Subdivision Surfaces analysis and interrogation exercises: Discussion

Analysis

- Over what shape region is the basis function non-zero, of the ternary scheme whose non-zero coefficients form the pattern

```
      *  *
   *  *  *  *
*  *  *  *  *  *
*  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *
*  *  *  *  *  *
   *  *  *  *
      *  *
```

- The scheme is over the square grid, and must be 'neither', because the rows have an even number of entries.

The convex hull of the non-zero entries is octagonal, with sides alternately 1 unit and $3\sqrt{2}$ units long.

A ternary scheme cannot have any influence beyond a boundary 3/2 times as large as the convex hull of its entries.

Applying a copy of that boundary scaled down (to 1/2 times the size of that convex hull) to each of the entries at the top of the mask indicates that the influence cannot fill that theoretical boundary. The true boundary has a fractal form, consisting of pieces of line at 45 degrees, touching the scaled convex hull all the way along the sloping sides and in a cantor set along the horizontal and vertical sides.

- What is the precision set (constant, linear , quadratic ?) of the binary scheme whose mask is

$$
\begin{array}{ccccccccc}
 & & 1 & & 3 & & 1 & & \\
 & 3 & & 4 & & 4 & & 3 & \\
1 & & 4 & & 8 & & 4 & & 1 \\
 & 3 & & 4 & & 4 & & 3 & \\
 & & 1 & & 3 & & 1 & &
\end{array} \quad /14
$$

- The scheme is on a triangular grid and is binary. By symmetry it is primal, and so it has new points at the old ones and also at the midpoints of the edges, which come in three orientations. The stencils for the vertex vertices and the horizontal edges are

$$
\begin{array}{ccccc}
 & 1 & & 1 & \\
1 & & 8 & & 1 \\
 & 1 & & 1 &
\end{array} \quad /14 \qquad \text{and} \quad
\begin{array}{ccc}
 & 3 & \\
4 & & 4 \\
 & 3 &
\end{array}
$$

Summing these independently we find that 1+1+1+1+1+1+8 = 14 and 3+4+4+3=14, so that the scheme has constant precision. We only need to check one of the edge stencils by symmetry.

Multiplying by the linear field of old vertex values

$$
\begin{array}{ccccc}
-2 & & 0 & & 2 \\
 & -1 & & 1 & \\
-2 & & 0 & & 2 \\
 & -1 & & 1 & \\
-2 & & 0 & & 2 \\
 & -1 & & 1 &
\end{array}
$$

gives the new field

$$
\begin{array}{ccccc}
-14 & & 0 & & 14 \\
 & -8 & & 8 & \\
-14 & & 0 & & 14 \\
 & -8 & & 8 & \\
-14 & & 0 & & 14
\end{array} \quad /14
$$

which is not a linear field, and so the scheme has only constant precision.

- What is an appropriate univariate scheme to use for the boundaries of a surface defined by Loop's scheme, a binary scheme with mask

$$
\begin{array}{ccccccccc}
 & & 1 & & 2 & & 1 & & \\
 & 2 & & 6 & & 6 & & 2 & \\
1 & & 6 & & 10 & & 6 & & 1 \\
 & 2 & & 6 & & 6 & & 2 & \\
 & & 1 & & 2 & & 1 & &
\end{array}
\quad /16
$$

- Multiplying the mask by the extruded cardinal data set

$$
\begin{array}{ccccccc}
 & & 0 & & 0 & & 0 \\
 & 0 & & 0 & & 0 & \\
1 & & 1 & & 1 & & 1 \\
 & 0 & & 0 & & 0 & \\
 & & 0 & & 0 & & 0
\end{array}
$$

we get

$$
\begin{array}{ccccccc}
 & & 2 & & 2 & & 2 \\
 & 8 & & 8 & & 8 & \\
12 & & 12 & & 12 & & 12 \\
 & 8 & & 8 & & 8 & \\
 & & 2 & & 2 & & 2
\end{array}
\quad /16
$$

and so a suitable boundary scheme is $[2, 8, 12, 8, 2]/16$, or $[1, 4, 6, 4, 1]/8$, the cubic spline scheme. This is interesting, because the Loop scheme is a box-spline with quartic surface pieces, and yet it naturally gets cubic boundaries.

- Analyse the continuity of Loop's scheme at an ordinary vertex
- Spotting that the scheme is a box-spline says that it has quartic pieces meeting with C^2 continuity. However, it is worth also running through the eigenanalysis.

Let us set up an algebraic data-set for one coordinate, and see how it is refined.

We observe that every new vertex in or on the first ring of the data is defined by data in or on the second ring, and, indeed that no points outside the second ring influence their images after one refinement. We only therefore need to consider four types of point:

$$\begin{array}{ccccc} & & d & & \\ & b & & c & \\ a & & b & & d \end{array}$$

where a is the centre of the refinement.

Label the original vertex values with upper case letters, A denoting the central value and B_i denoting the value on the i^{th} spoke. Label the 'C' points with $C_{i+1/2}$, so that the zero subscript is always on the zeroth spoke.

Label the values after refinement with corresponding lower case letters.

Then the new values are related to the old by

$$\begin{aligned} 16a &= 10A + \sum_i B_i \\ 16b_i &= 6A + 6B_i + 2B_{i+1} + 2B_{i-1} \\ 16c_{i+1/2} &= 2A + 6B_{i+1} + 6B_i + 2C_{i+1/2} \\ 16d_i &= A + B_{i+1} + B_{i-1} + 10B_i + C_{i+1/2} + D_{i-1/2} + D_i \end{aligned}$$

If we apply the Fourier[3] transform that $B_i = \sum_\omega \hat{B}_\omega\, cos(i\omega)$ and $\hat{b}_\omega = \sum_i b_i\, cos(i\omega)$ these equations become

$$\begin{aligned} 16a &= 10A + 6\hat{B}_0 \\ 16\hat{b}_\omega &= 6A + \hat{B}_\omega(6 + 4\,cos(\omega)) \\ 16\hat{c}_\omega &= 2A + 12\hat{B}_\omega\, cos(\omega/2) + 2\hat{C}_\omega \\ 16\hat{d}_\omega &= A + \hat{B}_\omega(10 + 2\,cos(\omega)) + 2\hat{C}_\omega\, cos\,\omega/2 + \hat{D}_\omega \end{aligned}$$

[3] This is not a Fourier transform but the closely related Discrete Cosine Transform. The full Fourier uses $e^{ij\omega}$ in place of the cosines here. DCT is a useful short cut for manual calculations where we can rely on symmetry to give the second eigencomponents which are associated with negative ω values.

For $\omega = 0$, $cos\,\omega = cos\,\omega/2 = 1$, and these become

$$\begin{aligned} 16a &= 10A + 6\hat{B}_0 \\ 16\hat{b}_0 &= 6A + 10\hat{B}_0 \\ 16\hat{c}_0 &= 2A + 12\hat{B}_0 + 2\hat{C}_0 \\ 16\hat{d}_0 &= A + 12\hat{B}_0 + 2\hat{C}_0 + \hat{D}_0 \end{aligned}$$

which may be expressed as the matrix equation

$$16\begin{bmatrix} a \\ \hat{b} \\ \hat{c} \\ \hat{d} \end{bmatrix} = \begin{bmatrix} 10 & 6 & 0 & 0 \\ 6 & 10 & 0 & 0 \\ 2 & 12 & 2 & 0 \\ 1 & 12 & 2 & 1 \end{bmatrix} \begin{bmatrix} A \\ \hat{B} \\ \hat{C} \\ \hat{D} \end{bmatrix}$$

The matrix here has eigenvalues 1, 1/4, 1/8 and 1/16

For $\omega = \pi/3$, $cos\,\omega = 1/2$, $cos\,\omega/2 = \sqrt{3}/2$. We also lose all the A and a terms, so that the equations become

$$\begin{aligned} 16\hat{b}_\omega &= 8\hat{B}_\omega \\ 16\hat{c}_\omega &= 6\sqrt{3}\hat{B}_\omega + 2\hat{C}_\omega \\ 16\hat{d}_\omega &= 11\hat{B}_\omega + \sqrt{3}\hat{C}_\omega + \hat{D}_\omega \end{aligned}$$

which may be expressed as the matrix equation

$$16\begin{bmatrix} b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 8 & & \\ 6\sqrt{3} & 2 & \\ 11 & \sqrt{3} & 1 \end{bmatrix} \begin{bmatrix} B \\ C \\ D \end{bmatrix}$$

whose eigenvalues are 1/2, 1/8 and 1/16.

For $\omega = 2\pi/3$, $cos\,\omega = -1/2$, $cos\,\omega/2 = 1/2$. We again lose all the A and a terms, so that the equations become

$$\begin{aligned} 16\hat{b}_\omega &= 4\hat{B}_\omega \\ 16\hat{c}_\omega &= 6\hat{B}_\omega\, cos(\omega/2) + 2\hat{C}_\omega \\ 16\hat{d}_\omega &= 9\hat{B}_\omega + \hat{C}_\omega\, cos\,\omega/2 + \hat{D}_\omega \end{aligned}$$

so that the eigenvalues are 1/4, 1/8 and 1/16.

Finally, for $\omega = \pi$, $cos(\omega) = -1$, $cos(\omega/2) = 0$

$$
\begin{aligned}
16\hat{b}_\omega &= 2\hat{B}_\omega \\
16\hat{c}_\omega &= 2\hat{C}_\omega \\
16\hat{d}_\omega &= 8\hat{B}_\omega + \hat{D}_\omega
\end{aligned}
$$

has eigenvalues 1/8, 1/8 and 1/16.

Thus the spectrum has: an eigenvalue of 1 from $\omega = 0$, two eigenvalues of 1/2 from $\omega = \pm\pi/3$, two eigenvalues of 1/4 from $\omega = \pm 2\pi/3$, and one eigenvalue of 1/4 from $\omega = 0$. All other eigenvalues are smaller, and in fact equal to 1/8 or less.

Thus we have $\mu = \lambda^2$ as needed for bounded curvature.

In fact the eigenvectors associated with λ give as the natural configuration the regular grid, and those associated with μ give second order natural configurations which are quadratic functions.

- Analyse the continuity of Loop's scheme at an extraordinary vertex of valency 8
- This is more interesting.

The matrix for $\omega = 0$ is

$$
\begin{bmatrix}
10 & 6 & & \\
6 & 10 & & \\
2 & 12 & 2 & \\
1 & 12 & 2 & 1
\end{bmatrix} /16
$$

and for general $\omega > 0$ is

$$
\begin{bmatrix}
6 + 4\,cos(\omega) & & \\
12\,cos(\omega/2) & 2 & \\
10 + 2\,cos(/omega) & 2\,cos(\omega/2) & 1
\end{bmatrix} /16
$$

We need to check the eigencomponents for $\omega = 0, \pi/4, \pi/2, 3\pi/4, \pi$. The first and last of these give one component each, the middle three pairs of equal eigenvalues.

ω	$cos(\omega)$	$cos(\omega/2)$	eigenvalues
0	1	1	$1, 1/4, 1/8, 1/16$
$\pi/4$	$.7071$	$.9237$	$.55, 1/8, 1/16$
$\pi/2$	0	$.7071$	$6/16, 1/8, 1/16$
$3\pi/4$	$-.7071$	$.3823$	$3.17/16, 1/8, 1/16$
π	-1	0	$1/8, 1/16$

$$\lambda = 0.55$$

$$\mu_0 = .25 < .55^2 (= .304)$$

$$\mu_2 = .375 > .55^2$$

All other eigenvalues are less than λ^2. Thus for nicely symmetric data around the extraordinary point there will be a flat point where the curvature tends to zero, while for saddle-shaped data there will be a sharp point where the curvature is unbounded.

The next problem is to find an appropriate set of rules so that the eigenvalues behave, giving $\mu_0 = \mu_2 = \lambda^2$ exactly. To achieve this the coefficients need to depend on the valencies of the vertices to which they refer.

- Does Loop's scheme have lateral artifacts ?
- Look back at the boundary scheme question. The extruded data gets refined into a new data set still extruded, and so there are no lateral artifacts.

It is also possible to divide the scheme by $(1 + x)$ to give the quotient

$$\begin{array}{ccccccc} & & 1 & & 1 & & \\ & 2 & & 4 & & 2 & \\ 1 & & 5 & & 5 & & 1 \\ & 2 & & 4 & & 2 & \\ & & 1 & & 1 & & \end{array}$$

without any remainder, and so the scheme can be shown this way also to be artifact-free.

Interrogations

- Write an algorithm (in the style of the pseudocode in the notes and to the same level of detail) for the intersection of two surfaces by subdivision. Note that it will come out to rather more lines, as there is more to be covered.

- Here it is

```
Pointset:Srfint(Surface:S_A, S_B)
Surface:S_1,S_2
Begin
    If    Hull(S_A) overlaps Hull(S_B)
    Then If    Simple(S_A)
         Then If    Simple(S_B)
              Then Srfint(Approx(S_A),Approx(S_B)                  (1)
              Else S_1, S_2 := Halves(S_B)
                   Union(Srfint(S_A,S_1),Srfint(S_A,S_2))          (2)
              Endif
         Else If    Simple(S_B)
              Then S_1, S_2 := Halves(S_A)
                   Union(Srfint(S_1,S_B),Srfint(S_2,S_B))          (3)
              Else If    Hull(S_A) < Hull(S_B)                     (4)
                   Then S_1, S_2 := Halves(S_B)
                        Union(Srfint(S_A,S_1),Srfint(S_A,S_2))
                   Else S_1, S_2 := Halves(S_A)
                        Union(Srfint(S_1,S_B),Srfint(S_2,S_B))
                   Endif
              Endif
         Endif
    Else  ∅
    Endif
End
```

I did warn you that this would be quite long.

Notes:

(1) This is a special Srfint for intersecting facets.

(2) The Srfint here could be a sawn-of version which already knows that its first argument is simple, so it doesn't have to test again. The Union here is also a version that knows that the gap it has to fix up is the one between the two second arguments.

(3) Now the Srfint could be a simpler version which knows that its second argument is simple, and Union knows that the gap it has to fix up is the one between the two first arguments.

(4) The $<$ denotes a test on the maximum size of the two boxes.

- Write an algorithm for finding the first intersection of a given straight line (defined by a point Q and a tangent vector T) with a surface S by subdivision. 'First' means that the value of $[P - Q].T$ has the smallest positive value of all points P which are both on the line and on the surface. (It is the one which you see when your eye is at Q looking along T.)

- This is a nice example for subdivision because it has lots of tests for culling and also the requirement to find the nearest, so it needs a priority queue.

```
Point: NPTL(S,Q,T)
Point:R
Point,dist: best so far
Begin
    If    Hull(S) intersects [Q,T]
    and   Hull(S).T > Q.T
    Then push S into queue at Hull(S).T
          best so far = no point, very far away
          Until  queue is empty
          Do     Pop S from queue
                 If     Hull(S).T < best so far
                 Then If     Simple(S)
                      Then R = NPTL(Approx(S),Q,T)
                           Update best so far with R
                      Else S1, S2 = Halves(S)
                           If     Hull(S1) intersects [Q,T]
                           and    Hull(S1).T > Q.T
                           Then push S1 into queue at Hull(S1).T
                           Endif
                           If     Hull(S2) intersects [Q,T]
                           and    Hull(S2).T > Q.T
                           Then push S2 into queue at Hull(S2).T
                           Endif
                      Endif
                 Endif
          Enddo
          return best so far
    Else
          return no intersection
    Endif
End
```

There is a variant on this which checks whether the thing being pushed into the queue is going to be worth looking at when it comes off, in terms of being potentially nearer than the current best so far. This is the type of judgement which can only be made on the basis of experiments with realistic data. Making the extra test makes things slower, because it takes time. It might make them faster by keeping the queue slightly shorter.

- The silhouette curve is defined as the border curve between parts of a surface S which are lit by illumination from a given point Q and those parts which are not.

  What is the equation linking a point P of the surface, the given point Q and the surface normal N to S at P ?

  Write a pseudocode algorithm to compute the silhouette by subdivision.

- Oops. The question as intended was meant to ignore the possibility of one piece of surface shading another.

  The equation is just

$$[P - Q].N = 0$$

We are therefore going to need a normal hull to judge whether a piece of surface might contain a part of the silhouette curve. That test needs to compare the inner product of two hulls. This produces an interval. For two boxes this can be computed by just checking the corners. (Supplementary questions: prove that this is the case. Does this result generalise to hulls which are not just boxes ?). The hull of $[P - Q]$ is obtained by just displacing all of the hull of P by the coordinates of Q.

```
Pointset:silhouette(Surface:S,Point:Q)
Surface:S1,S2
Begin
      If      [Hull(S)-Q].Nhull(S) spans 0
      Then If      Simple(S)
           Then silhouette(Approx(S),Q)
           Else  S1, S2 := Halves(S)
                 Union(silhouette(S1,Q),silhouette(S2,Q))
           Endif
      Else  ∅
      Endif
End
```