4}{5}l$ into their midpoint (or collapse over the vertex with higher valence)
4. Flip edges in order to minimize the deviation from valence 6 (or 4 on boundaries)
5. Relocate vertices on the surface by tangential smoothing
6. Repeat steps (2)-(5) until satisfy the stop criteria (good edges ratio)
Remeshing pipeline

Remeshing algorithm

1. Get a edge target length $l$
2. Split all edges that are longer than $\frac{4}{3}l$ at their midpoint
3. Collapse all edges shorter than $\frac{4}{5}l$ into their midpoint (or collapse over the vertex with higher valence)
4. Flip edges in order to minimize the deviation from valence 6 (or 4 on boundaries)
5. Relocate vertices on the surface by tangential smoothing
6. Repeat steps (2)-(5) until satisfy the stop criteria (good edges ratio)
7. Apply area based tangential smoothing to equalize triangles areas
1- Get the edge target length $l$

We compute some statistics for the mesh and set the target length as:

$$l = \text{Mean} (\text{edges length}) - \lambda \text{Deviation} (\text{edges length})$$

with $\lambda \in [0, 1]$. 
2- Split edges

All edges which are longer than $\frac{4}{3}l$ are split by inserting a new vertex at its midpoint. The two adjacent triangles are bisected accordingly. The upper and lower bound on the edges length are only compatible if $\epsilon_{max} > 2\epsilon_{min}$.

Figure: Left: original edge $(u, v)$; Right: Split of $(u, v)$ inserting vertex $w$. 

Aldo Zang, Fabian Prada
3- Collapse edges

All edges which are shorter than $\frac{4}{5}l$ are removed by collapsing the two-end vertices. We collapse that-end vertex with lower valence into the one with higher. This prevent accumulation of edges collapses.

Figure: **Left:** original mesh; **Center:** Accumulation of edges collapses; **Right:** Collapses over the higher valence vertex.
4- Flip edges

We perform edge-flipping in order to regularize the connectivity. For every two neighboring triangles $\Delta(A, B, C)$ and $\Delta(C, B, D)$ we maximize the number of vertices with valence six by flipping the diagonal $\overline{BC}$ if the total valence excess is reduced.

$$V(e) = \sum_{p \in A, B, C, D} (\text{valence}(p) - 6)^2$$
Remeshing algorithm: step by step

4- Flip edges

Figure: **Left:** Initial valence condition $V(e) = 2^2 + 1 = 5$; **Right:** Valence condition after flipping edge $V(e) = 1 + 1 + 1 = 3$. 
**Edge-Flip**

*Edge-Flip:* This operation consists in transforming a two-face cluster into another two-face cluster by swapping its common edge.

Figure: **Left:** input mesh; **Right:** result after flipping \((u, v)\) to \((s, t)\).
**Edge-Split operator**

**Edge-Split:** This operation consists in transforming a two-face cluster into a four-face cluster by inserting a vertex in the interior edge of the cluster.

![Split of the edge (u, v) by inserting a midpoint vertex w.](image)

**Figure:** Split of the edge \((u, v)\) by inserting a midpoint vertex \(w\).
**Edge-Flip operator**

**Lemma (Flip Condition):** Let $S$ a combinatorial 2-manifold. The *flip* of an interior edge that replaces $e = (u, v) \in S$ by $(s, t)$ preserves the topology of $S$ if and only if $(s, t) \notin S$.

**Figure:** The edge $(u, v)$ do not satisfies the flip condition because the new edge $(s, t)$ already exists in the mesh.
Edge-Collapse: This operator consists in removing an edge $e = (u, v) \in S$, identifying its vertices to a unique vertex $\bar{v}$. From a combinatorial viewpoint, this operator will remove 1 vertex, 3 edges, and 2 faces from the original mesh, thus preserving its Euler characteristic.

**Figure:** The edge $(u, v)$ is collapsed removing the vertex $u$ from the mesh.
Stellar operators theory

Edge-Collapse

**Lemma (Collapse Condition):** Let $S$ be a combinatorial 2-manifold. The collapse of an edge $e = (u, v) \in S$ preserves the topology of $S$ if the following conditions are satisfied:

- $\text{link}(u) \cap \text{link}(v) = \text{link}(e)$;
- if $u$ and $v$ are both boundary vertices, $e$ is a boundary edge;
- $S$ has more than 4 vertices if neither $u$ nor $v$ are boundary vertices, or $S$ has more than 3 vertices if either $u$ or $v$ are boundary vertices.
Stellar operators theory

**Edge-Collapse condition**

**Figure:** Left: \((u, v)\) satisfies the edge-collapse condition; Right: \((u, v)\) do not satisfies the edge-collapse condition because \(s \in \text{link}(u) \cap \text{link}(v)\) and \(s \notin \text{link}((u, v))\);

**Figure:** \((s, t)\) do not satisfies the edge-collapse condition because \((s, t)\) is interior edge but \(s\) and \(t\) are boundary vertices.
**Edge-Weld operator**

**Edge-Weld:** This operation consists in transforming a four-face cluster into a two-face cluster by removing its central vertex.

**Corollary:** Given a combinatorial 2-manifold $S$, and a interior vertex $v \in S$ with valence 4. The removal of the vertex $v$ by the Edge-Weld operation (along $(u, w)$) preserves the topology of $S$ if and only if there is no edge in $S$ connecting $u$ to $w$.

![Diagram](image)

**Figure:** Edge-weld by removing midpoint vertex $w$. 
Stellar operators theory

Edge-Collapse using basic stellar operators (edge-flip and edge-weld)

Figure: Edge \((u, v)\) is collapsed using 2 edges flips \((u, s)\) and \((u, t)\) and one edge-weld for edge \((u, v)\).
Stellar operators of the A48 library

Basic operators

- `flip(halfedge_type *h);`
- `face_weld(halfedge_type *h1, halfedge_type *h2, halfedge_type *h3);`
- `edge_weld(halfedge_type *h1, halfedge_type *h2);`
- `edge_split(halfedge_type *h);`
- `face_split(face_type *f);`
New operator and condition tests

<table>
<thead>
<tr>
<th>void edge_collapse( halfedge_type *h );</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool can_edgeCollapse( halfedge_type *h );</td>
</tr>
<tr>
<td>bool can_edge_flip( halfedge_type *h );</td>
</tr>
</tbody>
</table>
**Halfedges Set data structure**

```cpp
class RmSetComp {
public:
    bool operator() ( halfedge_type *hi, halfedge_type *hj ) const {
        if ( hi->l < hj->l )
            return true;
        else if ( hj->l < hi->l )
            return false;
        else return hi < hj;
    }
};

typedef std::set<halfedge_type *, RmSetComp> HalfedgeSet;
```
# Halfedges Set insert and remove functions

```c
void remove_from_HSet( HalfedgeSet &RmSet, halfedge_type *he ) {
    if( he > he->opposite() )
        RmSet.erase( he );
    else
        RmSet.erase( he->opposite() );
}

void insert_to_HSet( HalfedgeSet &RmSet, halfedge_type *he ) {
    if( hcurr > hcurr->opposite() )
        RmSet.insert( hcurr );
    else
        RmSet.insert( hcurr->opposite() );
}
```
Vertex Reallocation

**Step 1: Unconstrained Vertex Displacement**

New position for each vertex is calculated as a weighted sum of its neighbours positions:

\[
\hat{p}_i \leftarrow \frac{1}{\sum_{p_j \in N(p_i)} w_j} \left( \sum_{p_j \in N(p_i)} w_j p_j \right)
\]

**Neighbours Weights**

- **Uniform**: \( w_j = 1 \).
- **One Ring Area**: \( w_j = \sum_{f: p_j \in f} A(f) \).
Step 2: Reprojection

- Projection on the tangent plane:
  \[ p_i \leftarrow p_i + (I - n_i n_i^T)(\hat{p}_i - p_i). \]

- Projection on the lowest curvature direction:
  \[ p_i \leftarrow p_i + \gamma_{\text{min}} \gamma_{\text{min}}^T(\hat{p}_i - p_i). \]
Vertex Reallocation

Lowest Curvature Direction

Discrete Curvature Tensor Estimation:

\[ T(p) = \frac{1}{|B|} \sum_{e} \beta(e)|e \cap B| ee^T. \]

Lowest curvature direction is the eigenvector associated to the largest eigenvalue of \( T(p) \).
Min Curvatures Estimation: 1 Ring
Min Curvatures Estimation: 3 Ring
Remeshing Sequence: input mesh
Remeshing Sequence: 1- Split edges

Aldo Zang, Fabian Prada
Direct Triangle Meshes Remeshing using Stellar Operators
Remeshing Sequence: 2- Collapse edges

Aldo Zang, Fabian Prada
Direct Triangle Meshes Remeshing using Stellar Operators
Remeshing Sequence: 3- Flip edges
Remeshing Sequence: 4- Vertex reallocation
Remeshing iterations: input mesh
Remeshing iterations: Iteration 1

Aldo Zang, Fabian Prada

Direct Triangle Meshes Remeshing using Stellar Operators
Remeshing iterations: Iteration 2
Remeshing iterations: Iteration 5

Aldo Zang, Fabian Prada

Direct Triangle Meshes Remeshing using Stellar Operators
Some results

**Figure:** Input mesh.
Some results

**Figure:** Random remeshing using tangential area smoothing.
Some results

Figure: Sequential remeshing using tangential area smoothing.
Some results

Figure: Input mesh
Some results

Figure: Sequential remeshing using tangential area smoothing.
Some results

Figure: Input mesh.

Aldo Zang, Fabian Prada
Direct Triangle Meshes Remeshing using Stellar Operators
Figure: Sequential remeshing using tangential area smoothing.
Some results

Figure: Input mesh.
Some results

Figure: Sequential remeshing using tangential area smoothing.
Some results

Figure: Input mesh.
Some results

Figure: Sequential remeshing using tangential area smoothing.
Difusion tensor

\[ T(v_i) = \sum_{t_j \in N_t(v_i)} \mu_j n_t n_{t_j}^T \]

where \( t_j \) is a triangle, \( N_t(v_i) \) denote the set of neighboring triangles of \( v_i \), \( n_{t_j} \) is the normal of triangle \( t_j \), and \( \mu_j \) is the weight coefficient (we use \( \mu = 1 \)).
### Difusion tensor

For each vertex of the mesh, \( \lambda_1, \lambda_2, \lambda_3 \geq 0 \) are eigenvalues of the corresponding structure tensor, then the feature analysis is documented as:

- **Face**: if \( \lambda_1 > 0.1, \lambda_2 < 0.02 \)
- **Corner**: if \( \lambda_3 > 0.1 \)
- **Strong**: if \( \lambda_2 > 0.1, \lambda_3 < 0.02 \)
- **Weak**: if \( \lambda_1 > 0.1, 0.1 \geq \lambda_1 \geq 0.02, \lambda_3 < 0.02 \)

The eigenvector corresponding to \( \lambda_3 \) is the difusion direction, used to define the neighboring vertex coincidence (NVC) condition.
Features guided remeshing

Features detection
Features guided remeshing

Remeshing without using features constraints
Features guided remeshing

Remeshing using features constraints
THANKS!