Seamless Texture Mapping of Subdivision Surfaces by Model Pelting and Texture Blending

Dan Piponi and George Borshukov

MVFX, a division of Manex Entertainment

Abstract

Subdivision surfaces solve numerous problems related to the geometry of character and animation models. However, unlike on parametrised surfaces there is no natural choice of texture coordinates on subdivision surfaces. Existing algorithms for generating texture coordinates on non-parametrised surfaces often find solutions that are locally acceptable but globally are unsuitable for use by artists wishing to paint textures. In addition, for topological reasons there is not necessarily any choice of assignment of texture coordinates to control points that can satisfactorily be interpolated over the entire surface. We introduce a technique, *pelting*, for finding both optimal and intuitive texture mapping over almost all of an entire subdivision surface and then show how to combine multiple texture mappings together to produce a seamless result.

Keywords: Curves & Surfaces, Texture Mapping, Physically Based Animation

1 Introduction

Subdivision surfaces [5], [6] possess unique advantages over traditionally used NURBS surfaces that make them ideal for animated models. Complex models can be modeled with the efficiency of polygons and the smoothness of NURBS and other spline surfaces. Multi-limbed characters can be created using a single, contiguous mesh when using subdivision surfaces, something that is, in most cases, not possible using a single NURBS or other type of parametrised surface.

The most obvious advantage of using a single mesh to define, for example, a humanoid model, can be seen during the deformation or rigging process. Keeping closed seams and tangency between multiple NURBS patches, trimmed or untrimmed, becomes very difficult in areas where multiple joints exert influence over vertices of these multiple surfaces. With subdivision surfaces, the use of a single polygon mesh to define the head, torso, arms, and legs makes skeletal binding and weighting a much simpler and more effective process.

However, in order to apply surface detail with 2 dimensional texture maps [4],[12] the surface must be parametrised. Spline patches come with a natural parametrisation, but there is no such natural parametrisation on subdivision surfaces. In fact, as subdivision surfaces may have arbitrary topologies, there may in fact be no global parametrisation over the entire surface. In this paper we look at an approach to providing parametrisations for regions of subdivision surfaces and combining these into one continuous colour map on the surface. Much of our discussion also applies to polygonal models however subdivision surfaces have some extra complications making them more instructive to consider.

2 Background

Traditionally the way to represent the geometry of CG creatures is through the use of NURBS [9]. Unfortunately, there are many difficulties with arranging separate NURBS together so that they define a single smooth surface. The problems of arranging NURBS to abut against each other with a shared tangent are well known to those who have worked with commercial modelling packages. However, even when these problems have been solved there are difficulties with arranging textures to vary continuously over the seams between patches. Renderers need to sample texture values over a region in order to interpolate cleanly, anti-alias, and to calculate quantities like normals for correct lighting of displacement maps. As a result even when the geometry of NURBS join cleanly with a common tangent the surface detail can appear to have 'tears' along seams. Subdivision surfaces can provide a way to solve all of these problems resulting in texture and displacements that are continuous over the entire surface.

Subdivision surfaces are constructed by starting with a polygonal mesh that forms a manifold, M^0. A subdivision process is applied to this mesh to produce a new higher resolution mesh M^1 and then this process is iterated to form meshes $M^0, M^1, \ldots$ that better approximate the limiting mesh M^∞ [6], [7]. The subdivision scheme is chosen in such a way that under reasonable conditions the limiting surface is smooth.

In this paper we consider only Catmull-Clark surfaces but the techniques extend naturally to other subdivision schemes. (For clarity of exposition we consider mainly Catmull-Clark surfaces derived from quadrilateral meshes. After the first Catmull-Clark refinement any polygonal mesh becomes a quadrilateral mesh anyway [22].)

Unfortunately, assigning texture coordinates to a subdivision surface can be a difficult problem. Firstly, there is a purely topological problem. There is no way to assign texture coordinates to a sphere, for example, in such a way that the assignment is continuous and every point is assigned a unique pair of texture coordinates. Even worse - by the Borsuk-Ulam theorem [19] it is guaranteed that there are antipodal points on the sphere that are mapped to the same texture coordinates. So attempting to map a sphere using one global coordinate system is doomed to catastrophic failure. This problem is well known from cartography. Similar results hold for models with topologies other than that of the sphere. There are a number of ways to sidestep this problem:

- Use texture mappings with discontinuities

- Change the topology of the surface maps
- Use 3d solid textures rather than 2d textures
- Don't seek a global texture map but instead use multiple local textures

We consider these options in turn. It is difficult to ensure that texture mappings with discontinuities look seamless because renderers require continuous functions in order to calculate normals for bump mapping or sample areas for anti-aliasing (this is similar to the aforementioned problem with joining NURBS surfaces).

There are a number of approaches to changing the topology of the underlying objects. One approach to changing the model is to break it into individual pieces that can each be globally texture mapped - unfortunately this gives the problem of arranging that there are no seams between pieces. One can arrange for pieces to overlap but now we run into many of the same problems found with NURBS and additionally we lose the advantages of being able to work with a single model. Another approach can be best illustrated with a cylinder. A cylinder may be formed by rolling a sheet of paper, so that the opposite ends of the paper overlap. However, this *overwrapping* still requires breaking and rebuilding the original geometry and does not generalise easily to all topologies.

Using a solid texture for the entire model is inefficient and cumbersome. It is far easier for artists to paint 2d texture maps with familiar painting tools.

We chose the fourth approach because we found we could work uniformly with surfaces of any topology and always guarantee a final colour map that was completely smooth. In addition we found the task of managing multiple overlapping textures to be much easier than that of managing overlapping geometries because the former has no impact on the work of modellers or animators. (Note that from a mathematical viewpoint these approaches may be similar.)

We broke the problem down into two stages:

- Finding a way to texture regions on the model in a fashion that is intuitive for artists to work with and
- Joining together different regions in a completely seamless way

Assigning texture coordinates to a subdivision surface consists of two parts - assigning values to control vertices on the zeroth refinement of the mesh and a technique for interpolating these values over the surface. Therefore, we need to consider interpolation schemes.

Stam [22] describes a set of basis functions that parametrise patches on the subdivision surface corresponding to each quadrilateral in the zeroth refinement. This parametrisation defines a pair of coordinates on the surface to $[0,1] \times [0,1]$. Using these values we may bilinearly interpolate scalar values assigned at the vertices of the quadrilateral to define scalar fields over each patch and hence over the entire surface. Denote the set of functions on the subdivision surface obtained in this way by $L(M^\infty)$. Unfortunately, these fields are only guaranteed to be C^1 differentiable [1] on the interior of each patch. They have discontinuities in the first derivative along the edges and so texture coordinates chosen from $L(M^\infty)$ can result in texture maps that look unattractive. (Actually, these discontinuities are not necessarily a problem for single images, but they have a tendency to look bad when animated.)

According to [7] another technique for assigning texture coordinates to points on a subdivision surface is to consider scalar values assigned at vertices to be coordinates in an extra dimension. For example a vertex at (x, y, z) with texture coordinates (s, t) is considered to be a point (x, y, z, s, t). We then apply the subdivision process in the dimensionally extended space - 5 dimensional in this example. Each point in the subdivision surface (x', y', z', s', t') defines a point in 3D (x', y', z') with texture coordinates (s', t'). Under reasonable conditions these fields are C^1 over the entire subdivision surface. Denote the set of functions produced this way by $S(M^\infty)$. Our goal is to produce texture coordinate functions that lie in $S(M^\infty)$. It is useful to introduce the idea of the *domain of control* of a point p on M^∞. This is the set of control points that determine the value at p of a scalar field in $S(M^\infty)$. In Figure 1 we illustrate the domain of control of an example point - the vertices inside and on the boundary of the red region form the domain of control.

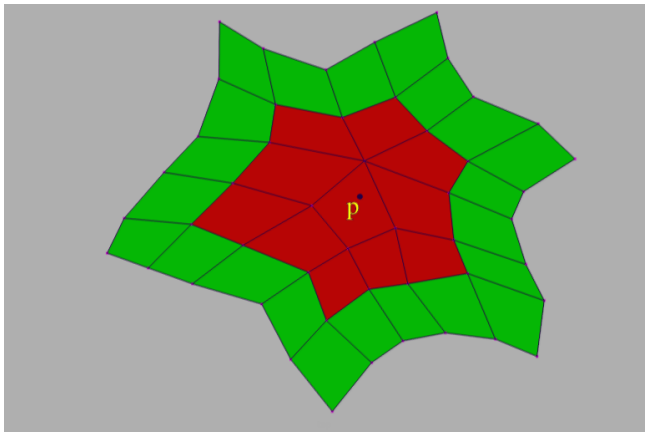

Figure 1: The domain of control of p

3 Texture Mapping Creatures

We now turn to the problem of assigning surface detail to a subdivision surface. Much of what we discuss applies equally well to polygonal models, however, the interpolation scheme that we use for subdivision surface texturing causes some extra complications that are not present in polygonal models. Typical methods included projection, solid textures [20] or two-part texturing [3]. These methods can be difficult to use with complex surfaces, so we chose to use 2 dimensional texture mapping using texture coordinate fields defined over the surface. We now look at the problem of choosing how to assign texture coordinates to control vertices. Even on a region that has no topological obstruction to being continuously texture mapped there is no *natural* choice of such texture coordinates. This makes the problem of choosing texture coordinates for subdivision surfaces similar to that for implicit surfaces or for surfaces derived from point coulds [13], [14],[15],[8]. One useful characteristic of any technique for assigning texture coordinates to a surface is to ensure that distances between points on the surface are represented accurately in the distances between the corresponding points in the texture space. (For surfaces with non-zero implicit curvature there will always be some distortion of distances [21].) This makes it much easier for artists to work with a 2 dimensional texture because the image they paint closely reflects how it will look in 3D. It also ensures uniformity in the look of the surface and efficiency in the storage of texture information. One approach to achieving this is to define a function that represents the extent to which distances are distorted by the texture mapping function. One can then find the mapping that minimises this parametric distortion. Ma [17] describes an algorithm that minimises a discrete

[1] C^0 means continuous. C^n means the nth derivative exists and is continuous.

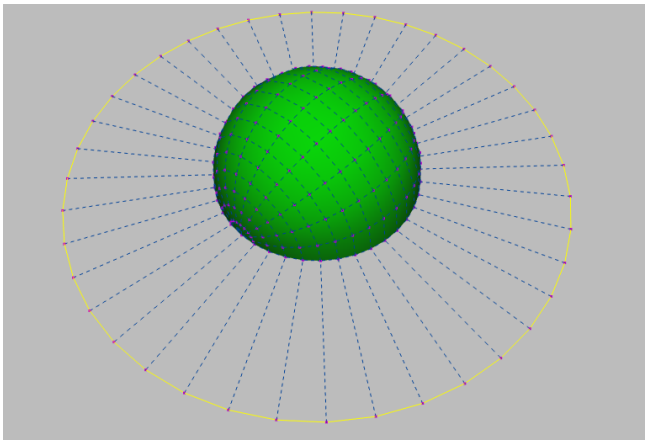

Figure 2: A pelting frame. The surrounding circle is the frame and the dashed lines represent the springs instantiated around the edge of the model and along the edges of the model.

approximation to a distortion measure on a grid. There are a number of different candidates for a measure of distortion such as the *Green-Lagrange deformation tensor* [18], [2]. An elementary approach is to consider the lengths of the edges of the polygon mesh in texture space and make the functional the sum of the squares of the deviations of the lengths of the edges.

$$E = \sum_{\text{edges } e} \frac{(L_e - l_e)^2}{L_e} \tag{1}$$

where L_e is the length of edge e and l_e is the distance between the two points in texture space. (Dividing by L_e provides a certain amount of invariance with respect to the details of the tesselation. Splitting an edge into two pieces by adding a new vertex in the middle will make no difference to the minimum value of E.) We can then use standard "black box" minimisation techniques to minimise this function.

Unfortunately, this is not necessarily adequate. This is a complex search problem with many local minima - especially for complex surfaces. Sometimes 'buckling' [18] can occur during minimisation. In addition, minima do not necessarily preserve approximate symmetries in the original model that artists involved in a production desire. Sometimes there are other factors that artists feel are more important than minimising distortion including aesthetic aspects of the resulting texture mappings. In short, many of the existing techniques provide mappings that are locally good but globally over the whole model the result is not suitable for production work - especially with complex models. We now consider a way to deal with all of these issues.

There are a number of different types of terms that can be added to Equation 1 in order to eliminate problems, for example, terms to prevent faces 'flipping' [18] or angular variation. We use a different approach. Equation 1 is formally identical to the total energy of a collection of springs that obey Hooke's law except that the dynamics is described only in 2D. Temporarily we will work with texture coordinates that lie in a 3D space. The resulting energy can now be minimised by deriving equations of motion from Equation 1, adding damping terms, and running a dynamics solver until a steady state is achieved [1]. Suppose we have a model that is topologically equivalent to a disk. We can add springs to the boundary of this disk with the opposing ends of these springs attached to a surrounding fixed frame. (See Figure 2 for an example using a hemisphere model.)

What this does is ensure that the edges of region to be textured

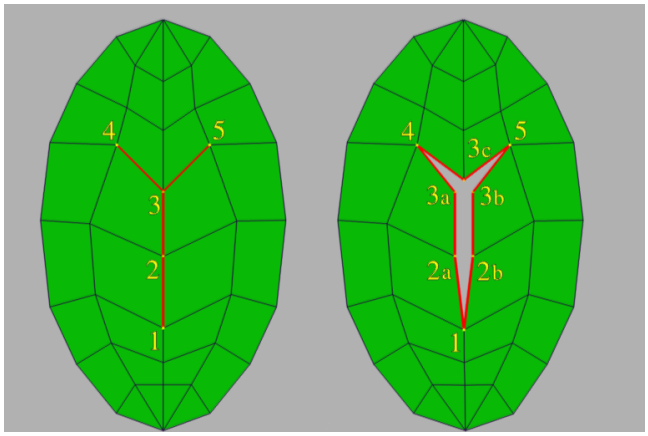

Figure 3: An example cut

are 'spread out' and it pulls the entire system out of local minima. The fact that we have extended the system to 3 dimensions is significant. If the orientation of a face in the 2D texture space is opposite to what it should be at the global minimum then it will often be caught in a local minimum because in order to 'flip' it over one or two vertices will have to actually pass through other edges in the polygon in order to reach the global minimum and hence pass through a higher energy state. In 3D no such flipping is necessary because each face can smoothly rotate to its correct orientation. With suitable choice of spring length and stiffness the system quickly achieves a state where it is almost flat and now texture coordinates may be applied by planar projection.

There is, of course, another way of looking at this: we are carrying out a simulation of a procedure traditionally used to stretch out animal hides for tanning. As a result the user can intuitively interact with the minimisation procedure and if starting with a creature model can arrange to end up with a texture that is in the form a pelt. We call this technique *pelting*.

The dynamics model we have just described is very elementary. In principle it could be made much more sophisticated - in particular we could add extra springs arranged so as to minimise shearing of the texture. However we have found that in practice that the strength of the pelting process is that it quickly finds a global texture mapping that is an excellent starting point for a subsequent local refinement. For this reason the precise details of the dynamics used are relatively unimportant.

4 The Pelting Procedure in Detail

For a model topologically equivalent to a sphere we need only make one cut in order to have a model that is topologically equivalent to a disk. Before the cutting procedure is described there is an important point to make: these cuts are for the sake of generating texture coordinates only. After our procedure is described we will show how texture coordinates can be transferred back to the original *uncut* model in a way that provides smooth colour mapping.

A cut is a collection of edges that form a connected tree. In our pelting tool the user indicates this set of edges and the software operates on the model by duplicating the edges and vertices as in figure Figure 3. Although we have illustrated point 2, say, as duplicated into distinct points 2a and 2b, these two points actually have the same location in 3D space. The cutting process changes the connectivity of the faces, edges and vertices only. The edges around the cut form a topological circle and we walk around this circle connecting each vertex (via a connecting spring) to a corre-

sponding point in the frame. We ensure that the stiffness of each spring and the spacing between the connecting points on the frame is proportional to the spacing between the other ends of the springs in the original model. A spring is then instantiated along each edge in the original model with stiffness proportional to the edge length. A mass is placed at each vertex and the outer frame is held fixed. (We chose to use an equal mass at each vertex. One could choose the mass of each vertex in such a way that the mass density per unit area of the model is approximately constant but in practice we found this unnecessary.) The user can then adjust various parameters such as spring stiffness and frame configuration and run the simulation in order to find a pelt that is optimal. For alternative topologies we can use more than one frame but the principles are the same. The user can make alterations to the geometry of the frame before or even during the minimisation. Our first test was with a model for a rat-like creature. (See Figure 4 (a,c).)

Once the model is stretched out flat the user can make manual adjustments if required and then texture coordinates can be applied using planar projection. In addition, further modifications may be carried out using algorithms more suited to local refinement. (See Figure 4 (b).) These assigned texture coordinates may now be transferred back to the original model. Note that as the vertices in the original cut were duplicated there is no unique way to transfer texture coordinates back to the original model at these points. We assign texture coordinates at these points completely arbitrarily (we see later why this is acceptable). At this stage what we have is the original model with texture coordinates assigned to all vertices. Unfortunately, in the region of the cut we will have an unsightly seam because (1) we have assigned arbitrary texture coordinates to some vertices and (2) we will be interpolating between texture coordinates that correspond to opposite sides of the flattened model.

5 Blending Textures

Fortunately, there is a way to deal with this problem. We can split up the surface into a number of pieces, each of which is a topological disk, and then generate texture coordinates for each piece. (Again we stress that although we are talking about splitting up the model our ultimate goal is to leave the original model intact.) If we split up the model into non-overlapping pieces we face the problem that the domain of control of points near the boundary may in fact lie in a neighbouring region. So we choose to work with a set of overlapping pieces.

There is a standard technique used in topology for dealing with functions on arbitrary topological surfaces that are defined on overlapping pieces. Suppose we have a manifold M and that we can express it as a union of open subsets [2]

$$M = \bigcup_{i \in I} U_i$$

(I is some indexing set.) We choose the sets U_i so that they are all topologically disks. Suppose we can find texture coordinates s_i and t_i on each U_i so that any function f_i on U_i can be written as $f_i(s_i, t_i)$ (Each U_i is called a *chart* and the whole collection is called an *atlas* [18], [19].)

Suppose we have a set of continuous functions $b_i: M \rightarrow R$ such that each b_i is zero outside of U_i and

$$\sum_{i \in I} b_i(p) = 1$$

[2] An open set is one where for every point there is a real ϵ such that it has a ball of radius ϵ around it that is completely contained in the set. As we are dealing with *compact* manifolds we are guaranteed that we can write M as a union of a finite number of sets. [19]

(Traditionally such a set of b_i is called a *partition of unity* [19] but we will call the b_i *blend functions*.) Given any continuous function f on M the functions $b_i f$ have the property

$$\sum_{i \in I} b_i f = f$$

and each $b_i f$ is zero outside of U_i. This gives a way to write any function on M as a sum of functions of s_i and t_i. Conversely, given a collection of continuous functions $f_i: U_i \rightarrow R$ we can construct the function

$$f = \sum_{i \in I} b_i f_i$$

and it is guaranteed to be continuous over the entire manifold. This gives a way to build a continuous function on the whole manifold out of the individual pieces. More generally by choosing blend functions that are themselves C^n we can build functions that are C^n out of individual pieces.

Our problem now is to build blend functions out of the functions contained in the sets $L(M^\infty)$ and $S(M^\infty)$ because these are the functions we are able to specify using control points.

Although it makes sense to use smooth functions for texture coordinates - when blending two different but similar functions it is acceptable to use functions in $L(M^\infty)$. This is because, as we will see later, we will be blending textures that are approximately equal on the overlap making the discontinuity in the derivative of the texture small.

We will now consider in detail how to construct blend functions for the pelting procedure above.

In our scheme we will use two regions: U_0 and U_1. U_0 will correspond to the pelt that we have described above and U_1 is a region that we will call the *patch*. We will have two sets of texture coordinates: (s_0, t_0) and (s_1, t_1). (s_0,t_0) are the coordinates derived by the pelting process above and (s_1, t_1) are discussed below. In this case we have only two blend functions $b_0(p)$ and $b_1(p) = 1 - b_0(p)$.

We discuss first how we construct control points for the blend function b_0 because this scheme determines how we choose the patch U_1.

In any overlap between the pelt and the patch regions we require a polygon over which both of the texture maps interpolate correctly because this gives a region over which we may smoothly switch from one texture map to another. Consider Figure 5. This represents a region spanning both sides of a cut (which is represented by ej). As discussed above we assign arbitrary values for the pelt texture coordinates to e and j and texture coordinates derived from the planar projection of the pelt at a, b, c, d, f, g, h and i (and similarly on the opposite site of the cut). The domain of control of the points on the interior of polygon $chid$ includes e and j so we know that the pelt texture coordinates interpolated here will have an arbitrary component and so will not represent the planar projection. The polygon closest to the cut that is a good representation is $bghc$. So in order to make the patch as small as possible we make the patch extend out as far as a and f so that on $bghc$ the patch texture coordinates are also good. We now define the blend function to be the function in $L(M^\infty)$ that has control value 1 at a, b, f, and g and zero on c, d, e, h, i, j. Over $bghc$ the blend function can be used to smoothly interpolate between the two textures and outside this region either one or the other texture mapping will be used. In this way we define a smooth transition between two texture maps by defining one extra scalar field and more importantly: not changing the geometry of the model in any way. In practice we design our textures so that the two different texture maps map into textures so that on $bghc$ they approximate each other.

When the topology of the faces is more complex than in this diagram a more complex definition is required. Call the set of vertices and edges in the cut C_0. Define C_n for $n > 0$ to be the loop of

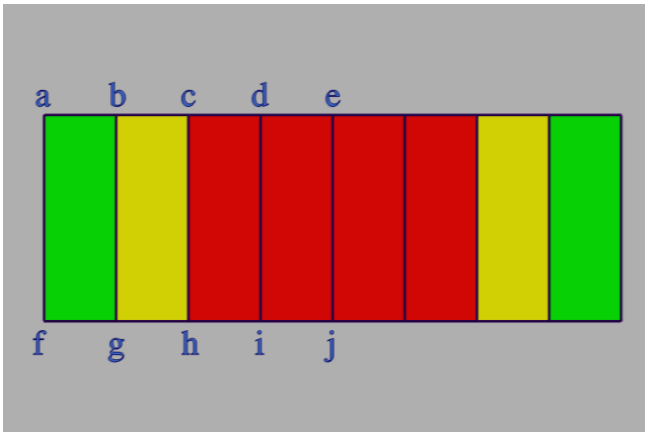

Figure 5: Regions of the blend function

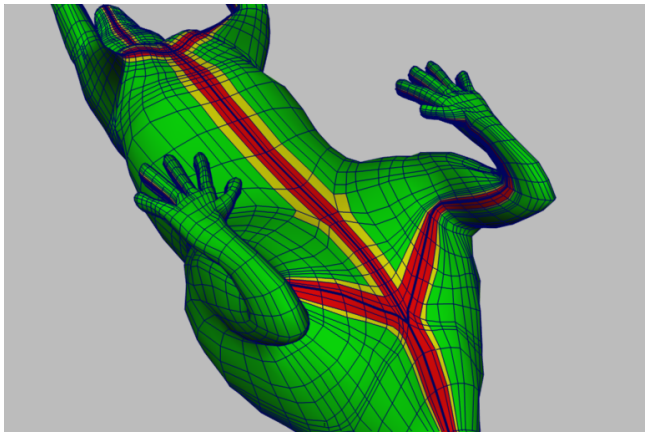

Figure 6: The regions of the blend function over the example rat model

edges and vertices that goes completely around C_{n-1}, containing the smallest area, that does not share any vertices with it. We are now in a position to define the patch - it is the region enclosed by C_4. Vertices within and including C_2 are given blend functon value 0 and those outside and including C_3 are given value 1. It can be seen that the region between C_2 and C_3 contains no points whose domain of control intersects with the cut. If we choose models with all vertices at least trivalent (at least three edges meeting at each vertex) and with every polygon a quadrilateral (after one Catmull-Clark subdivision all faces are quadrilateral) then there are guaranteed to be no isolated vertices between C_2 and C_3. This means that every vertex between and including C_2 and C_3 can be unambiguously assigned blend value 0 or 1 by the above method. On the other hand if other polygons are present we can interpolate blend values to vertices between C_2 and C_3 by any reasonable scheme - e.g. by taking the average of the values at nearest neighbours.

We can assign texture coordinates to the patch by detaching it from the full model and applying the pelting procedure to it. We then transfer the texture coordinates assigned to the patch back onto the full model. This mapping is extended to the entire model to produce (s_1, t_1) - assigning texture coordinates arbitrarily to points not in the patch. At this stage we now have two global sets of texture coordinates, and a global blending function that seamlessly blends between texture maps rendered using these coordinates in such a way that the regions with arbitrary texture coordinates are invisible.

The procedure for painting the models typically went as follows: once texture coordinates have been assigned to the model through the pelt int is roughly painted directly in a 3d paint package. The resulting low resolution texture map was then transferred to a 2d paint package where additional detail was applied. Using the texture coordinates derived from the pelting process we are able to render the patch region of the model after it has been flattened by pelting. As a result of the scheme described above the edges of the patch will be textured correctly. The final task of the artist is then to paint the small region of the patch in the immediate vicinity of the cut.

Implementing the texture blending is easily achieved with modern shaders [10] or with multipass texturing.

Although we have concentrated on the implementation details for subdivision surfaces a simplified scheme can be implemented for use with polygon models using linear interpolation of texture coordinates.

In Figure 5 we mark the region textured solely by the patch texture in red, that using solely the pelt texture in green and the region in which blending occurs in yellow. In Figure 6 we show how these regions extend over a large portion of our example model.

6 Results

We implemented the system to set up the dynamics using MEL scripting and C++ plug-ins within Alias-Wavefront's *Maya* and the dynamics solver used was the standard one in *Maya*. On an R10000 SGI O2 it took under a minute to find a good texture mapping for the 11000 polygon rat model and we were able to find good texture mappings for models such as the alien creature example with no difficulty. We used *Flesh* from Digits 'n' Art software to carry out local refinement on the final texture mappings and the initial 3d model painting. (See Figure 7.) We were able to implement the rendering using Pixar's *Renderman* which supports Catmull-Clark surfaces, multiple scalar attributes at control vertices, and both the $L(M^\infty)$ and $S(M^\infty)$ interpolation schemes. The resulting mappings were found to be very suitable for work with 2D paint packages such as Adobe *Photoshop* or with scans of real paint and brush. For the alien creature example we added geometric detail to the 6500 polygon model by using a displacement map painted using the same set of texture coordinates.

7 Conclusions and Future Research

We have found a way to texture map the entire surface of subdivision surfaces without modifying the model. We have used a dynamics solver to find optimal texture mappings and have used two techniques to find a good global solution: solving in 3D instead of 2D and using springs and a frame to impose boundary conditions. We have also introduced a scheme for blending smoothly between different texture mappings on a subdivision surface.

We would like to reimplement the dynamics more directly and efficiently (possibly replacing it with a more specialised statics solver that could work in real time) so that we can allow users to manipulate texture maps directly and easily as if they were rubber sheets [16]. We would also like to implement the entire blending process in an automatic way so that the user simply paints in 3D on a single model and the software automatically updates the (possibly multiple) texture map contributing to each point [11]. Although the technique has only been applied in the case of two regions it generalises to multiple regions by using multiple blend functions. We would like to find good ways to enable the user to define such regions with various types of cuts allowing extremely complex topologies to be dealt with efficiently.

One difficulty is with finding ways to make the cutting and blending scheme compatible with level of detail (LOD) techniques. LOD methods substitute low resolution proxies for models whenever

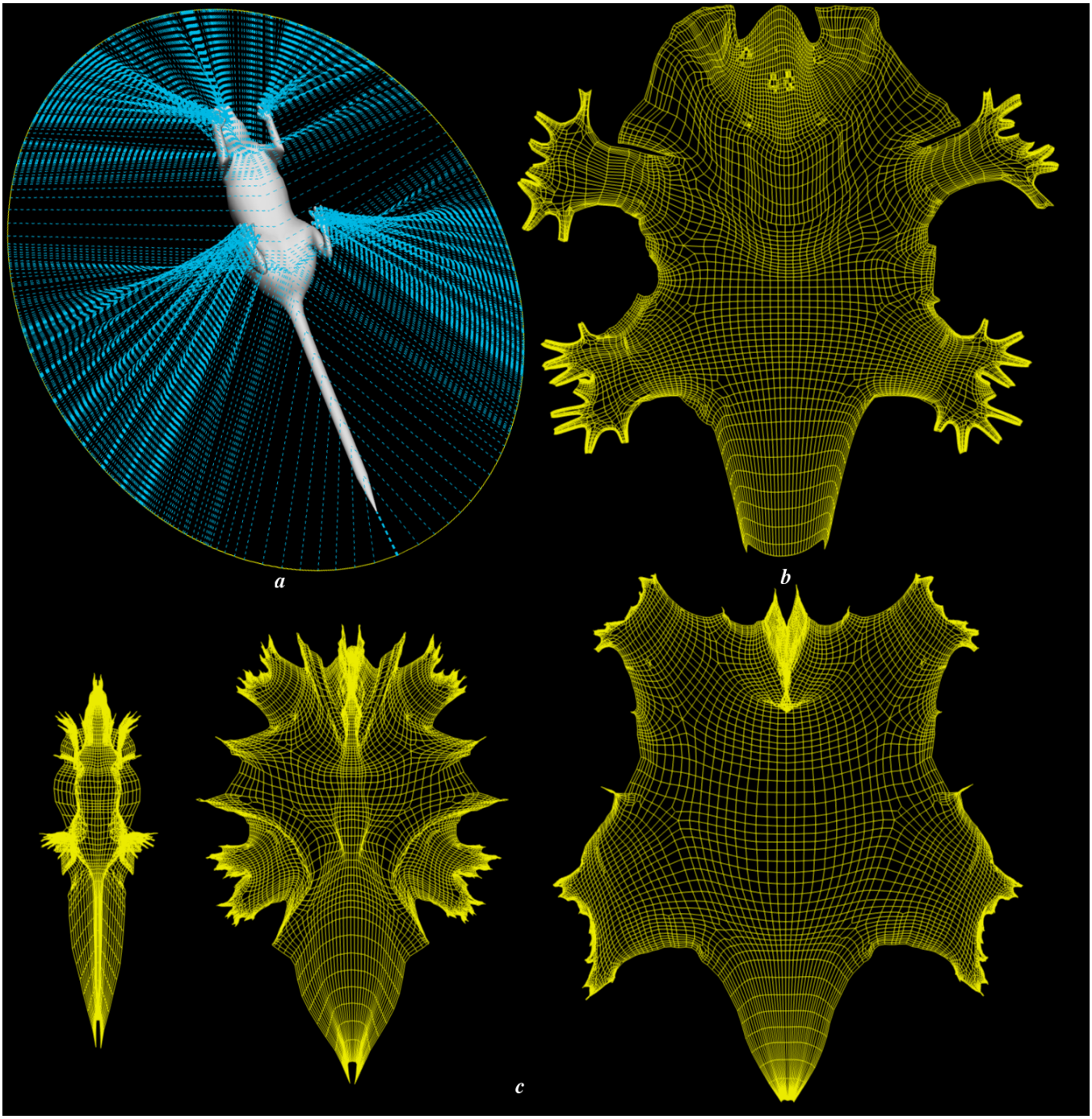

Figure 4: (a) The initial setup for the rat pelting (b) The final texture mapping after local refinement (c) 3 stages during the pelting simulation

high resolution details are not required. Unfortunately, the patch is itself a high resolution detail and so model simplification tends to remove it entirely. For very low resolution substitutes where the seam is too small to be noticeable we have used pelting combined with the earlier mentioned overwrapping approach in production.

We have had great success using pelting within a production environment. Animators and modelers have found it easy to generate texture mappings and 2D and 3D painters have found the mappings extremely well adapted to their needs. We believe the future of character animation lies with subdivision surfaces and pelting has played a major role in allowing us to move along this path.

8 Acknowledgements

Thanks to John Tissavary, Brett Hartshorne, Dan Klem, Mauricio Baiocchi among the artists who worked with and constructively criticised our work. A special thanks goes to Devorah Petty for both modelling and painting the alien model. Thanks also to Peter Plevritis for testing our approach in production.

References

[1] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. In *SIGGRAPH '92 Conference Proceedings*, Annual Conference Series, pages 303–308. ACM SIGGRAPH, July 1992.

[2] Chakib Bennis, Jean-Marc Vézien, and Gérard Iglésias. Piecewise surface flattening for non-distorted texture mapping. In *SIGGRAPH '91 Conference Proceedings*, Annual Conference Series, pages 237–246. ACM SIGGRAPH, July 1991.

[3] E. Bier and K. Sloan. Two-part texture mapping. *IEEE Computer Graphics and Applications*, pages 40–53, September 1986.

[4] J.F. Blinn and M.E. Newell. Texture and reflection in computer generated images. *Communications of the ACM,19,10*, pages 542–547, October 1976.

[5] E. Catmull. *A subdivision algorithm for the computer display of curved surfaces*. PhD thesis, University of Utah, December 1974.

[6] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design, 10(6):350-355*, 1978.

[7] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *SIGGRAPH '98 Conference Proceedings*, Annual Conference Series, pages 85–94. ACM SIGGRAPH, July 1998.

[8] Helaman Ferguson, Alyn Rockwood, and Jordan Cox. Topological design of sculptured surfaces. In *SIGGRAPH '92 Conference Proceedings*, Annual Conference Series, pages 149–156. ACM SIGGRAPH, July 1992.

[9] J. Foley and A. van Dam. *Computer Graphics: Principles and Practice*. Addison-Wesley, 1990.

[10] Pat Hanrahan. A language for shading and lighting calculations. In *SIGGRAPH '90 Conference Proceedings*, Annual Conference Series, pages 289–298. ACM SIGGRAPH, August 1990.

[11] Pat Hanrahan and Paul E. Haeberli. Direct WYSIWYG painting and texturing on 3D shapes. In *SIGGRAPH '90 Conference Proceedings*, Annual Conference Series, pages 215–223. ACM SIGGRAPH, August 1990.

[12] P. S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, pages 215–223, August 1990.

[13] Hans Køhling Pedersen. Decorating implicit surfaces. In *SIGGRAPH '95 Conference Proceedings*, Annual Conference Series, pages 291–300. ACM SIGGRAPH, August 1995.

[14] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *SIGGRAPH '96 Conference Proceedings*, Annual Conference Series, pages 313–324. ACM SIGGRAPH, aug 1996.

[15] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. In *SIGGRAPH '98 Conference Proceedings*, Annual Conference Series, pages 95–104. ACM SIGGRAPH, 1998.

[16] Peter Litwinowicz and Gavin Miller. Efficient techniques for interactive texture placement. In *SIGGRAPH '94 Conference Proceedings*, Annual Conference Series, pages 119–122. ACM SIGGRAPH, 1994.

[17] S.D. Ma and H. Lin. Optimal texture mapping. In *EUROGRAPHICS '88*, September 1988.

[18] Jérôme Maillot, Hussein Yahia, and Anne Verroust. Interactive texture mapping. In *SIGGRAPH '93 Conference Proceedings*, Annual Conference Series, pages 27–34. ACM SIGGRAPH, August 1993.

[19] C. R. F. Maunder. *Algebraic Topology*. Dover, 1996.

[20] D. Peachey. Solid texturing of complex surfaces. *Computer Graphics 19(3)*, pages 253–260, July 1984.

[21] M. Spivak. *A Comprehensive Introduction to Differential Geometry*. Publish or Perish, Inc., 1979.

[22] Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH '98 Conference Proceedings*, Annual Conference Series. ACM SIGGRAPH, 1998.

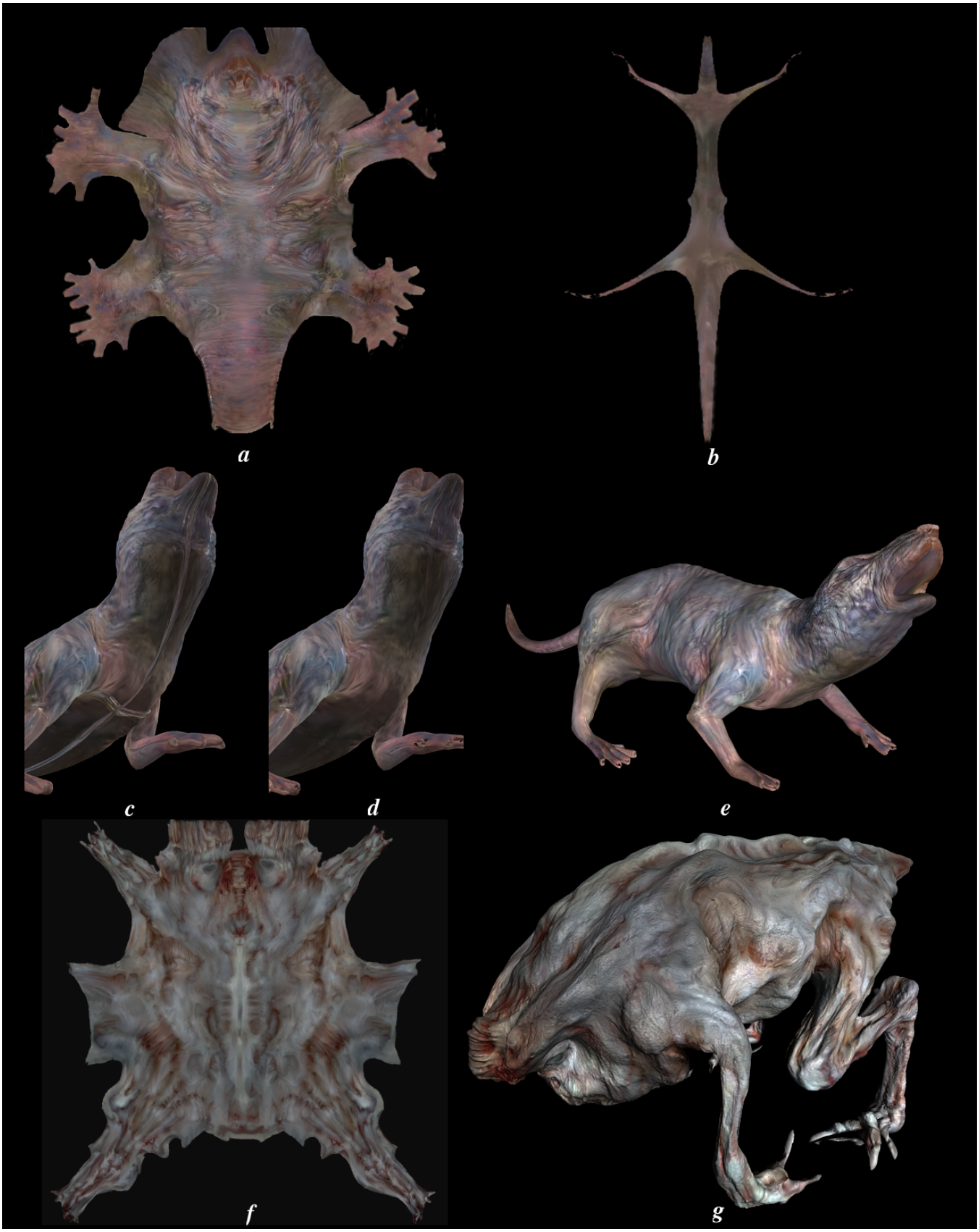

Figure 7: (a) Pelt texture for rat (b) Seam texture (c) Rendered rat using only pelt texture (note the seam) (d) Rendered rat using both textures (e) The final rat (f) Pelt texture for alien (g) Rendered alien (6500 polygons)