Shape-Preserving Parametrization of Genus 0 Surfaces

Hermann Birkholz
Scientific Assistant
Dep. of Computer Science, University of Rostock
Albert Einstein Str. 21
18059 Rostock, Germany
hb01@informatik.uni-rostock.de

ABSTRACT

The parametrization of 3-d meshes can be used in many fields of computer graphics. Mesh-texturing, mesh-retriangulation or 3-d morphing are only few applications for which a mesh parametrization is needed. Because, many polygonal surfaces are manifolds of genus 0 (topological equivalent to a sphere), we can apply a mapping, in which 2-d polar coordinates of a sphere can be directly transformed onto the 3-d coordinates of a polygonal object. In this paper we present a hierarchical mapping algorithm, that preserves the local surface properties. Our method consists of three main-steps. First, the mesh is simplified to a tetrahedron. Next, the tetrahedron will be transformed to a spherical surface in which the previous simplification process will be reversed on the surface of the sphere. Hereby, in every refinement step the new vertices are inserted and the resulting parametrization mesh is optimized to be barycentric. Finally, the resulting barycentric mesh is used as the basis for a shape-preserving optimization process. The efficiency of these method will be shown by using our parametrization algorithm on different 3-d objects.
Keywords: surface parametrization, texture, meshes, multi-resolution

1 Introduction

For the rendering of polygonal surfaces, 2-d texture maps and normal maps are commonly used. In order to find a mapping from the polygonal surface to these 2-d maps a parametrization of the 3-d object surface is needed. The vertices of the polygonal surfaces have to be transformed to a corresponding parameter space, with low distortion and no foldovers. For manifolds with borders the appropriate parameter space is a simple plane, while for a closed genus 0 manifold the natural parameter space is the surface of a sphere.

This paper presents a new technique to create a parametrization of genus 0 manifolds, that preserves the local properties of the surface. In our case it means that the shapes of local regions of the original model are similar to those in parameter space. Due to the preserved surface properties, this parametrization can be used together with multi-resolution models. In this case, the model and its parametrization are simplified simultaneously to produce an textured approximation of the original model. Examples can be found in the last section.

2 Previous Work

Methods for parametrization of 3-d meshes have been studied by a number of researchers. Most of them map the surface of the mesh to a plane by solving linear equation systems while others build a mapping between the mesh surface and the surface of a sphere and have to solve systems of nonlinear equations.

Tuette [Tutte63] introduced barycentric maps for embedding connectivity graphs. This approach maps the boundary of a given manifold with borders to a convex polygon in 2-d space and places each interior point in the barycentric center of its neighbors. This means each interior point p_i in parameter space is a linear combination of its equally weighted neighbors, given by

$$p_i = \sum_{j=0}^{n-1} w_{j,i} \cdot p_j$$

$$w_{j,i} = \begin{cases} \frac{1}{d_i} & , j \epsilon N(i) \\ 0 & , else \end{cases} , \quad (1)$$

where $N(i)$ denotes the point indices that are connected to p_i through edges (Figure 1) and d_i the number of indices in $N(i)$. When we put all border

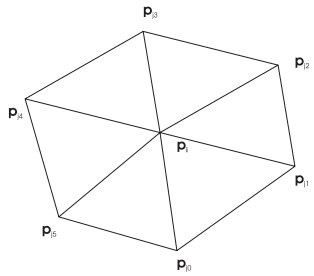

Figure 1: Vertex star for p_i

points at the end (indices m to M-1, where M is the number of vertices), this can be formulated as:

$$\begin{pmatrix} 1-w_{0,0} & -w_{1,0} & \cdots & -w_{n-1,0} \\ -w_{0,1} & 1-w_{1,1} & \cdots & -w_{m-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ -w_{0,m-1} & -w_{1,m-1} & \cdots & 1-w_{m-1,m-1} \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{m-1} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{i=m}^{M-1} w_{i,0} p_i \\ \sum_{i=m}^{M-1} w_{i,1} p_i \\ \vdots \\ \sum_{i=m}^{M-1} w_{i,m-1} p_i \end{pmatrix} . \quad (2)$$

By solving this equation system we get the positions for the interior points that match the barycentric condition. Because this mapping does not preserve any mesh properties except the connectivity, there are few applications for this technique (e.g. mesh morphing).

To produce mappings that preserve mesh properties, we have to calculate the weights $w_{i,j}$ in Equation 2 in another way. Floater [Floater97] introduced weights that reflect the real geometry. For this Floater maps each vertex star to a plane with a planar projection (scales the inner angle sum to 2π and preserves inner edge lengths as shown in Figure 2). Then, the projection weights must be computed for the outer points $\dot{\mathbf{p}}_{\mathbf{jk}}$, so that their linear combination results in the central point $\dot{\mathbf{p}}_{\mathbf{i}}$. Because there exist to many solutions, Floater shows how to find a proper set of positive weights, which are necessary in order to avoid overlappings that occurs in the case of negative weights.

Another approach to preserve mesh properties uses discrete harmonic maps that minimize the Direchlet Energy. Pinkall and Polthier [Pinkall93] discretized the problem for the weights of Eq. 2. This results in a angle-preserving mapping. A problem

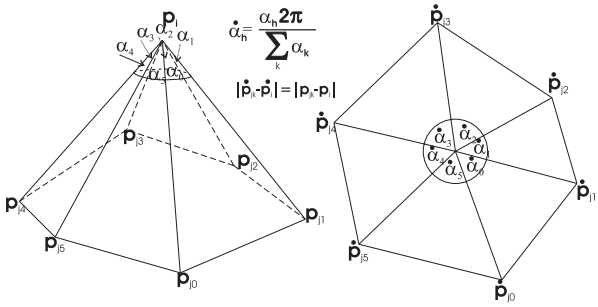

Figure 2: Planar projection

of this technique is occurrence of negative weights, and thus foldovers.

In [Hormann99] Hormann introduces a method to parameterize meshes using a simplify and refinement strategy. He starts to parameterize the coarsest version of its mesh and iteratively refines it while updating the parametrization.

The mentioned parametrization techniques until here can only be used for disc-like meshes, because we always have to define a border. Parameterized closed meshes cannot be defined, without partitioning them. Some other techniques allow us to parameterize closed genus 0 surfaces with a sphere surface as the parameter domain. The problem for this mappings is that the equation systems we have to solve are not linear. Therefore optimization algorithms have to be used to approximate the solution.

Shapiro [Shapiro98] in 1998 presented a mapping technique that is based on a simplification (vertex decimation) followed by a constrained refinement (vertex insertion) of the model. During the refinement all vertices are inserted in the mesh in a way, that the resulting mesh remains convex. The surface of such meshes can easily be mapped to a sphere, which is then used as parameter space for the original model. An advantage of this approach is its stability. Other techniques that work with optimization algorithms are not deterministic at all. There does not exists any closed genus 0 manifold that cannot be parameterized with this technique. But since none of the original surface properties are preserved during the parametrization process, there are hardly other applications for such parametrizations then mesh morphing.

Another way to build parameterizations of genus 0 manifolds has been shown by Alexa [Alexa99]. He simply projects the vertices of the model to a sphere from an interior point and starts an relaxation process. The relaxation iteratively places each param-

eter point in the center of its neighbor points (connected through edges) and project the result to the sphere surface. The process stops when all foldovers have disappeared. The problem is that this technique produces only a barycentric mapping, which does not preserve any surface properties. Furthermore the relaxation process does not always terminate. Therefore, several processes must be started with different interior initial points.

A technique to create an angle-preserving mapping on a spherical parameter domain was introduced by Haker [Haker00]. He transfers the nonlinear mapping problem to a linear disc parameter domain, solves it and projects the result to a sphere surface by means of stereo-projection. Because stereo-projection is not conformal for discrete mappings, he produces a mapping that is not actually angle-preserving.

In [Gotsman03] Gotsman describes some fundamentals of spherical parameterizations and presents a generalization of the barycentric coordinates for planar parametrization problems, that works in local tangent space and can also be used in the spherical case.

Praun [Praun03] introduced another hierarchical technique that works with stretch metrics and is used for geometry images of closed meshes. The metrics are constructed to reduce undersampling how it normally occurs into a parameter space.

A conformal parametrization method for closed manifolds with arbitrary genus is presented by Xianfeng Gu and Shing-Tung Yau [Gu03].

3 Shape-Preserving Mapping

Our aim was to produce a parametrization that is stable, fast, and preserves surface properties. We decided to use shape-preserving weights, because we can ensure that no foldovers occur and the resulting parametrization will preserve the angles and the shape of our input mesh. Such mappings will be well suited for textured multi-resolution meshes, since we can simultaneously simplify geometry and parameterize with low visible distortion.

Our approach for generating an shape-preserving parametrization of a genus 0 triangle mesh consists of the following steps. First, the original mesh is simplified by using an edge-collapse [Hoppe93] technique. The result of this simplification is a tetrahedron. This tetrahedron is mapped to the surface of a sphere as a uniform tetrahedron. Afterwards, the simplification process is reversed by iteratively inserting the vertices on the surface of the sphere. After each insertion the resulting mesh is optimized to be barycentric. In the final step we transform the resulting barycentric mapping to a shape-preserving one, with a similar optimization process.

Simplification. The objective of the simplification process is the step-by-step elimination of the edges of a polygonal object by using the edge collapse method (two child points collapse to one parent point). This process terminates if a tetrahedron is generated. The sequence of the edge collapses is not important because we do not try to preserve any properties while simplifying the mesh. We chose a collapse order depending on edge lengths (shortest edge first). The influence of different simplification metrics on the following refinement process has not yet been analyzed. Figure 3 shows three steps of simplifying a cube model.

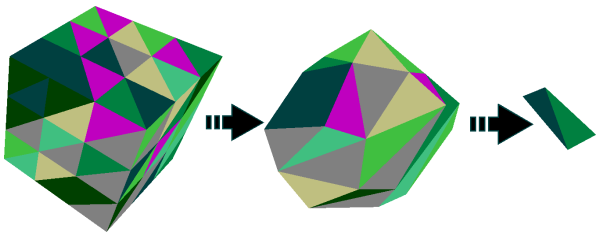

Figure 3: Cube model simplified to tetrahedron

Refinement. First the simplification result will be mapped onto the parameter sphere surface as an uniform tetrahedron. The following refinement process of this mesh is carried out by reconstructing the vertices corresponding to the reverse edge collapsing order which has been recorded before. First, the points are placed on the same position as their split parent. Then the position of all vertices on the sphere surface is optimized to form a barycentric mapping. Therefore, we iteratively optimize the position of a single point until it is the barycenter of the neighborhood (points connected through one edge). The optimization sequence is determined by the obtainable position improvement (the point whose position can be most improved is chosen first). The improved position is calculated and evaluated with the same functions used in the final optimization process, but with equal weights for all points.

New point positions that produce overlappings are not used during the optimization to except foldovers. This process ends when the possible improvement falls below a given threshold. We use 0.5% as a threshold value because the quality of the barycentric mesh is not relevant for the final

parametrization. This mapping is inevitably modified by the final optimization. At the end of the refinement process we have a barycentric mapping of our original mesh on a sphere surface. Figure 4 shows three steps from the refinement process. The parametrization stays barycentric after each point insertion.

Figure 4: Refinement of cube parametrization

Shape-preserving optimization. This final step transforms the barycentric mapping into a shape-preserving one. 4) This means, the angle between the edges and the ratio of the edge-lengths should be locally preserved as well as possible. For this reason we have to calculate weights as described in Floater [Floater97]. These weights are used to describe the relative positions of the points to each other in the parametrization.

In a plane, a linear combination of vectors like $p_i = \sum_{j \epsilon N(i)} p_j w_{j,i}$ minimizes the weighted sum of squared distances of all neighboring points p_j to p_i, where $N(i)$ denotes the indices of all points connected to p_i with an edge and $w_{j,i}$ is the weight of p_j relative to p_i.

$$\sum_{j \epsilon N(i)} \|p_j - p_i\|^2 w_{j,i} \rightarrow min$$

The global optimization target in a plane is to minimize by

$$\sum_{i=0}^{N-1} \left(\sum_{j \epsilon N(i)} \|p_j - p_i\|^2 w_{j,i} \right) \rightarrow min,$$

where the border points are fixed. The solution for the interior points can be found by equation 2 with the shape preserving weight matrix.

In our approach we formulate a similar problem for the sphere surface. For this, we have to minimize the weighted square sum of angles to all neighboring points in a vertex star.

$$ds(p_i) = \sum_{j \epsilon N(i)} \arccos(\mathbf{p_j} \cdot \mathbf{p_i})^2 w_{j,i} \rightarrow min$$

Because all points $\mathbf{p_k}$ are located on the unit sphere, the normalization dividend for the scalar product is normalized. As long as the entire vertex star is mapped to the same hemisphere, we can always find a unique solution for the minimization problem.

The interior point $\mathbf{p_i}$ cannot be calculated explicitly. Therefore we use an approximation algorithm. To find the approximation, it is easier to describe our respective point $\mathbf{p_i}$ with polar coordinates (α_i, β_i):

$$ds(\alpha_i, \beta_i) = \sum_{j \epsilon N(i)} \arccos \left(\mathbf{p_j} \cdot \begin{pmatrix} \sin(\alpha_i)\cos(\beta_i) \\ \sin(\beta_i) \\ \cos(\alpha_i)\cos(\beta_i) \end{pmatrix} \right)^2 w_{j,i}$$

$\mathbf{p_i}$ must be placed within the area of its vertex star before the iteration starts. So if it does not, we simply place it to the gnonomic center of the vertex star. After this initialization step, our method calculates by means of Newton-iteration the displacement vectors for α_i and β_i. The Iteration process terminates if the correct values of α_i and β_i are found. An additional problem is that β_i must be within the limits $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. By adding the displacement we can exceed these boundaries and would not find a solution. Therefore, we always rotate the parameter sphere until β_i is located on the equator. Then we calculate the displacement, add it to α_i and β_i and rotate the sphere back to its initial position. In this way, we always get the correct minimum.

If not all parameter points of a vertex star are positioned on the same hemisphere, the approximation algorithm can not always find the correct solution. In such cases, we use a simpler but less exact method to compute the position of $\mathbf{p_i}$ which is described in the following.

We iteratively project the neighbor parameter points $\mathbf{p_j}$ to the tangent plane of $\mathbf{p_i}$ via stereo projection, then we calculate the weighted sum of the projected points and reproject it to the sphere surface. The result is a coarse approximation for each iteration step. This method is always used to find a starting position for the optimization. If all points are found on the same hemisphere, the optimization continues with Newton's method. Because, we consider only meshes with a relatively high amount of vertices, all vertices are finally optimized with Newton's method.

With this technique we are able to optimize a single vertex star on the sphere surface, but our objective is the solution of the global problem

$$\sum_{i=0}^{N-1} \left(\sum_{j \epsilon N(i)} \arccos(\mathbf{p_j} \cdot \mathbf{p_i})^2 w_{j,i} \right) \rightarrow min. \quad (3)$$

This can be achieved by using a "Gauss-Seidel-like" method. In order to find the minimum in eq. 3, we iteratively select a single parameter point. After this step we compute its optimal placement by Newton-iteration until we obtain a satisfactory solution. The sequence of local optimizations steps depends on the obtainable improvement.

Therefore, we divide the local sum of the weighted square distances before and after an optimization step $ds(\mathbf{p_i})$, $ds(\mathbf{p_i^*})$. If $\frac{ds(\mathbf{p_i^*})}{ds(\mathbf{p_i})} < 1$ a local improvement is obtained and we can choose

$$min\left\{\frac{ds(\mathbf{p_i^*})}{ds(\mathbf{p_i})}\right\}$$

as the next parameter point to position.

After applying a local optimization we recalculate the possible improvements in the neighborhood and optimize the next parameter point. This algorithm terminates, when the local improvement is below a given threshold (for example, 0.995 for 0.5% improvement), or when no improvement occurs for the next n iterations. In figure 5 we show a barycentric mapping (left) in contrast to a shape-preserving mapping (right).

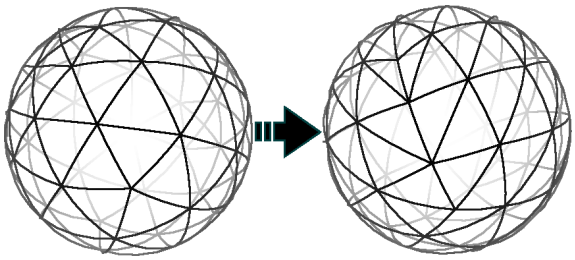

Figure 5: Parametrizations

4 Results

The described techniques have been implemented in C++ and been tested with several 3-d meshes, which where all genus 0 manifolds. We measured the time for the following parametrization steps: simplification step t_1, refinement t_2 and final optimization step t_3.

Model	Triangles	t_1	t_2	t_3	Minimum
cube5	300	63	1234	1500	15.39
lollypop	134	31	484	532	15.5
cross	280	62	985	1265	13.81

Table 1: Low resolution results

All steps are measured in ms. For the evaluation of the quality of the algorithm, we have used three low resolution models, which are shown in figure 6.

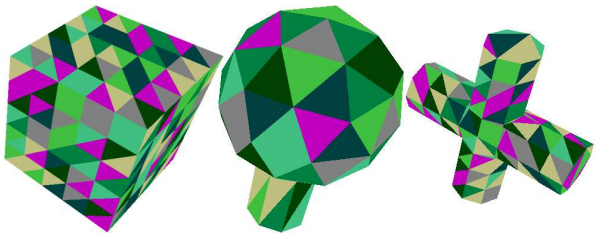

Figure 6: Low resolution models

All meshes have been textured with cube mapping (figure 7). The cube surface is projected to the parameter sphere surface from the common center of both objects.

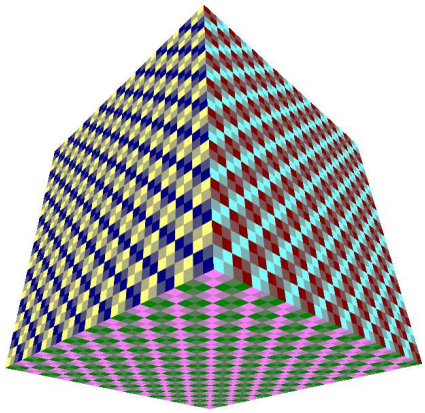

Figure 7: Cube texture

Figure 8 shows the parametrization and the textured meshes for the low resolution models. Table 1 contains the measured data for the test models. All tests have been executed on a Pentium 4 machine with 2 GHz and 1 GByte of Ram.

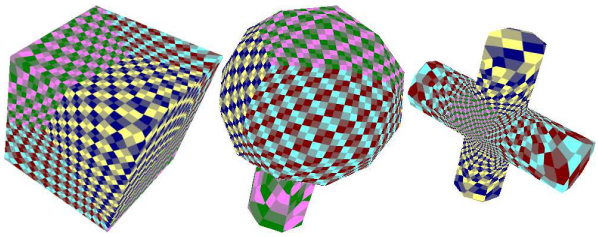

Figure 8: Textured low resolution models

As can be seen, the texture is mapped relatively smooth without any overlapping.

In figure 9 we show three high resolution models with more than 10k triangles. The *Dragon* and the model of the *Stanford-Bunny* are quite uniformly triangulated. That means the size and shape of the triangles do not differ much all over the surface. The triangulation of the *Bull* looks very irregular, because the sizes of the triangles are very variable over the whole surface.

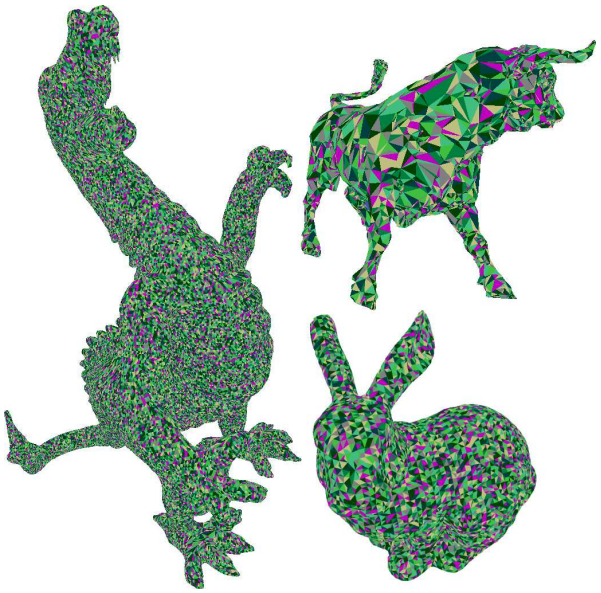

Figure 9: High resolution models

Table 2 shows the results for the textured versions of the high resolution models (fig. 10).

Model	Triangles	t_1	t_2	t_3	Minimum
Dragon	108586	42093	279390	245094	15.67
Bull	12396	3688	38797	64312	11.11
Bunny	16234	5156	62562	90469	15.48

Table 2: High resolution results

The texture is mapped very smooth. So we can apply it to simplified versions of the models, too.

In figures 11 to 13 we show simplified versions of the high resolution models. The number of triangles was decreased to 2325(Bull), 20460 (Dragon) and 3045(Bunny). Combined with normal-mapping we are now able to reach high simplification ratios with low visual effects. The problem of high undersampling (e.g. bunny ears) can be avoided with some kind of multi-resolution textures.

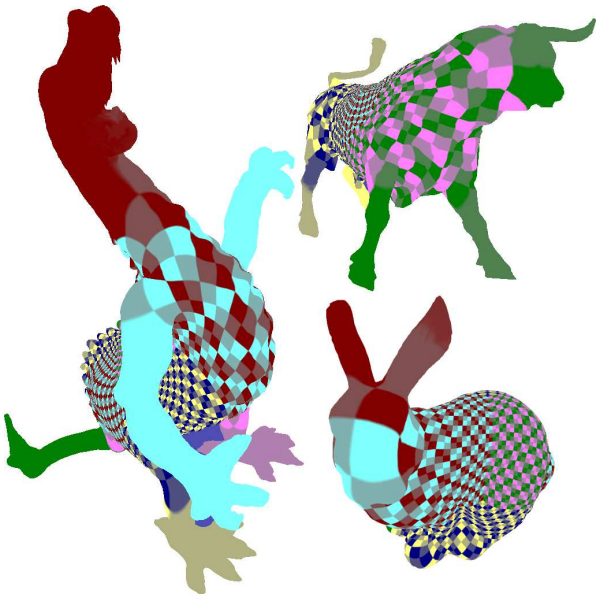

Figure 10: Textured high resolution models

5 Conclusion

In this paper a shape preserving parametrization technique was presented. This technique maps the surface of an genus 0 manifold to a sphere surface. For this purpose we introduce a method to find a approximated solution for the nonlinear equation system on the sphere surface. Furthermore, we showed that due to the preservation of angles, the resulting parameter space can be shared by miscellaneous triangulated versions of the same model.

References

[Tutte63] Tutte, W. T. *How to draw a graph.* Proc. London Math Soc. 13 (1963)

[Floater97] Floater, M. S. *Parametrization and smooth approximation of surface triangulations.* Comp.Aided Geom. Design (1997)

[Pinkall93] Pinkall, U. and Polthier, K. *Computing discrete minimal surfaces and their conjugates.* In Experimanetal Mathematics (1993)

[Hormann99] Hormann, K., Greiner, G., Campagna, S. *Hierarchical Parametrization of Triangulated Surfaces.* Vision, Modelling, and Visualization (1999)

[Shapiro98] Shapiro, A. *Polyhedron Realization for Shape Transformation.* Proc. of the fifteenth an-

nual symposium on Computational geometry (1998)

[Alexa99] Alexa, M. *Merging polyhedral shapes with scattered features.* The Visual Computer (2000)

[Haker00] Haker, S., Angenent, S., Tannenbaum, A., Kikinis, R., Sapiro, G., and Halle, M. *Conformal surface parameterization for texture mapping.* IEEE TVCG 6 (2000)

[Gotsman03] Gotsman, C., Gu, X., Sheffer, A. *Fundamentals of Spherical Paramerization for 3D Meshes* In Proceedings of ACM SIGGRAPH (2003)

[Praun03] Praun, E., and Hoppe, H. *Spherical Parametrization and Remeshing.* In Proceedings of ACM SIGGRAPH (2003)

[Gu03] Xianfeng Gu, Shing-Tung Yau *Global Conformal Surface Parameterization.* ACM Symposium on Geometry Processing (2003)

[Hoppe93] Hoppe, H. *Mesh optimization.* In Proceedings of ACM SIGGRAPH (1993)

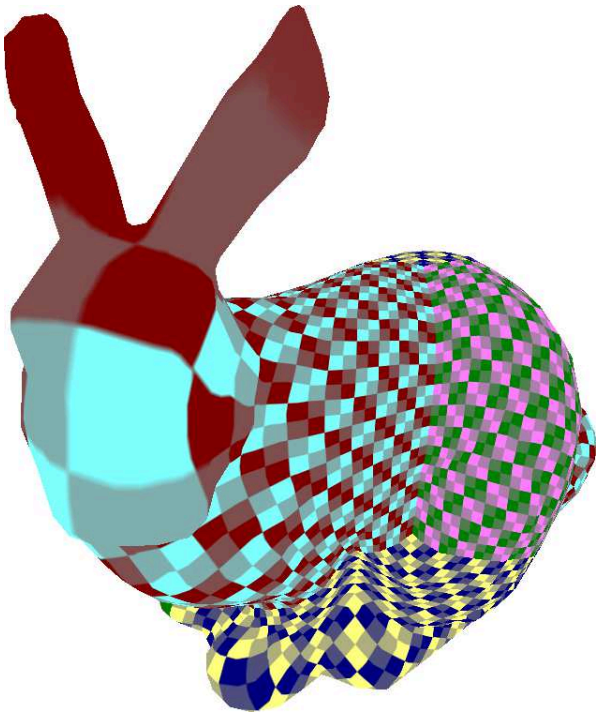

Figure 11: Textured low resolution bunny

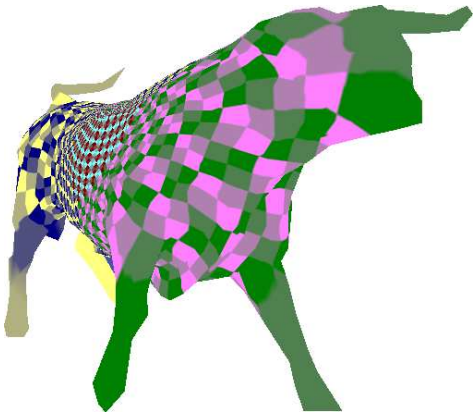

Figure 12: Textured low resolution bull

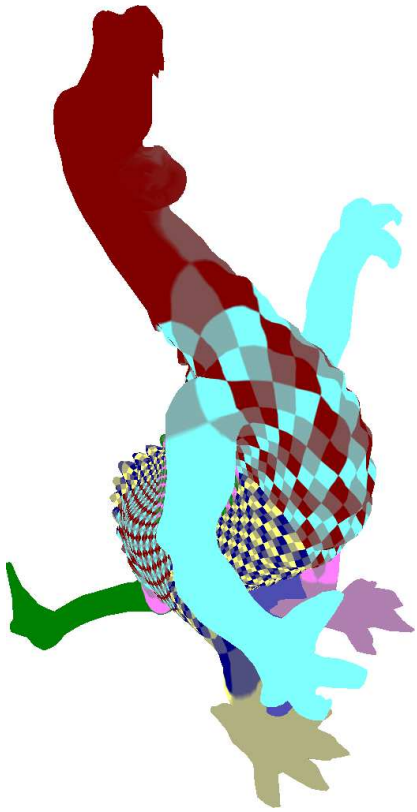

Figure 13: Textured low resolution dragon