Subdivision Surfaces
or
How to Generate a Smooth Mesh?

Subdivision Curves and Surfaces

Subdivision – given polyline(2D)/mesh(3D) recursively modify & add vertices to achieve smooth curve/surface

• Each iteration generates smoother + more refined mesh
Corner Cutting
Corner Cutting

Corner Cutting
Corner Cutting
Corner Cutting

- control point
- limit curve
- control polygon

The 4-point scheme
The 4-point scheme

The 4-point scheme
The 4-point scheme
The 4-point scheme
The 4-point scheme

The 4-point scheme
The 4-point scheme
The 4-point scheme

Subdivision curves

Non interpolatory subdivision schemes
- Corner Cutting

Interpolatory subdivision schemes
- The 4-point scheme
Basic concepts of Subdivision

Subdivision curve – limit of recursive subdivision applied to given polygon

Each iteration
- Increase number of vertices (approximately) * 2

Initial polygon - control polygon

Central questions:
- Convergence: Given a subdivision operator and a control polygon, does the subdivision process converge?
- Smoothness: Does subdivision converge to smooth curve?

Subdivision schemes for surfaces

Each iteration
- Subdivision refines control net (mesh)
- Increase number of vertices (approximately) * 4

Mesh vertices converge to limit surface

Every subdivision method has:
- Method to generate net topology
- Rules to calculate location of new vertices
**Triangular subdivision**

*Defined for triangular meshes (control nets)*

Every face replaced by 4 new triangular faces

Two kinds of new vertices:
- Green vertices are associated with old edges
- Yellow vertices are associated with old vertices

---

**Loop’s scheme**

New vertex is weighted average of old vertices

List of weights called subdivision mask or stencil

- Rule for new yellow vertices
  
  \[
  w_n = \frac{64n}{40 - (3 + 2\cos(2\pi/n))^2} - n
  \]
The original control net

After 1st iteration
Loop Scheme

Loop Limit Surface

- Limit surfaces of Loop’s subdivision is $C^2$ almost everywhere
Butterfly Scheme

- Interpolatory scheme
- New yellow vertices inherit location of old vertices
- New green vertices calculated by following stencil:

The original control net
After 3rd iteration

Butterfly Scheme
Butterfly limit surface

Limit surfaces of Butterfly subdivision are $C^1$, but do not have second derivative

Comparison
Boundaries & Features: Loop

**Boundary vertices** – depend **ONLY on other boundary vertices**

**Corner vertices left in place**
- Sometimes modified rules for corner neighbors

**Treat features (creases) like boundaries**

---

Scheme Zoo

**More schemes:**
- Catmul-Clark
- Kobbelt
- Duo-Sabin
- …

**Proving scheme works:**
- Convergence
- Degree of continuity
- Affine invariance
Affine Invariance

- Coefficients of masks sum to 1 – weighted average

Coefficients of masks must sum to 1
Analysis of Subdivision

**Test:**
- Convergence
- Smoothness

**Goals:**
- Help to choose the rules
- Ensure that all surfaces have desired properties

**Plan**
- Define subdivision surfaces
- Relate properties to coefficients

Subdivision Matrix

*Relate control on finer level to coarser level*

**Useful for**
- Analysis of properties
  - smoothness
  - affine invariance
- Formulas for normals
- Explicit evaluation of surfaces at arbitrary points of the domain
Controls of K-Sided Patch

Simplest case - consider only 1-ring

Subdivision Matrix

<table>
<thead>
<tr>
<th></th>
<th>7/16</th>
<th>3/16</th>
<th>3/16</th>
<th>3/16</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/8</td>
<td>3/8</td>
<td>1/8</td>
<td>1/8</td>
<td></td>
</tr>
<tr>
<td>3/8</td>
<td>1/8</td>
<td>3/8</td>
<td>1/8</td>
<td></td>
</tr>
<tr>
<td>3/8</td>
<td>1/8</td>
<td>1/8</td>
<td>3/8</td>
<td></td>
</tr>
</tbody>
</table>
Subdivision Matrix

\[
\begin{pmatrix}
7/16 & 3/16 & 3/16 & 3/16 \\
3/8 & 3/8 & 1/8 & 1/8 \\
3/8 & 1/8 & 3/8 & 1/8 \\
3/8 & 1/8 & 1/8 & 3/8
\end{pmatrix}
\]
Subdivision Matrix

\[
\begin{array}{cccc}
7/16 & 3/16 & 3/16 & 3/16 \\
3/8 & 3/8 & 1/8 & 1/8 \\
3/8 & 1/8 & 3/8 & 1/8 \\
3/8 & 1/8 & 1/8 & 3/8 \\
\end{array}
\]

Eigen Values

\[
\begin{array}{cccc}
7/16 & 3/16 & 3/16 & 3/16 \\
3/8 & 3/8 & 1/8 & 1/8 \\
3/8 & 1/8 & 3/8 & 1/8 \\
3/8 & 1/8 & 1/8 & 3/8 \\
\end{array}
\]

Eigenvalues

\[
\begin{array}{c}
1 \\
0.25 \\
0.25 \\
0.0625 \\
\end{array}
\]
Eigen Decomposition

Diagonalize subdivision matrix

- eigenvectors
- eigenvalues
- : vector of points in a neighborhood

\( (N+1) \)-vector of 3D points

Eigenvectors

“Good” case:
- \( \lambda_0 = 1 \) & \( |\lambda_i| < 1, i = 1, \ldots, n - 1 \)

\[
\begin{align*}
c^e_w &= \delta_{0-0} + \delta_{1-1} + \delta_{2-2} + \delta_{3-3} + SSS \\
\end{align*}
\]

- can make \( \delta_{0-H} \) zero by moving control points (by affine invariance)
Subdominant Eigenvectors

Next higher order terms

- assume
  \[ \left( I - \lambda I \right)_Q > Q \cdot 0 \]
- move control points so that
  \[ \frac{f}{e} C^e \sim I \times_{\infty_f} f \times_{\infty_Q} \frac{1}{Q} \times_{\infty_Q} \frac{1}{Q} \]

\[ \Rightarrow \quad \frac{W_e}{e} \quad \_ \_ \_ 1 \quad \xi \xi \xi \]

Subdivision Drawbacks

Not always intuitive
Can have artifacts
Hard to control
Quad: Catmull-Clark & Doo-Sabin

Properties

*Works best on regular connectivity (valence 6)*

*Easy to implement*
- Efficiency is another matter
- Use Eigen-analysis for exact computation

*Local support – operations depend on local data*
- Order does not matter
- Data structure support

*Allow LOD*
- View mesh at any level as control mesh
  - Apply geometric modifications

*Continuous*
- Scheme dependent
Geri’s Game