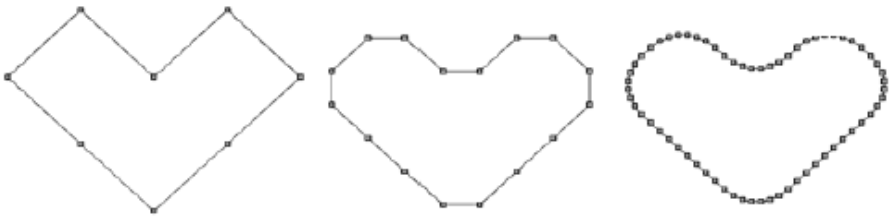
9 Subdivision Of Curves

9.1 Motivation

Consider a set of control points and the resulting quadratic B-spline. Is it possible to find a new set of finer control points that produces exactly the same curve?



It turns out that yes, this is possible. What then is the relationship between these two sets of control points? How can we find the new control points from the original control points?

9.2 Knot Insertion Process For Subdividing B-splines

We can insert midpoints as new knot values and keep the curve unchanged. This results in a relationship between the two sets of control points. In a mathematical sense this is:

old: $Q(u) = \sum_{i=0}^{m} C_i N_{i,e}(u)$ C_i is a coarse control point.

new: $Q^-(u) = \sum_{j=0}^{m'} F_j N^-_{j,e}(u)$ F_j is a fine control point.

where $N^-_{j,e}(u)$ are the set of new B-spline basis functions over the finer knot values.

If we set $Q(u) = Q^-(u)$ then $F = \begin{bmatrix} F1 \\ F2 \\ ... \end{bmatrix}$ can be obtained. In the case of the quadratic B-spline we obtain the following operations for the regular case (i.e., the areas of the curve where the basis functions are evenly spaced according to the standard knot sequence).

$$F_{j-1} = \frac{3}{4}C_{i-1} + \frac{1}{4}C_i \quad F_j = \frac{1}{4}C_{i-1} + \frac{3}{4}C_i$$

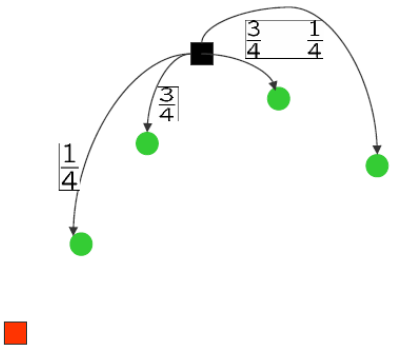


Figure 9.1: Quadratic B-spline subdivision.

We can repeat this process several times producing a finer and finer set of control points. These finer control points approach a limit where the points become a quadratic B-spline. It should also be noted that each refinement operation is affine.

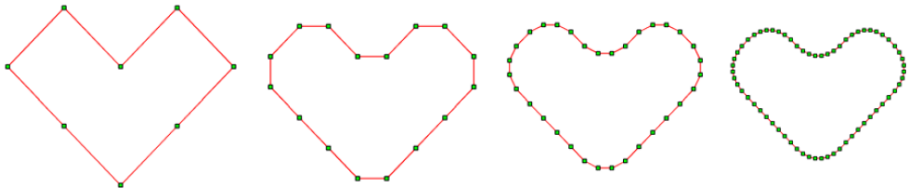


Figure 9.2: Several steps of Chaikin subdivision.

As we can see in the above figure, after several steps of the knot insertion process the control polygon is a very good approximation of the curve. This simple method of generating a quadratic B-spline is called "Chaikin subdivision".

Our filter values for Chaikin subdivision are $\frac{1}{4}$, $\frac{3}{4}$, $\frac{3}{4}$, $\frac{1}{4}$.

9.2.1 Cubic B-spline Subdivision

Using the same knot insertion approach, the following operations are obtained for cubic B-splines:

$$F_{j-1} = \frac{1}{2}C_{i-1} + \frac{1}{2}C_i F_j = \frac{1}{8}C_{i-1} + \frac{3}{4}C_i + \frac{1}{8}C_{i+1} F_{j+1} = \frac{1}{2}C_i + \frac{1}{2}C_{i+1}$$

Our filter values for cubic b-spline subdivision are $\frac{1}{8}$, $\frac{1}{2}$, $\frac{3}{4}$, $\frac{1}{2}$, $\frac{1}{8}$.

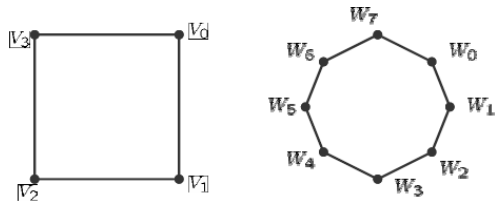


Figure 9.3: Cubic B-spline subdivision.

9.2.2 Filters For General B-spline Subdivision

Table 1: General B-spline subdivision filters.

Order	Name	Scale	Pascal's Triangle	Filter Values
1	Harr	1	1 1	1 1
2	Faber	$\frac{1}{2}$	1 2 1	$\frac{1}{2}$ 1 $\frac{1}{2}$
3	Chaikin	$\frac{1}{4}$	1 3 3 1	$\frac{1}{4}$ $\frac{3}{4}$ $\frac{3}{4}$ $\frac{1}{4}$
4	Cubic B-spline	$\frac{1}{8}$	1 4 6 4 1	$\frac{1}{8}$ $\frac{1}{2}$ $\frac{3}{4}$ $\frac{1}{2}$ $\frac{1}{8}$
5	. . .	$\frac{1}{16}$	1 5 10 10 5 1	$\frac{1}{16}$ $\frac{5}{16}$ $\frac{10}{16}$ $\frac{10}{16}$ $\frac{5}{16}$ $\frac{1}{16}$

9.3 Non B-spline Subdivision

The main advantages of the knot insertion process for B-splines:

- Convergence of successive refining.
- Smoothness of the result.

In order to have a self-contained subdivision method both of these properties must be guaranteed.

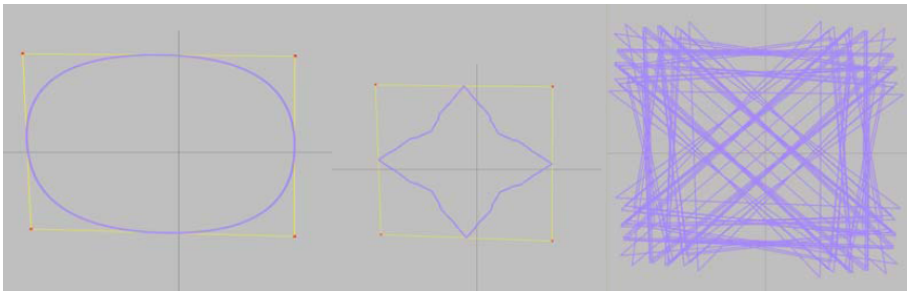


Figure 9.4: Different values for the subdivision filters may decrease continuity (middle) or result in divergence (right).

9.3.1 Dyn-Levin Interpolating Subdivision Scheme

This is not a B-spline based subdivision. The properties mentioned above have been proved directly in their paper. The filter values in this scheme are:
$F_{j-1} = -\frac{1}{16}C_{i-2} + \frac{9}{16}C_{i-1} + \frac{9}{16}C_i + -\frac{1}{16}C_{i+1}$
$F_j = C_i$
$F_{j+1} = -\frac{1}{16}C_{i-1} + \frac{9}{16}C_i + \frac{9}{16}C_{i+1} + -\frac{1}{16}C_{i+2}$

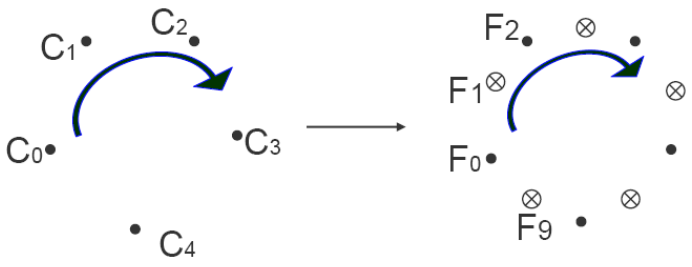


Figure 9.5: An example of Dyn-Levin subdivision.

9.4 Matrix Notation

We can specify the subdivision operation in terms of a matrix. Given a column vector C^j, the set of course control points, a high resolution C^{j+1} set of control points is great via: $C^{j+1} = P^j C^j$ (note that j and $j+1$ are superscripts and not powers in this context) where P^j is a $m \times n$ $(m < n)$ subdivision matrix and j refers to the level of subdivision. P^j has a regular banded structure. The following is the matrix for cubic B-spline subdivision.

$$P^j = \begin{bmatrix} & \frac{1}{8} & 0 & 0 & 0 & \\ & \frac{1}{2} & 0 & 0 & 0 & \\ & \frac{3}{4} & \frac{1}{8} & 0 & 0 & \\ \cdots & \frac{1}{2} & \frac{1}{2} & 0 & 0 & \cdots \\ & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & 0 & \\ & 0 & \frac{1}{2} & \frac{1}{2} & 0 & \\ & 0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & \\ & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \\ & 0 & 0 & \frac{1}{8} & \frac{3}{4} & \end{bmatrix}$$

The non-zero entries of each column are the filter values from table 1. These entries are shifted from its predecessor's by two rows. This shifting patter is due to the midpoint knot insertion scheme.

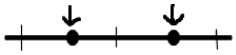


Figure 9.6: New knots are inserted midway between existing knots.

The direct consequence of this kind of insertion is that the number of new control points is almost two times the number of old control points.

9.5 Open and Closed Curves

Up until this point we have not discussed what happens at the boundaries of the curve. If we only consider the closed (periodic) case of curve we simply apply the regular filter over the entire curve.

So for Chaikin this looks like:

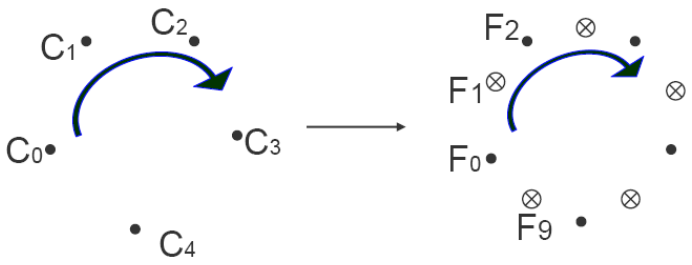


Figure 9.7: Boundary conditions for Chaikin subdivision.

We would like to have open curves treated exactly as the result of the standard knot sequence. That is, we would like the beginning and end control points to be interpolated and not have the curve shrink as we apply levels of subdivision to it. This can be done by recalculating the knot insertion process.

Consider a quadratic b-spline. Using these two knot sequences and evaluating F_j from these equations:

$$Q(u) = \sum C_i N_{i,e} Q(u) = \sum F_j N_{j,3}^{-}(u)$$

We obtain some non-regular (extraordinary) filter values to be applied near the boundaries:

$F_0 = C_0$
$F_1 = \frac{1}{2}(C_0 + C_1)$
$F_2 = \frac{3}{4}C_1 + \frac{1}{4}C_2$
.
. (regular case)

.

$F_{2n-3} = \frac{3}{4}C_{n-1} + \frac{1}{4}C_{n-2}$
$F_{2n-2} = \frac{1}{2}(C_n + C_{n-1})$
$F_{2n-1} = C_n$

The same knot insertion process gives us filters for cubic b-spline subdivision near the boundaries:

$F_0 = C_0$
$F_1 = \frac{1}{2}(C_0 + C_1)$
$F_2 = \frac{3}{4}C_1 + \frac{1}{4}C_2$
$F_3 = \frac{3}{16}C_1 + \frac{11}{16}C_2 + \frac{1}{8}C_3$
$F_4 = \frac{1}{2}(C_2 + C_3)$

and as in the previous case if we change the order of the filter values, we obtain the values around the end point.

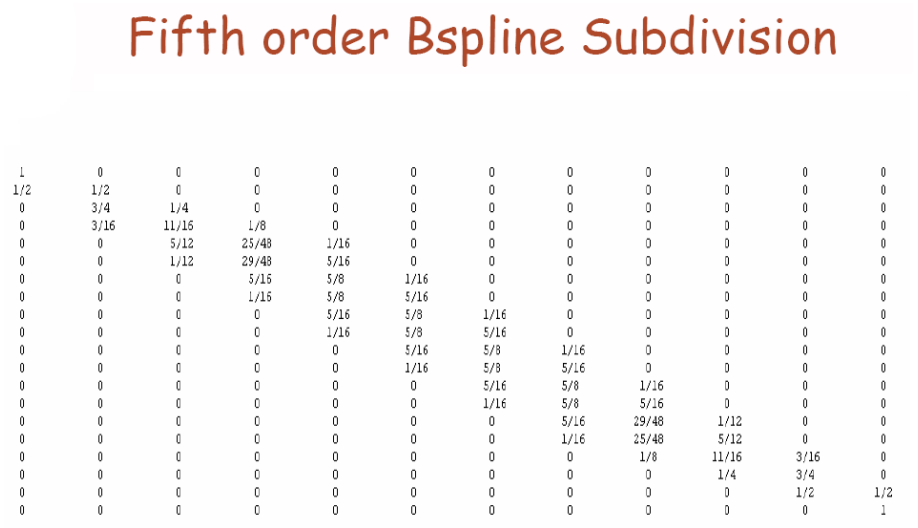
Thus complete subdivision matrix of open B-splines of order 3, 4, and 5 are:

Boundary conditions of Quadratic Bspline

1	0	0	0	0	0	0	0	0	0
1/2	1/2	0	0	0	0	0	0	0	0
0	3/4	1/4	0	0	0	0	0	0	0
0	1/4	3/4	0	0	0	0	0	0	0
0	0	3/4	1/4	0	0	0	0	0	0
0	0	1/4	3/4	0	0	0	0	0	0
0	0	0	3/4	1/4	0	0	0	0	0
0	0	0	1/4	3/4	0	0	0	0	0
0	0	0	0	3/4	1/4	0	0	0	0
0	0	0	0	1/4	3/4	0	0	0	0
0	0	0	0	0	3/4	1/4	0	0	0
0	0	0	0	0	1/4	3/4	0	0	0
0	0	0	0	0	0	3/4	1/4	0	0
0	0	0	0	0	0	1/4	3/4	0	0
0	0	0	0	0	0	0	3/4	1/4	0
0	0	0	0	0	0	0	1/4	3/4	0
0	0	0	0	0	0	0	0	1/2	1/2
0	0	0	0	0	0	0	0	0	1

Boundary conditions of Cubic Bspline

1	0	0	0	0	0	0	0	0	0	0
1/2	1/2	0	0	0	0	0	0	0	0	0
0	3/4	1/4	0	0	0	0	0	0	0	0
0	3/16	11/16	1/8	0	0	0	0	0	0	0
0	0	1/2	1/2	0	0	0	0	0	0	0
0	0	1/8	3/4	1/8	0	0	0	0	0	0
0	0	0	1/2	1/2	0	0	0	0	0	0
0	0	0	1/8	3/4	1/8	0	0	0	0	0
0	0	0	0	1/2	1/2	0	0	0	0	0
0	0	0	0	1/8	3/4	1/8	0	0	0	0
0	0	0	0	0	1/2	1/2	0	0	0	0
0	0	0	0	0	1/8	3/4	1/8	0	0	0
0	0	0	0	0	0	1/2	1/2	0	0	0
0	0	0	0	0	0	1/8	3/4	1/8	0	0
0	0	0	0	0	0	0	1/2	1/2	0	0
0	0	0	0	0	0	0	1/8	11/16	3/16	0
0	0	0	0	0	0	0	0	1/4	3/4	0
0	0	0	0	0	0	0	0	0	1/2	1/2
0	0	0	0	0	0	0	0	0	0	1



9.6 Subdivision Without Knot Insertion

The idea of the subdivision method is easy and simple enough to generalize for many curves and surfaces. However, the knot insertion process only works for B-spline curves and tensor product surfaces. The question now is are there subdivision methods that do not use this knot insertion process?

First all, we must ask what are the benefits of using B-splines and the knot insertion process. The benefits of this process are that we automatically obtain the following properties:

- affine invariance
- convergence
- smoothness

For example, when we use midpoint knot insertion for the quadratic B-spline it is obvious that the resulting Chaikin subdivision curve converges to a quadratic B-spline and generates has a smooth limit. However if we start a similar subdivision scheme with the filter values $\frac{1}{3}$, $\frac{2}{3}$, $\frac{2}{3}$, $\frac{1}{3}$ we need some method of showing such properties.

9.6.1 Analysis Of Subdivision Methods

We would like to investigate the properties of $\{C^{j+1}\}$ where $C^{j+1} = P^j C^j$. Although as we perform several levels of subdivision the matrices P^j are different matrices when j varies; however they have a regular structure and are locally the same although with difference sizes. The important fact is that properties such as convergence, continuity, and smoothness of the limit curve can be checked locally. For example, if we only focus on five points in the cubic B-spline subdivision (see figure 9.9) then a constant

Figure 9.8: Local cubic B-spline subdivision

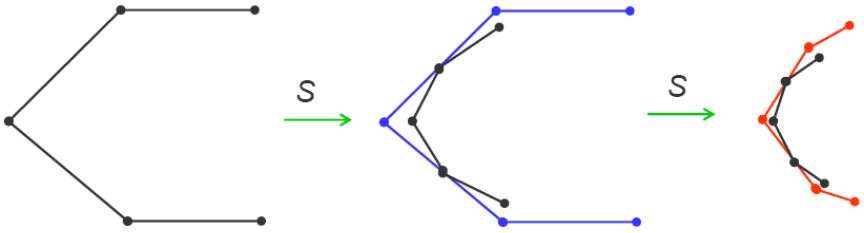


Figure 9.9: Local subdivision matrix for dyn-levin subdivision scheme.

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} & 0 & -\frac{1}{16} \\ 0 & 1 & 0 & 0 & 0 \\ -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} \\ 0 & 0 & 0 & 1 & 0 \\ -\frac{1}{16} & 0 & -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} \\ 0 & 0 & 0 & 0 & 1 \\ \frac{9}{16} & -\frac{1}{16} & 0 & -\frac{1}{16} & \frac{9}{16} \end{bmatrix}$$

and simple matrix, S, is all that is required. This matrix expresses a portion of the subdivision matrix P^j.
Therefore we can simply replace

$$C^{j+1} = P^j C^j$$

with

$$L_{j+1} = SL_j$$

where L_{j+1} and L_j represent the two five element vectors in the levels j and $j+1$ we are interested in and S is the local subdivision matrix.

This gives us the result:

$$L_{j+1} = SL_j = S(SL_{j-1} = \ ... \ = S^{j+1}L_0$$

where L_0 refers to the intial set of control points and S^{j+1} is the power $j+1$ of S. Consequently all properties can be investigated using "powers" of S. The Eigenvalues and Eigenvectors of S are very important tools in this analysis. A detailed discussion of this topic is beyond the scope of this course, however two brief examples follow.

Example 9.1
Analysis of Chaikin The leading eigenvalues of S are:

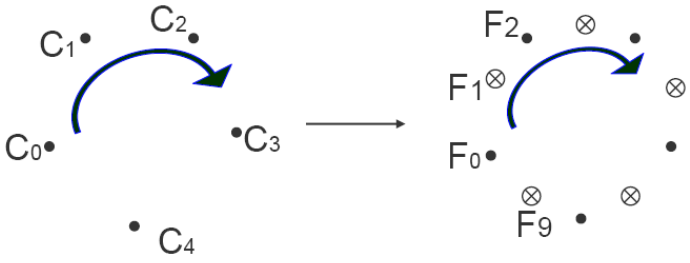
$$eigs(S): \quad \lambda_0 = 1,\ \lambda_1 = \frac{1}{2},\ \lambda_2 = \frac{1}{4},\ \lambda_3 = \frac{1}{4}$$

This arrangement of eignvalues can lead us to all the three properties that we mentioned in subsection 9.6.

Example 9.2
Dyn-levin-Gregory Interpolating Curve Subdivision

This is a subdivision that interpolates the control points. The even points are exactly the original set of controls points. The odd control points use $-\frac{1}{16},\ \frac{9}{16},\ \frac{9}{16},\ -\frac{1}{16}C_{i+2}$ as the weights. These numbers of have been selected carefully to dictate smoothness

Figure 9.10: Dyn Levin subdivision



of the limited curve. This can be checked by the eigenvalues of the local subdivision matrix.

Figure 9.11: Local cubic B-spline subdivision

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} & 0 & -\frac{1}{16} \\ 0 & 1 & 0 & 0 & 0 \\ -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} \\ 0 & 0 & 0 & 1 & 0 \\ -\frac{1}{16} & 0 & -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} \\ 0 & 0 & 0 & 0 & 1 \\ \frac{9}{16} & -\frac{1}{16} & 0 & -\frac{1}{16} & \frac{9}{16} \end{bmatrix}$$

The leading eigenvalues are:

$$eigs(S): \quad \lambda_0 = 1,\ \lambda_1 = \frac{1}{2},\ \lambda_2 = \frac{1}{4},\ \lambda_3 = \frac{1}{4}$$

These eigenvalues are exactly t he same as Chaikin subdivision's eigenvalues.

9.7 Tensor Product Surfaces

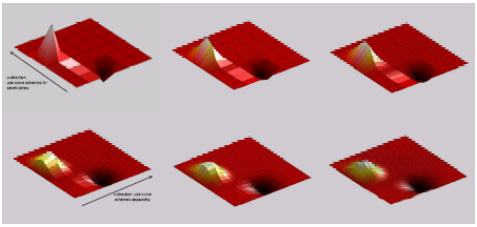
We can use any subdivision curve schemes along u and v direction for a given net of control points. Image can be viewed as a net of control points where pixel values refer to the coordinate of the vertices. Consequently we can obtain the following techniques for images:

- resolution enhancement
- smoothness filter

Figure 9.12: Subdivision used to increase the size of an image.

The same approach can be applied to tensor product surfaces. In this example cubic B-spline subdivision is employed.

Figure 9.13: Cubic B-spline subdivision of a tensor product surface.



10 General Subdivision

10.1 Loop Subdivision

Developed by Charles Loop as part of his Master's research in 1987.

10.1.1 Face Split

The refinement step splits each triangular face into four new sub-faces.

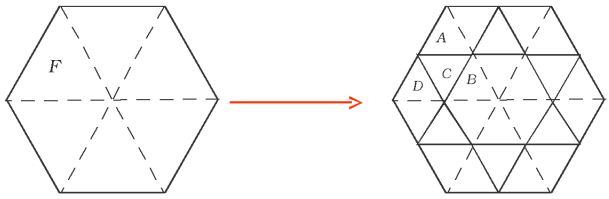


Figure 10.1: Face Splitting of Loop Subdivision.

10.1.2 Masks

The vertices of the refined mesh are positioned using a weighted average of coarse mesh. Two types of vertices exist

- Vertex-vertex or even vertex.
- Edge-vertex or even vertex.

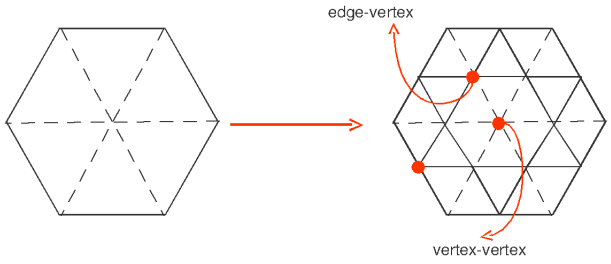
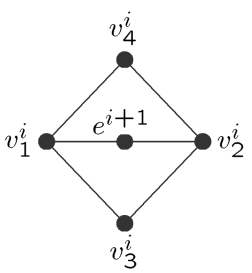
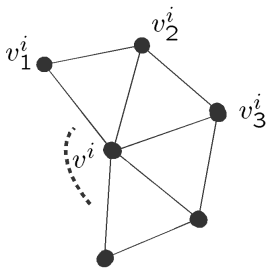


Figure 10.2: Types of vertices in Loop Subdivision.





$$e^{i+1} = \frac{3}{8}v_1^i + \frac{3}{8}v_2^i + \frac{1}{8}v_3^i + \frac{1}{8}v_4^i \qquad v^{i+1} = (1 - n\alpha)v^i + \alpha \sum v_j^i$$

Figure 10.3: Loop Subdivision masks. $\alpha = \frac{1}{n}\left(\frac{5}{8} - (\frac{3}{8} + \frac{1}{4}cos\frac{2\pi}{n})^2\right)$

Example 10.1

Masks for Regular case

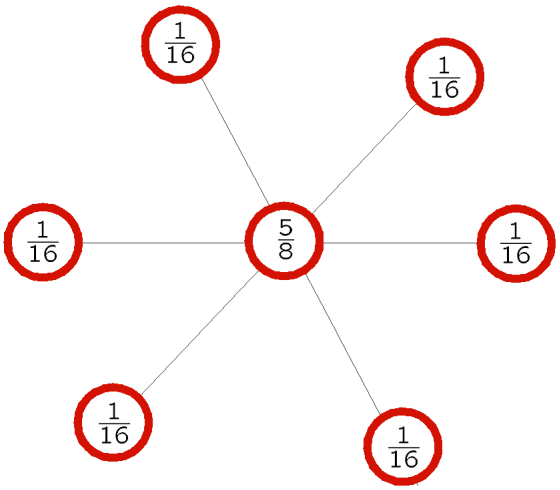
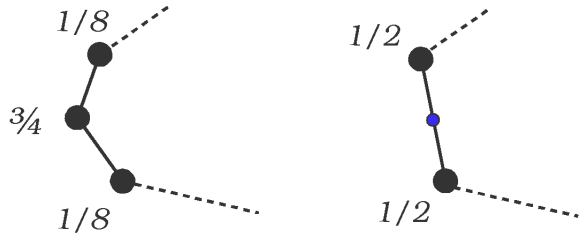


Figure 10.4: Mask of vertex-vertex for regular case of Loop subdivision.

10.1.3 Boundary

In the case of boundary, the cubic B-spline curve subdivision mask is applied.



Even Mask Odd Mask

Figure 10.5: Loop subdivision masks for boundary case

10.1.4 Creases and Corners

Not all objects are everywhere smooth. Creating sharp edges and corners by using Loop subdivision would require a lot of triangles around the crease or corner. Apply a different mask for creases to relax tangent plane continuity across the sharp edge.

A crease is a tagged edge at which the Loop subdivision surface is not tangent plane smooth. There are three vertex types depending on the number neighbouring creases:"

- Normal vertex: A vertex with no crease edges. Apply normal odd mask of Loop subdivision.
- Dart: A vertex with only one crease edge. Generally, apply normal odd mask of Loop subdivision. However, a few extra cases must be taken care of.
- Crease: A vertex with two crease edges. Apply cubic B-spline curve subdivision masks only using the neighbouring vertices on crease edges.
- Corner: A vertex with more than two crease edges. Vertex position remains unchanged.

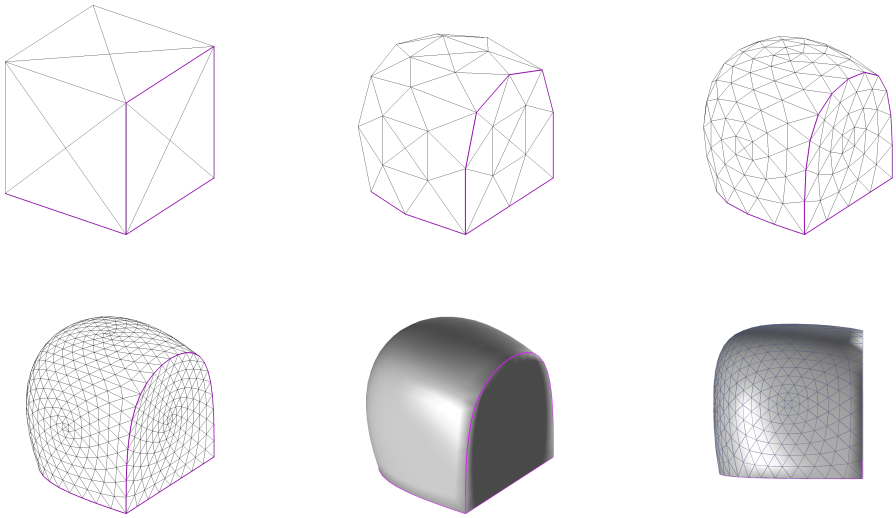


Figure 10.6: Creases and corners in Loop Subdivision

10.1.5 Properties of Loop Subdivision

- Subdivision surface is C^2 everywhere except at extra-ordinary vertices
- Subdivision surface is C^1 at extra-ordinary vertices
- Loop subdivision has strong convex hull property
- Mostly ordinary vertices are created (odd vertices).

10.2 Catmull-Clark Subdivision

Catmull-Clark subdivision was introduced in Computer Aided Design in 1978. It is generalization of cubic B-spline subdivision to arbitrary surfaces. Produces C^2 surface at ordinary vertices and smooth tangent at extra-ordinary points.

10.2.1 Face Split

Each face is split into four new faces.

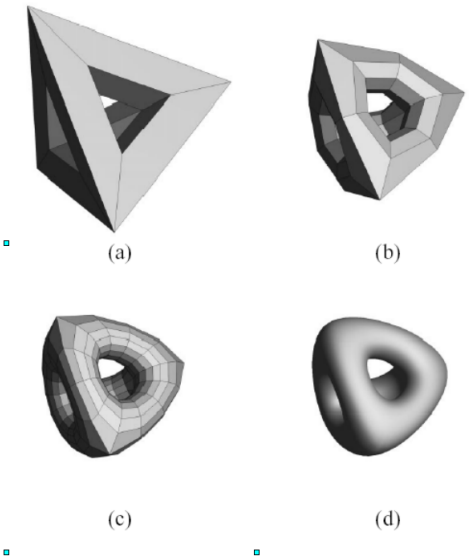


Figure 10.7: Example of Catmull-Clark subdivision.

10.2.2 Masks

Three types of vertices exist

- Face-vertex (f_i^j)
- Edge-vertex (e_i^j)
- Vertex-vertex (v_i^j)

Each new face consists of a loop of four points

$$v_i^j \rightarrow e_i^j \rightarrow f_i^j \rightarrow e_{i+1}^j$$

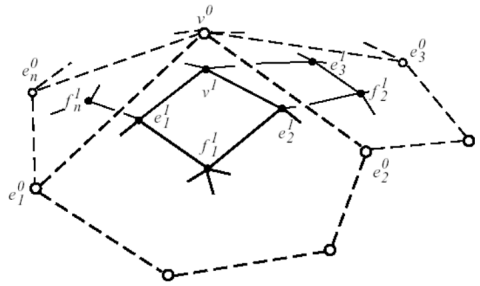


Figure 10.8: Three types of vertices in Catmull-Clark subdivision.

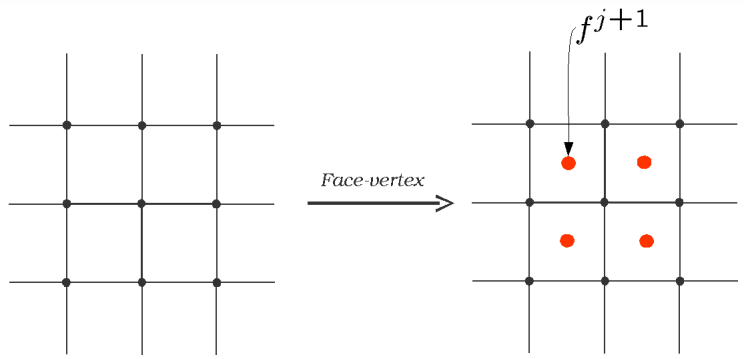


Figure 10.9: Face-vertex mask. $f^{j+1} = \frac{1}{n}\sum_{k=1}^{n} v_k^j$ where n is the number of vertices in face.

Example 10.2

Face-vertex Mask for Quad

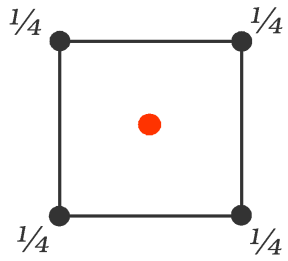


Figure 10.10: Mask of face-vertex for regular case of Catmull-Clark subdivision.

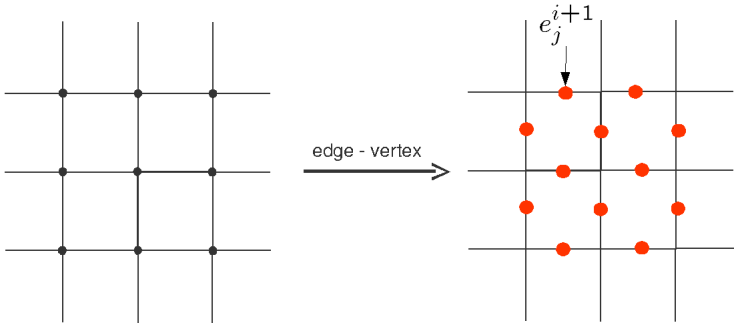


Figure 10.11: Mask for edge-vertex in Catmull-Clark subdivision. $e_i^{j+1} = \frac{1}{4}(v^j + e_i^j + f_{i-1}^{j+1} + f_i^{j+1})$ where $f_{-1}^j = f_n^j$. (NOTE: In above picture, switch i and j).

Example 10.3

Edge-vertex Mask for Quad

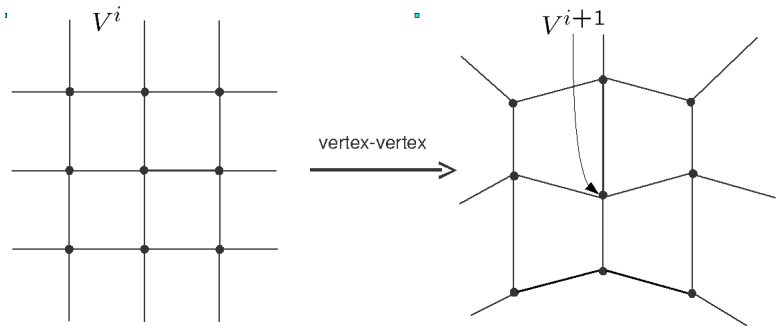


Figure 10.13: Mask of vertex-vertex in Catmull-Clark subdivision. $v_i^{j+1} = \frac{n-2}{n} v_i^j + \frac{1}{n^2} \sum_i e_i^j + \frac{1}{n^2} \sum_i f_i^{j+1}$. (NOTE: In the above figure, replace i with j).

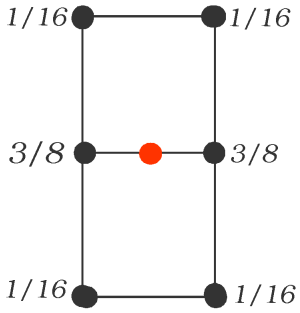


Figure 10.12: Mask of edge-vertex for quad in Catmull-Clark subdivision.

10.2.3 Boundary

Follow the same rules as in section 10.1.3, but use Catmull-Clark masks for normal vertices.

10.2.4 Creases and Corners

Follow the same rules as in section 10.1.4, but use Catmull-Clark masks for normal vertices.

10.3 Butterfly Subdivision

Proposed by Dyn, Levin and Gregory at TAG in 1991. It is a triangular based subdivision that produces a C^1 interpolating subdivision.

10.3.1 Face Split

Each face is split into four new sub-faces similar to Loop subdivision.

10.3.2 Masks

Two type of vertices

- Even (old) vertices: unchanged
- Odd (new) vertices: according to figure.

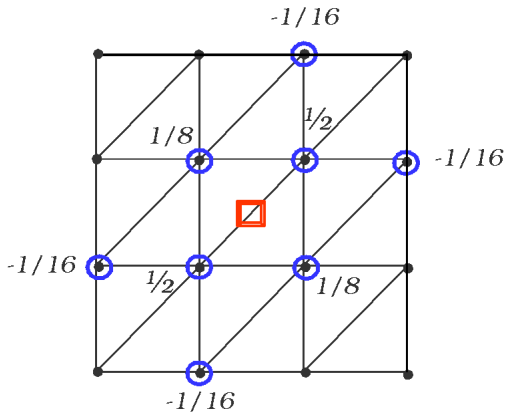


Figure 10.14: Edge-vertex masks in Butterfly subdivision

A modified version handles extra-ordinary vertices. Three cases exist

- 6-vertices to 6-vertices (regular)
- k-vertices to 6-vertices
- k-vertices to k-vertices

Regular Case of Butterfly Subdivision

$a = \frac{1}{2}, b = \frac{1}{8}, c = -\frac{1}{16} and\ d = 0$

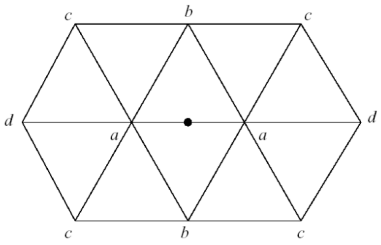


Figure 10.15: Butterfly edge-vertex mask for 6-vertices to 6-vertices (regular)

Extra-ordinary Case of Butterfly Subdivision

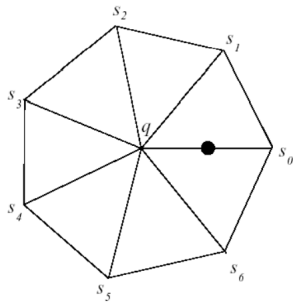


Figure 10.16: Butterfly edge-vertex mask for 6-vertices to k-vertices ($k \neq 6$). $s_j = \frac{1}{k}\left(\frac{1}{4} + cos(\frac{2\pi j}{k}) + \frac{1}{2}cos(\frac{4\pi j}{k})\right)$

In the case of k-vertices to k-vertices, the value of s_j is computed twice and then averaged.

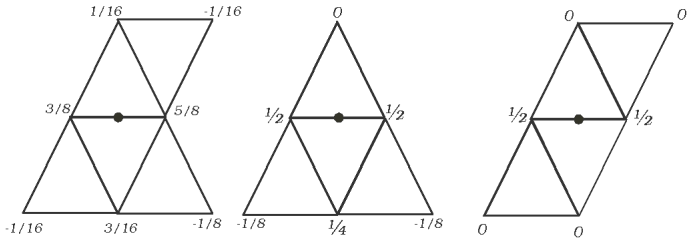


Figure 10.18: Near boundary cases for Butterfly subdivision.

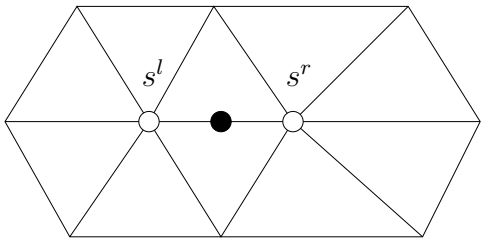


Figure 10.17: Butterfly edge-vertex mask for k-vertices to k-vertices. $s = \frac{s^l + s^r}{2}$

10.3.3 Boundary

Use interpolating curve subdivision masks. Use special rules for near boundary cases.

10.4 Adaptive Subdivision

10.4.1 Motivation

Sometimes we don't need to subdivide the whole model. An object with 1000 faces, will have 16000 faces after only 2 subdivision steps. Rather than subdividing the whole model, we only subdivide faces that match our refinement criteria.

10.4.2 Adaptive Loop Subdivision

The problem with adaptive loop subdivision is cracks that are due to vertices with incomplete neighbourhoods.

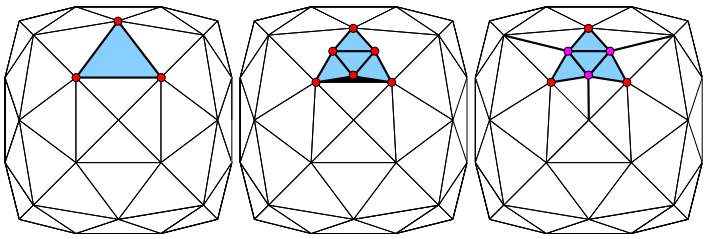


Figure 10.19: Cracks when only some faces are subdivided.

Cracks can be avoided by connecting the odd vertices on the boundary to their opposite vertex. However, this causes the valence of these vertices to increase when subdivision is repeated. In a new algorithm, suggested by Reza Pakdel and Faramarz Samavati, the neighbouring faces are also subdivided during refinement. This in effect creates vertices with proper neighbourhood in the subdivision area and, also limits subdivision depth difference between neighbouring triangles to at most one. Hence, the resolution of the subdivision area increases gradually from coarse to fine.

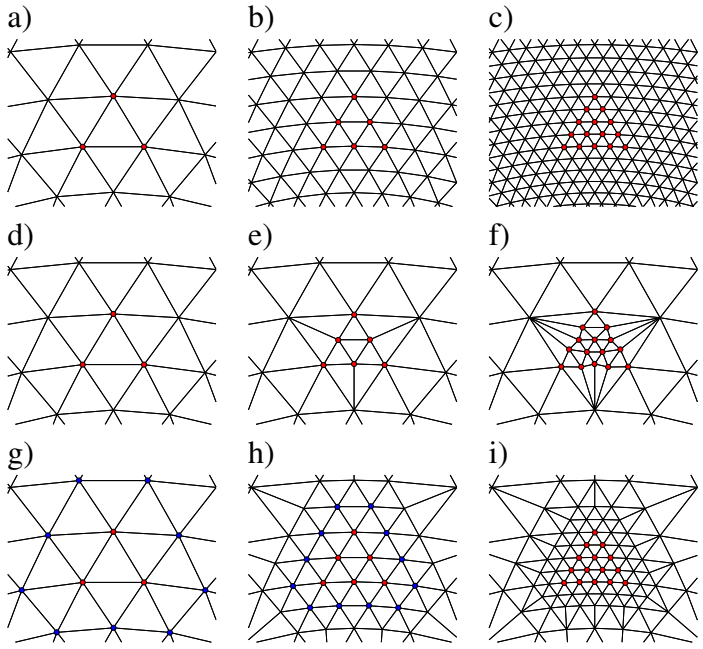
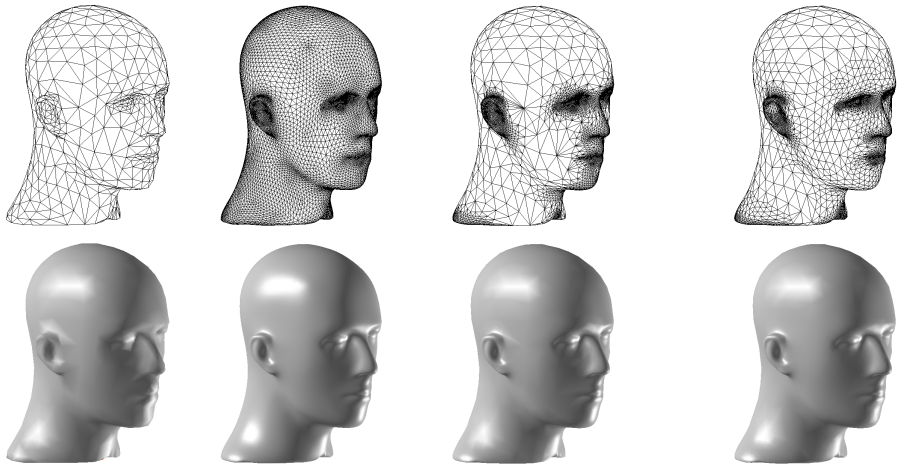


Figure 10.20: Comparison of conventional Loop subdivision to adaptive subdivision, and our incremental algorithm.



Coarse	Normal	Simple adaptive	Incremental adaptive
1420 faces	22720 faces	10446 faces	10238 faces

Figure 10.21: Comparison of conventional Loop subdivision, simple adaptive subdivision and our incremental adaptive algorithm.

11 Multiresolution, Wavelets, and Reverse Subdivision

11.1 Introduction

As we have discussed before, it is possible to increase the resolution of our models using various subdivision schemes.

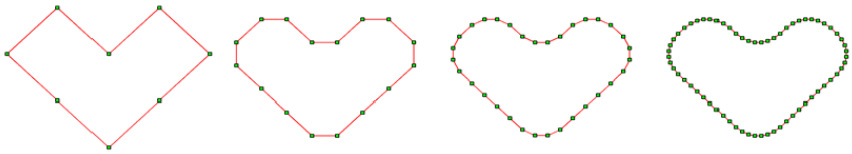


Figure 11.1: Subdivision of a closed curve.

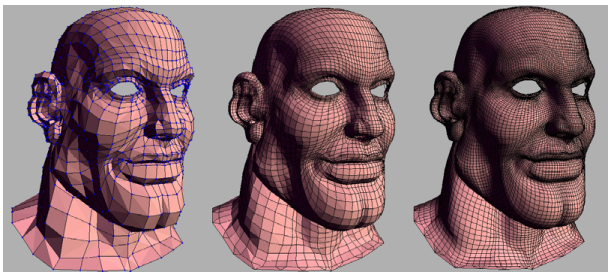


Figure 11.2: Subdivision of a mesh with arbitrary topology.

Figure 11.3: Subdivision used to increase the resolution of an image.

However, in many applications both the increasing and decreasing of the model's resolution is necessary. In order to reduce the resolution we could simply just remove every other point. This is not a perfect solution. In a multiresolution framework the following properties must be considered:

- feasibility of reducing and increasing the model's resolution.
- reconstruction of the fine model.
- efficiency of the method (a linear time algorithm is desired.
- efficiency of memory (a coarse model should not use more memory than a fine model).
- shape preservation (we would like the coarse model to be a good approximation of the fine model).

So in looking at our solution of removing every other point we find that the second point is contradicted. To fix this we might suggest keep in memory all the two resolutions (coarse and fine) of the model. This contradicts the memory constraint.

So what is the right way to do this? Obviously in reducing the number of points in a curve or a mesh we are removing some information. We can look at this problem in terms of another common graphics operation, projection. When we project a 3D model into 2D screen space we lose depth information. Normally this operation is irreversible unless this depth information is captured. If we have depth information we can simply unproject the image and then add the depth information. Our approach for reverse subdivision is the same; we will coarsen our model and capture the detail information that we are missing.

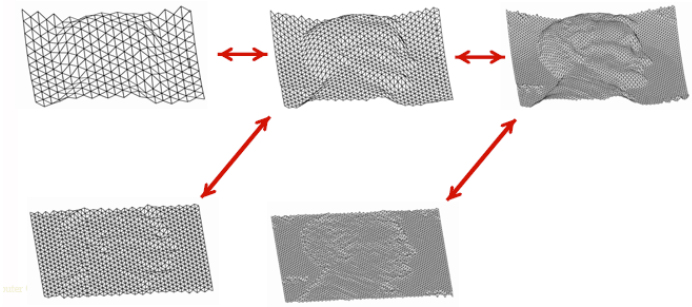


Figure 11.4: Multiresolution information of a model of an American penny.

The act of converting a fine model into a coarse approximation and the corresponding details is called **decomposition**. We start our discussion with a simple example of multiresolution based on Haar wavelets.

11.2 Harr Wavelets

Consider a row of a small picture:

$$[5, 7, 10, 14, 13, 13, 8, 6]$$

These values can also be considered as coordinates of a curve. This can also be represented by an approximating function

$$\bar{f}(x) = 5\bar{\phi}_0(x) + 7\bar{\phi}_1(x) + \ \ldots \ + 6\bar{\phi}_0(x)$$

where ϕ_i is a box function with height one over the interval $[i, i+1]$. The bar notation (i.e., $\bar{f}$) is used to mark the fact that we are referring to fine space.

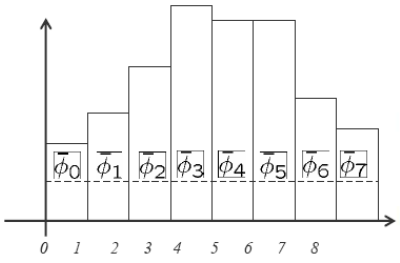


Figure 11.5: Our function approximating the row of a small picture.

A simple and good method for creating a coarse estimate is to simply take use the average of every two samples.

$c_0 = \frac{5+7}{2} = 6 \qquad c_1 = \frac{10+14}{2} = 12$
$c_2 = \frac{13+13}{2} = 13 \quad c_3 = \frac{8+6}{2} = 7$

$C = [6, 12, 13, 6]$

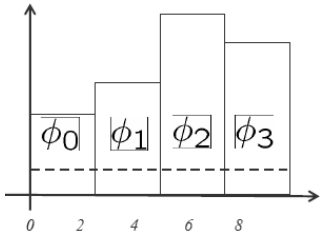


Figure 11.6: Our coarse approximation of the row of a small picture.

This now leads us to our previous question of how we should return to our fine data from the coarse data. Considering our data we find that c_0 is one less than f_0 and one

greater than f_1. Also we find that c_1 is two less than f_2 and two greater than f_3. The details that we need to save are simply the difference between the average used for the coarse data and the fine data value. So our detail is: $first - average$.

Figure 11.7: The details mark the difference between the fine point (circle) and the coarse point (square).

$d_0 = 5 - \frac{5+7}{2} = -1 \qquad d_1 = 10 - \frac{10+14}{2} = -2$
$d_2 = 13 - \frac{13+13}{2} = 0 \quad d_3 = 8 - \frac{8+6}{2} = 1$

$D = [-2, -1, 0, 1]^T$

So our resulting multiresolution representation of the data is $[6, 12, 13, 7, -1, -2, 0, 1]$.
This gives us two operations:
Decomposition:

$$\begin{array}{lll} F & \rightarrow & C \\ & \searrow & D \end{array}$$

Reconstruction:

$$\begin{array}{ll} C & \rightarrow F \\ D & \nearrow \end{array}$$

F is the coefficient vector of the fine basis $\bar{\phi}$.
C is the coefficient vector of the coarse basis ϕ.
The question is, what is D a coefficient vector of?

11.2.1 Interpretation of D

Let's find a basis function for the left and right operations.

left = $a + 1 * details$
right = $a - 1 * details$
therefore $\bar{\phi}_0 - \bar{\phi}_1 = \psi_0$

$$\psi(x) = \begin{cases} 1 & x < a \\ -1 & x > a \end{cases}$$

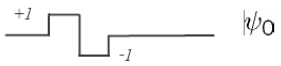


Figure 11.8: The Haar wavelet function (ψ).

So D is the coefficient vector of the new basis ϕ.

11.2.2 Decomposition

$\bar{f}(x) = f(x) + g(x)$

$$\begin{array}{ccc} F & \rightarrow & C \\ & \searrow & D \end{array} \qquad \begin{array}{ccc} \bar{f}(x) & \rightarrow & f(x) \\ & \searrow & g(x) \end{array}$$

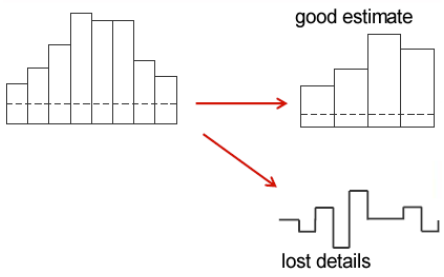


Figure 11.9: Decomposition.

Remember that:

- $\bar{f}(x)$ is the high frequency function
- $f(x)$ is the low frequency approximation
- $g(x)$ is the high energy part of $\bar{f}(x)$ (the small variations)

We can apply this decomposition several times. So for our simple example this is:

$[5, 7, 10, 14, 13, 13, 8, 6] \longrightarrow [6, 12, 13, 7, -1, -2, 0, 1] \quad C^{k+1} \rightarrow C^k,\ D^k$
We then repeat this for C^k:

$[6, 12, 13, 7] \longrightarrow [9, 12, -3, -3]$
and we can represent all of our data as:

$[5, 7, 10, 14, 13, 13, 8, 6] \longrightarrow [9, 12, -3, -3, -1, -2, 0, 1] \quad C^{k+1} \rightarrow C^{k-1},\ D^{k-1},\ D^k$

11.3 Applications

11.3.1 Data Transmission

A major application of wavelets is in data transmission. In sending a large amount of data through a network we can first transform the data into a multiresolution wavelet representation and send across a coarse representation (i.e., C^0) and then the levels of detail necessary to reach the final fine level of detail (C^k). This allows the receiver of this data to view the coarse data as it comes it to ensure that the desired data is being transmitted. This is much like the processes progressive jpg encoding allows.

11.3.2 Image Compression

An image can be thought of as rows and columns of pixel values. These values can be used by wavelets to decrease the resolution of the image. To do this we first decompose the rows of the image.

Figure 11.10: The row decomposition of our image. The resulting coarse approximation is on the left, the details are on the right.

Then we decompose the columns of the image and the row details.

Figure 11.11: The final resulting decomposition of our image. Notice that most details are close to zero.

As can be seen in the figure above most of the details are close to zero, therefore we can simply through out some of the details and achieve a good compression rate.

11.3.3 Video Compression

A video is a sequence of frames (images). We can use wavelets to manipulate the resolution of time. We can reduce the number of frames stored and, as in image compression, we throw out the insignificant details to achieve compression.

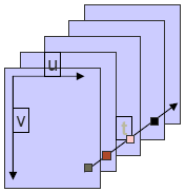


Figure 11.12: A pixel in a sequence of frames that can be represented by wavelets.

In the figure below we can see that decomposition reduces the number of frames creating a short, rough animation and some details.

Figure 11.13: The decomposition of a short animation.

As the video is played we use reconstruction to increase the number of frames again (without the details we have thrown out).

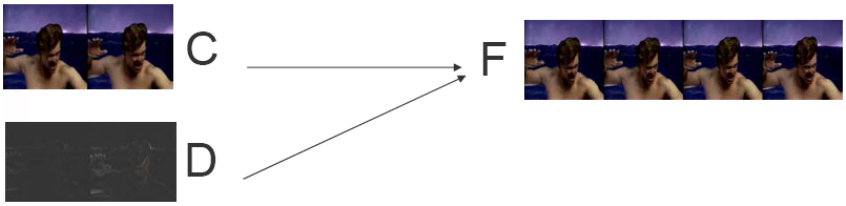


Figure 11.14: The reconstruction of the animation.

11.4 Multiresolution Filter Matrices

For Haar wavelets we can easily create the following matrices for decomposition and reconstruction:

$$P = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} Q = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}$$

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} B = \begin{bmatrix} \frac{1}{2} & \frac{-1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{-1}{2} \end{bmatrix}$$

So the decomposition is performed by:

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \end{bmatrix}$$

$$\begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ D_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{-1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{-1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{-1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{-1}{2} \end{bmatrix} \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \end{bmatrix}$$

or more simply as $C = AF$ and $D = BF$.

The reconstruction is then the following operation:

Firstly, remember that $F_0 = C_0 + D_0$, $F_1 = C_0 - D_0$, $F_2 = C_1 + D_1$, $F_3 = C_1 - D_1$, . . . We can do this with the following matrices:

$$\begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ D_3 \end{bmatrix}$$

or more simply $F = PC + QD$.

Mathematically, we can specify the multiresolution operations at a high level in terms of the filter matrices A^n, B^n, P^n, and Q^n of a specific scheme. Given a column vector C^n of samples, a lower-resolution level C^{n-1} is created via decomposition using the analysis filters A^n and B^n. The superscripts refer to the decomposition level, where n is the original, highest resolution. The matrix A^n is used to down-sample the input, giving a coarsened approximation to C^n:

$$C^{n-1} = A^n C^n.$$

The details D^{n-1} lost through the down-sampling are captured using B^n:

$$D^{n-1} = B^n C^n.$$

Recovering C^n, the previous level's points, is called reconstruction. It involves refinement of the coarsened points C^{n-1} and details D^{n-1} using the synthesis filters P^n and Q^n which reverse the operations of A^n and B^n:

$$C^n = p^n C^{n-1} + Q^n D^{n-1}.$$

The multiresolution operations are successively applied to each generated level creating a hierarchy of details that gives us the original input when reconstructed. Although A^n, B^n, P^n, and Q^n are varied at different levels, they have a regular and repetitive structure. This fact is important for the efficient implementation of these operations.

A very important property is that $[\frac{A}{P}][B|P] = I$, or written another way, $[\frac{A}{P}] = [B|P]^{-1}$.

Haar wavelets are good for approximating flat functions (i.e., straight lines), however they are not good for approximating smooth curves and surfaces. Better wavelets for curves are multiresolution filters that correspond to B-splines.

11.5 Creation of Filter Matrices

So the wavelets approach to constructing our decomposition matrices is:

- Build the wavelets functions.
- Use the functions to find multiresolution filter matrices A, B, P, and Q.

The drawback to this approach is that conventational B-spline wavelets are rather complicated to deal with and yield very ugly matrices.

The reverse subdivision matrices can also be found using a discrete local least squares method. The resulting filters automatically produce the best coarse points for local neighborhoods of fine points. The width of this neighborhood dictates the bandwidth of A and B. Wider filters generate a better coarse approximation but require more computations. There is an efficient technique for finding filters that considered all the fine points to create coarse points described by the paper at *http://pages.cpsc.ucalgary.ca/ samavati/papers/Subdiv.pdf*.

Example 11.1
The Chaikin Filters

The A Filter:

$$C_0 = \frac{-1}{4}F_{-1} + \frac{3}{4}F_0 + \frac{3}{4}F_1 + \frac{-1}{4}F_2$$

$$A = \begin{bmatrix} & & & \cdot & \cdot & \cdot & & \\ \frac{-1}{4} & \frac{3}{4} & \frac{3}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-1}{4} & \frac{3}{4} & \frac{3}{4} & \frac{1}{4} & 0 & 0 \\ & & & \cdot & \cdot & \cdot & & \end{bmatrix}$$

The B Filter:

$$D_0 = \frac{1}{4}F_{-1} + \frac{-3}{4}F_0 + \frac{3}{4}F_1 + \frac{-1}{4}F_2$$

$$B = \begin{bmatrix} & & & & \cdots & & & \\ \frac{1}{4} & \frac{-3}{4} & \frac{3}{4} & \frac{-1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & \frac{-3}{4} & \frac{3}{4} & \frac{-1}{4} & 0 & 0 \\ & & & & \cdots & & & \end{bmatrix}$$

The Q Filter:

$$F_{-1} = \frac{3}{4}C_{-1} + \frac{1}{4}C_0 + \frac{3}{4}D_{-1} + \frac{-1}{4}D_0$$

$$F_0 = \frac{1}{4}C_{-1} + \frac{3}{4}C_0 + \frac{1}{4}D_{-1} + \frac{-3}{4}D_0$$

$$Q = \begin{bmatrix} & \frac{-1}{4} & 0 & \\ & \frac{-3}{4} & 0 & \\ \cdots & \frac{3}{4} & \frac{-1}{4} & \cdots \\ & \frac{1}{4} & \frac{-3}{4} & \\ & 0 & \frac{3}{4} & \\ & 0 & \frac{1}{4} & \end{bmatrix}$$

11.5.1 Diagram/Mask View

The standard notation does not help us when we are dealing with surfaces with arbitrary topology. Instead we now use a diagrammatic notation that is compact and easily translatable to general surfaces.

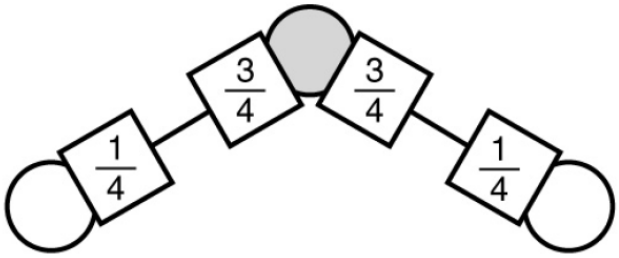


Figure 11.15: Diagrammatic notation of reverse Chaikin subdivision.

This mask corresponds to the matrix columns. The weights on the fine points show the amount of contribution there is from the fine point to derive the coarse point.

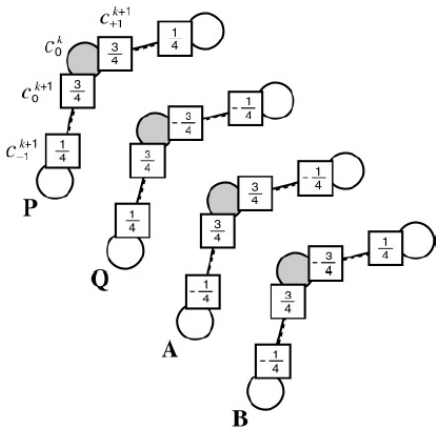


Figure 11.16: Diagrammatic notation for all the Chaikin matrices.

Above are the multiresolution masks for the quadratic B-spline subdivision (Chaikin subdivision). This is a situation in which there is a tie between two fine points for the associates with a given coarse point. The association can be made arbitrarily, but must be made consistently for all points.

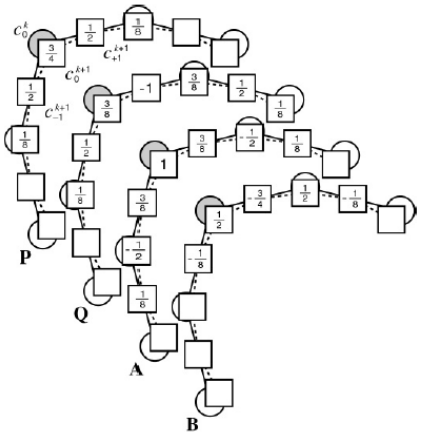


Figure 11.17: Diagrammatic notation for all the cubic B-spline matrices.

References

Local Filters: Reversing Subdivision Rules: Local Linear Conditions and Observations on Inner Products, R.H. Bartels and F.F. Samavati, Journal of Computational and Applied Mathematics, Vol 119, Issue 1-2, pp. 29-67, 2000.

Global Filters: Multiresolution curve and surface representation by reversing subdivision rules. F.F. Smavati and R.H. Bartels, Computer Graphics Forum, 18(2), 97-119, June 1999.

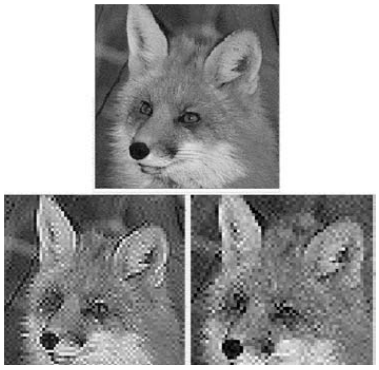


Figure 11.18: Top image is the original uncompressed image. Bottom left image is an approximation based on local filters, bottom right image is an approximation based on global filters.

12 Implicit Modeling

12.1 Motivation

We can represent an object by many mathematical models. Parametric and implicit representations are two important modeling approaches.

Example 12.1

The unit circle:

- Parametric form:

$$Q(u) = \begin{bmatrix} x(u) \\ y(u) \end{bmatrix} = \begin{bmatrix} \cos u \\ \sin u \end{bmatrix} \quad 0 \leq u \leq 2\pi$$

vary u and evaluate $x = x(u)$ and y=y(u)

- Implicit form:

$$f(x, y) = x^2 + y^2 - 1 \quad \text{or} \quad f(p) = \|p\|^2 - 1$$

Find all $p = (x, y) \in \mathbf{E}^2$ such that $f(x, y) = 0$ or $f(p) = 0$. Function f does not not explicitly describe the curve, but implies its existence.

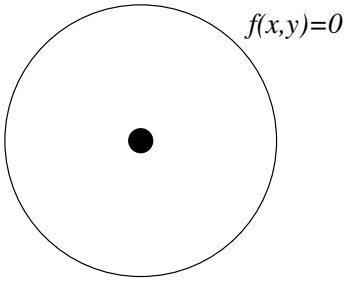


Figure 12.1: The unit circle

In the above example, finding all p is not a simple problem, parametric modeling is usually preferred. However, in some special kind of applications, implicit modeling is advantageous. Intuitively, an implicit surface consists of set of points in 3D that satisfy a particular requirement. This requirement is represented mathematically by function f. Function f characterizes a volume. In addition, $f(p)$ is proportional to the distance between p and the surface.

12.2 Benefit of Implicit Modeling

In solid modeling, it is important to be able to find the inside and outside of the solid. In implicit modeling, $f(x, y) < 0$ implies inside the model and $f(x, y) > 0$ implies outside.

Example 12.2

In the previous example

- $f(0, 0) = -1 \Rightarrow (0, 0)$ is inside
- $f(0.5, 0.5) = -0.5 \Rightarrow (0.5, 0.5)$ is inside
- $f(1, 0) = 0 \Rightarrow (1, 0)$ is on the curve
- $f(1, 1) = 1 \Rightarrow (1, 1)$is outside

To extend inside/outside determination to 3D, we use a 3 variable function $f(x, y, z)$ and 3D point p as input. For example, the unit sphere is $x^2 + y^2 + z^2 - 1 = 0$ or $f(p) = \|p\|^2 - 1$.

12.3 Normal to the Implicit Surface

The vector function, gradient $\bigtriangledown f = (\frac{\delta f}{\delta x}, \frac{\delta f}{\delta y}, \frac{\delta f}{\delta z})$ is a the normal of the surface at point $p(x, y, z)$.

Example 12.3

For the unit sphere:
$\bigtriangledown f(x, y, z) = (2x, 2y, 2z)$

- $\bigtriangledown f(1, 0, 0) = (2, 0, 0)$
- $\bigtriangledown f(0, -1, 0) = (0, -2, 0)$
- $\bigtriangledown f(0, 0, 0) = (0, 0, 0)$

If the gradient is zero, we cannot have a suitable normal.

Point p is a *singular* point if $\bigtriangledown f(p) = 0$,otherwise it is *regular*.

Example 12.4

- $p = (1, 0), x^2 + y^2 - 1 = 0$
- $p = (2, 0), x^2 + y^2 - 4 = 0$

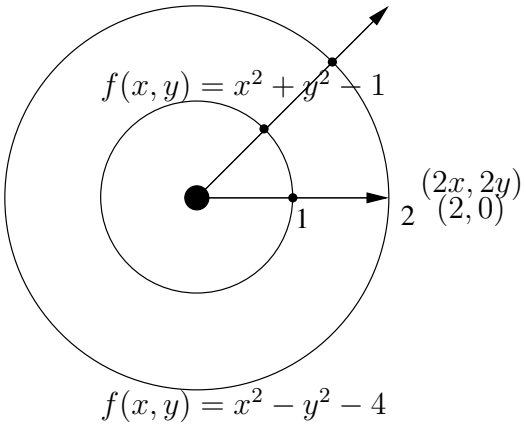


Figure 12.2: Normal to the Implicit Surface

To evaluate the gradient:

$$\nabla f(p) = (\frac{\delta f}{\delta x}(p), \frac{\delta f}{\delta y}(p), \frac{\delta f}{\delta z}(p))$$

$$\frac{\delta f}{\delta x}(p) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x, y, z)}{\Delta x}$$

$$\frac{\delta f}{\delta x}(p) \simeq \frac{f(x + \delta, y, z) - f(x - \delta, y, z)}{2\delta} \quad \text{for a small step size } \delta$$

Similarly for $\frac{\delta f}{\delta y}(p)$ and $\frac{\delta f}{\delta z}(p)$

12.4 Implicit Representation of Primitives

Each primitive is represented by a skeletal element. The iso-surface is all the points with the same distance (r). It is determined by using an energy function, e.g. $f(r) = \frac{1}{r^2}$.

12.4.1 Point

Assume we have a *hot* point at c. An implicit surface is formed by all points around c, with the same *temperature* (iso-surface). For all points p on the iso-surface, $\|p - c\| = r$. The implicit function here is the *temperature*. Set $f(r) \simeq \frac{1}{r^2}$. Therefore, the only important term for making this primitive is the radius r. Hence, the iso-surface is $f(r) = T$

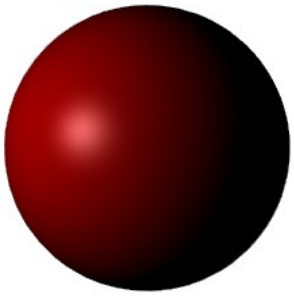


Figure 12.3: A point primitive

12.4.2 Lines

If the skeleton is a line (a hot bar), r is taken to be the closest distance to the line. Here the iso-surface is also $f(r) = T$

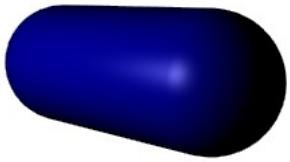


Figure 12.4: A line primitive

12.5 Blending

For simplicity, assume $f(r) \simeq \frac{1}{r^2}$. What happens for the combination of two primitives? We add the field strength at given points.

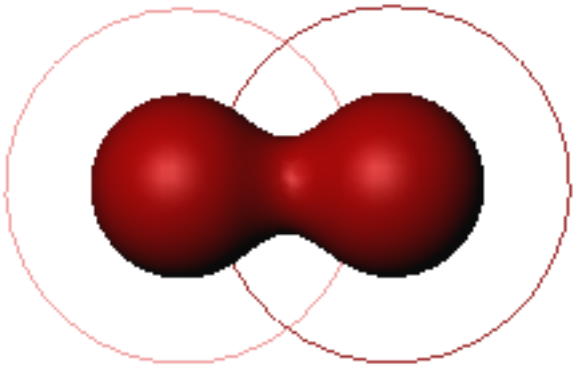


Figure 12.5: Blending of two points.

Example 12.5

For the 2D case

Given then two points $c_1 = (0,0), c_2 = (4,0)$, what is the total field strength at $p = (0,2)$ and $p = (2,2)$?

$$E(0,2) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

$$E(2,2) = \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$$

If each atom is represented by a primitive with the center $c_i$then each molecule's field strength at p is

$$\underbrace{\sum_i f(r_i)}_{\text{new function}} \qquad r_i = \|p - c_i\|$$

12.5.1 Better Blending

The simple energy function $f(r) = \frac{k}{r^2}$ usually results in some practical difficulties. For example, consider the blended surface as A in figure 12.6. If another primitive, say B, is added very far away from A, it still has an effect on A because the energy function is only zero at infinity.

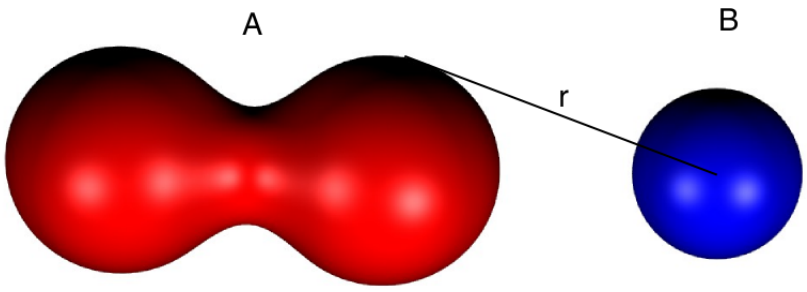


Figure 12.6: A very far new primitive such as B affects the shape of shape A. In addition, it causes many new computations for shape A.

To perform a local effect, we should change the energy function. This can be done by sigmoid functions. Figure 12.7 shows such a function. Note that we would like to have a smooth function at R. A smooth transition from $f(r)$ to zero results in a smooth iso-surface and smooth overlaps.

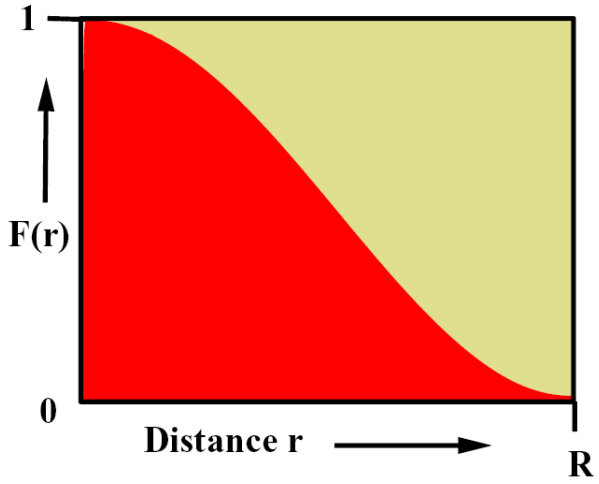


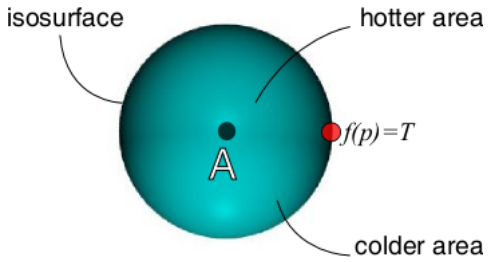
Figure 12.7: Sigmoid function that gives better blending

- Metaballs (Alaska University)

$$f(r) = \begin{cases} a(1 - \frac{3r^2}{b^2}) & 0 \leq a \leq \frac{b}{3} \\ \frac{3a}{2}(1 - \frac{r}{b})^2 & \frac{b}{3} \leq r \leq b \\ 0 & b \leq r \end{cases} \quad \text{quadratic B-spline}$$

- Soft objects (Wyvill)

$$f(r) = 1 - \frac{4r^6}{9R^6} + \frac{17r^4}{9R^4} - \frac{22r^2}{9R^2}$$



12.6 Boolean Operations on Implicit Models

We would like to have a simple method for boolean operations in implicit modeling (skeletal approach). We need a simple test for inside/outside of the combined surfaces. In the first step, we slightly change the notation. Assume the energy function f, center A and threshold T.

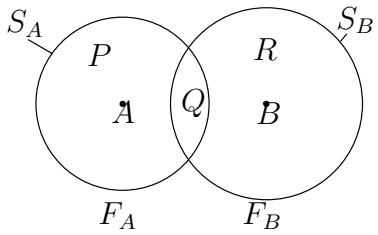
We introduce F as:

$$F(p) = f(p) - T$$

to have zero value on the iso-surface. To find the inside and outside of the solid

- $F(\text{inside}) > 0$
- $F(\text{outside}) < 0$
- $F(\text{on surface}) = 0$

With this new definition, having two primitives, we can find their intersection, union and difference.



	P	Q	R	S
F_A	+	+	-	-
F_B	-	+	+	-
max	+	+	+	- $(S_A \cup S_B)$
min	-	+	-	-$(S_A \cap S_B)$

To find the difference $S_A - S_B = S_A \cap S'_B = \max(F_A, -F_B)$

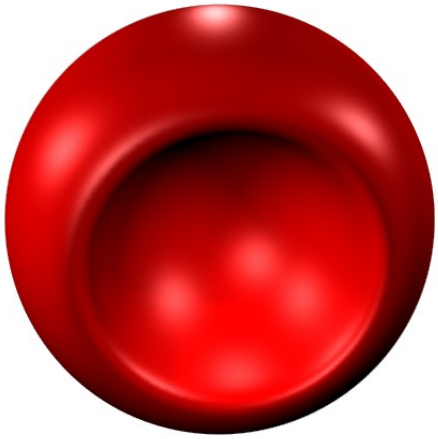


Figure 12.8: Difference of two point primitives

Since we use a try for boolean operations, the polygonization step is only needed for the root of the tree.

$$S = (S_A \cup S_B) \cap S_C \Rightarrow F(S) = \min(\max(S_{A,} S_B), S_C)$$