Simplified Gamut Boundary Representation using Mesh Decimation

Arne M. Bakke and Ivar Farup
Gjøvik University College
Gjøvik, Norway

Abstract

Gamut boundary determination is an important step in device characterisation and colour gamut mapping. Many different algorithms for the determination of colour gamuts are proposed in the literature. They vary in accuracy, computational efficiency, and complexity of the resulting triangulated gamut surface. Recently, an algorithm called uniform segment visualization (USV) was developed. The gamut surfaces produced by the USV algorithm is more accurate than the ones produced by the the segment maxima algorithm, while at the same time, they are significantly simpler than the ones produced by the somewhat more accurate modified convex hull. In this paper, we propose a new method. First, an accurate gamut boundary is computed using the modified convex hull. The resulting surface is then simplified using an established mesh decimation technique. This results in surfaces that are significantly more accurate than the ones produced by the USV algorithm at a comparable complexity.

Introduction

A colour gamut is the set of all colours that can be produced by a given device or that are present in a given image. Although these sets are in principle discrete, gamuts are most often represented as volumes or blobs in a 3D colour space using a *gamut boundary descriptor*. When images are to be reproduced between different devices, the problem of gamut mismatch has to addressed. This is usually referred to as *gamut mapping*. There is a vast amount of literature about the gamut mapping problem. Fortunately, much of this was summarised by Morovic and Luo in 2001 [1].

Spatial gamut mapping has become an active field of research in the recent years [2, 3]. In contrast to the conventional colour gamut mapping algorithms, where the mapping can be performed once and for all and stored as a LUT, e.g., in an ICC profile, the spatial algorithms are image dependent by nature. Thus, the algorithms have to be applied for every single image to be reproduced, and make direct use of the gamut boundary descriptors many times during the mapping process.

There are many algorithms available that can be used to calculate an approximation to the color gamut of an image or a device. The gamut boundaries that are constructed can then be utilizied for the purpose of gamut mapping or comparison of gamut volumes. There are several advantages in having a compact description of the gamut surface, e.g., reducing the computation time of the algorithms using the gamut surface. The reduced surface complexity makes it possible to use less data to describe the gamut, which can be a factor when embedding gamut information in profiles. Perhaps more importantly, calculations of intersection points between the gamut surface and lines can be done in less time, improving the performance of gamut mapping algorithms. Any inaccuracies in the gamut surfaces might introduce errors in the result of the gamut mapping, possibly affecting the analysis of the results. Therefore, having a simple but accurate gamut surface is the goal of our work.

Bakke et al. [4] compared many methods for the determination of gamut boundaries for accuracy on simulated device data. It was found that the modified convex hull algorithm proposed by Balasubramanian and Dalal [5] consistently provided the most accurate gamut surfaces. However, the resulting gamut surfaces can consist of very many triangles, slowing down the gamut mapping algorithm. To address this, an algorithm called uniform segment visualization (USV) was recently proposed [6]. It was based on combining the modified convex hull algorithm and a sphere tesselation technique for uniform segmentation of the color space and is described in more detail below. The USV algorithm results in surfaces that can be significantly simpler than the ones provided by the modified convex hull at the price of lowering the accuracy somewhat. The number of segments can be determined by the user, but he still has to deal with the trade-off between accuracy and complexity.

Here, we propose an alternative approach. First an accurate gamut boundary is computed using the modified convex hull algorithm. The resulting surface is then simplified using an established mesh decimation technique. There are several mesh decimation algorithms that can be used for this purpose. Previous surveys have shown that an algorithm that performs mesh simplification using a local optimization criterion to collapse triangle edges into points performs well for generic triangulated surfaces [7]. We investigate the performance of this mesh decimation technique when applied to color gamuts, and compare the results to the USV algorithm for gamut surfaces having a varying number of triangles. The surfaces are compared to a reference gamut, and the differences are quantified using a previously proposed metric for gamut differences [4]

The paper is organised as follows. First, the most relevant establised GBD algorithms are described in more detail. We then look at mesh decimation techniques and our choice of simplification algorithm. We describe our proposed method, and look at implementation details. The resulting gamut boundary accuracies of the new method and the USV are compared for five devices, and the results are presented and discussed.

Established GBDs

Our goal is to develop a new GDB algorithm and compare it to the USV algorithm. For reference, we present also the modified convex hull algorithm and the segment maxima algorithm, since they both appear as important parts in the two compared algorithms, and since the modified convex Hull is used for generating the reference gamut for the comparisons of accuracy. A reference gamut with approximately 100 000 surface points is shown in Figure 1.

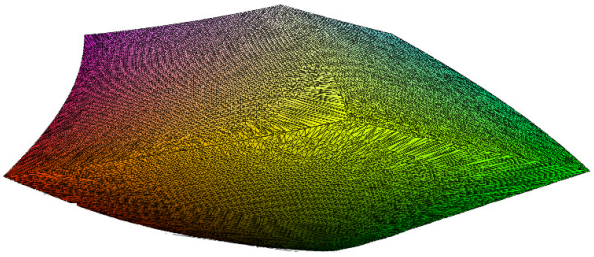


Figure 1. *Reference gamut of Hitachi monitor with approximately 100 000 surface point. The gamut surface was produced by the modified convex hull algorithm.*

Modified Convex Hull

One of the methods in use is to find the convex hull of the data points by using, e.g., the quickhull [8] algorithm. This results in a convex approximation of the gamut [9, 10] that does not follow the concavities typically found in both device and image gamuts when represented in colour spaces typically used for gamut mapping. An effective method to deal with this deficiency was introduced by Balasubramanian and Dalal [5]. They suggested to pre-process the data points using a non-linear gamma function based on the distance from the color to a center point within the gamut, before the convex hull is calculated from the altered points. By varying the γ parameter from 1 to 0, the detail level of the gamut can be increased by making the object more concave. A γ value of 1 leaves the points unchanged before the convex hull is applied, while smaller parameter values make the pre-processed data more convex. In the limiting case $\gamma = 0$, all the data points are mapped to a spherical surface and are thus included in the convex hull. With an optimal choice of γ, the final gamut boundary will closely follow the perceived surface of the data points. Previous work [4] has suggested $\gamma = 0.2$ as a reasonable value for a variety of gamuts and data sets, and that with a good choice of γ, the modified convex hull is consistently the most accurate algorithm for genererating gamut surfaces from generic data points.

Segment Maxima

Segment maxima [11] is another method used to find the gamut boundary. It starts by performing a subdivision of the color space into segments based on the spherical coordinates of the colors relative to a chosen centre, e.g., the centre of the gamut or the centre of the colour space. Each segment represents a uniform interval of spherical coordinates (polar and azimuth). For each segment, the color with the largest radius from the color space or gamut center is stored. These points can then be triangulated by taking advantage of the inherent neighbor structure of the segments.

There are, however, some deficiencies to this approach. First, the higher the number of segments, the higher the probility of empty segments. Since the triangulation of the final surface is based upon the regularity of the chosen points, there has to be a point in every segment. Thus, an intricate interpolation algorithm has to be used. Secondly, the segment maxima method results in a gamut where the surface is sampled more densely near the top and the bottom due to the uniform quantisation of the spherical angles. Lastly, there is no guarantee that all of the data points are inside the resulting closed surface. Despite these shortcomings, segment maxima is still much in use.

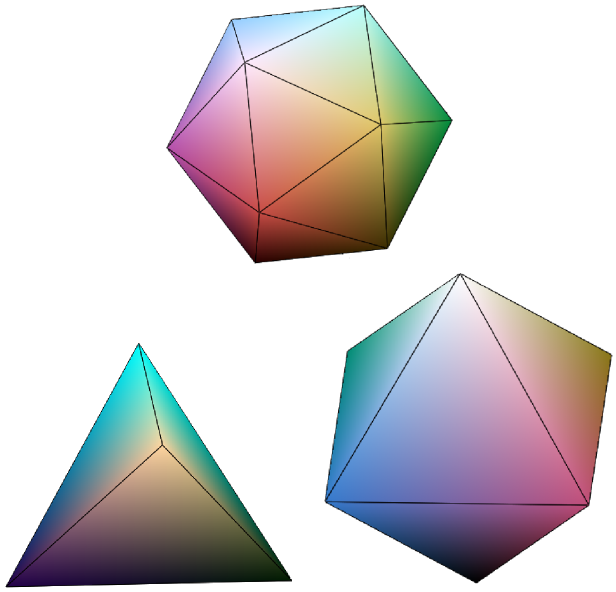


Figure 2. *The objects used as a basis for sphere tesselation: Icosahedron, tetrahedron, and octahedron*

USV

The USV algorithm was developed by Bakke et al. [6] in order to solve two of the three problems of the segment maxima algorithm: the inhomogeneity of the sampling points and the need to create artificial points by interpolation. It is a well known fact that tessellation of a sphere by uniform subdivision of spherical coordinates results in a surface with highly varying size of surface segments. Alternative solutions [12] are often used when working with sphere appropximations, such as when drawing a 3D sphere approximated by polygons.

The USV method computes the gamut boundary from any arbitrary selection of data points in CIELAB or a similar color space, given that a sufficiently dense sampling of the entire gamut volume has been performed. First, a triangle-based approximation of a sphere is computed using a well-known tesselation technique. A basic triangle-tesselated geometric object is generated, e.g, a tetrahedron, octahedron or icosahedron (Figure 2). A subdivision of the surface triangles is performed as shown in Figure 3. Each triangle is subdivided into 4 smaller triangles recursively, until the desired number of triangles is achieved. The resulting vertices are then projected onto a spherical surface, giving a close approximation of a sphere with a more uniform triangle size than uniform subdivision of spherical coordinates. Figure 4 shows the result of running the USV algorithm with approximately 10 000 segments on the reference gamut from Figure 1.

Upon construction of the tesselated sphere, a data structure is generated that includes the neighborhood connectivity information for the triangles. Each surface triangle defines a segment around the chosen center, represented by the tetrahedron given by the triangle vertices and the center point.

Figure 5 shows a surface triangle as solid lines, while the tetrahedron defined by the triangle and the center point is illustrated using dashed lines. Each pair of vertices P_0, P_1, and P_2 define an edge of the triangle. The tetrahedron is surrounded by

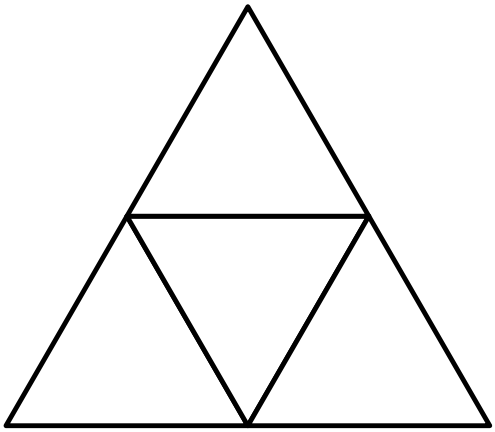


Figure 3. *The subdivision of a triangle*

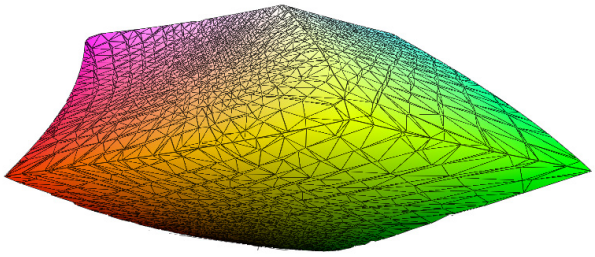


Figure 4. *The result of running the USV algorithm on the reference gamut from Figure 1 with approximately 10 000 segments.*

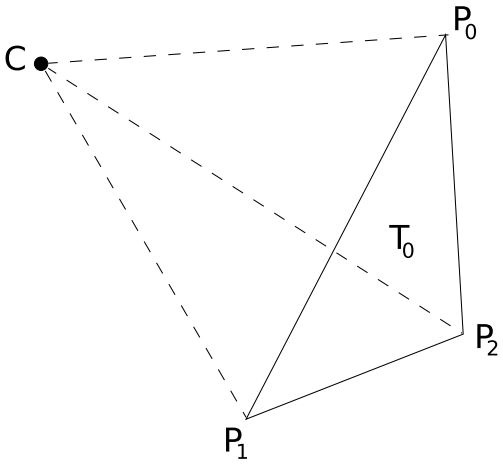


Figure 5. *The surface triangle T_0 and the center point C. The center point and each pair of the triangle vertices define planes used to check whether a point is inside the segment.*

three other tetrahedra, each sharing a different edge of the triangle.

The data points belonging to the gamut are then processed by finding the enclosing segment for each point. In order to determine which segment encloses each data point, a walking algorithm was used. When the enclosing segment has been located for each point, the length of the radius from the center point to the data point is calculated. Similarly to the segment maxima method, only the single maxima point per segment having the largest radius from the center point is kept for the next part of the algorithm.

When all the input points have been processed, some of the segments may be empty. Instead of inserting artificial interpolated points like the segment maximat method, USV directly utilises the modified convex hull algorithm with $\gamma = 0.05$ on the maxima points from the non-empty segments to create the final surface triangles constituting the gamut boundary. This small value of γ is chosen to ensure that all the points are included in the surface. One could correctly argue that $\gamma = 0$ would be the best choice to achieve this. However, experience has shown that this choice can lead to a suboptimal triangle subdivisions in some special cases.

Mesh Decimation

Simplification of polygonal surfaces is a commonly used technique in computer graphics. Typically, an initial model has been constructed using a large amount of triangles. The challenge is then to reduce the number of triangles without introducing large visual or computational errors to the simplified model. A large number of techniques have been proposed [7, 13] for performing this task. The complexity of the algorithms differs, and they also result in approximations that have varying accuracy.

Garland and Heckbert introduced a technique for surface simplification in [14]. Their approach based on the quadrics error metrics has been show to perform quite well on objects without open boundaries [7], which should make it suitable for use on gamut surfaces. In addition, the algorithm is fast enough [15] to make it suitable for interactive construction of gamuts.

Method

We propose that a mesh decimation algorithm can be used to construct simplified gamut surfaces that maintain a higher accuracy than comparable algorithms. While the USV algorithm uses

a pre-processing step based on segmentation of the color space to reduce the number of input points to the surface construction algorithm, our new method utilizes all input points to construct the initial surface. Then, a mesh decimation algorithm is used to reduce the complexity of the generated surface.

Generally, the recommended procedure for generating a gamut boundary based on a device model and a point-based method for gamut boundary determination is to generate a high number of simulated data points using the model to impose a gamut constraint on the point position. We start by using the modified convex hull algorithm to construct a very detailed surface that consists of many very small triangles. We ensure that the data structure contains neighbor information in order to be able to simplify the surface.

For our experiment, we perform a preprocessing step to make the surface follow gamut concavities as described by Balasubramanian and Dalal [5], using $\gamma = 0.2$ as recommended by [4]. Figure 1 shows one of the initial gamuts used in our experiment. The very detailed gamut follows the inherent surface of the device data closely, but the large amount of triangles make it less suitable for both visualization and further computations.

Although we chose the modified convex hull algorithm to generate the detailed gamut surface, we could have chosen any algorithm that produces a surface of triangles. The mesh simplification algorithm can be used to simplify any triangulated surface as long as the data structure contains connectivity information and does not duplicate vertices. However, it is clearly an advantage to have an accurate gamut surface as the starting point for the simplification process, and the modified convex hull has been shown [4] to give the best approximation of the gamut.

The mesh simplification algorithm proposed by Garland and Heckbert [14] based on quadrics error metrics is then used to reduce the number of triangles in the surface. By collapsing an edge to a point and removing the two triangles that shared the edge, the surface is iteratively simplified until the desired level of detail is reached. This step is illustrated in Figure 6. The contraction $(\mathbf{v}_1, \mathbf{v}_2) \rightarrow \overline{\mathbf{v}}$ is done by replacing all references to the vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ with $\overline{\mathbf{v}}$, where the position of the new vertex $\overline{\mathbf{v}}$ is calculated as described later in this section. The list of triangles is also changed by discarding the two shaded triangles and updating the neighbor information.

The edge that should be contracted must be chosen for each iteration of the algorithm. This is done efficiently by maintaining a heap structure, using a calculated cost for each edge as the key of the heap. The cost represents the error introduced by contracting the edge. By using a heap one can then always perform the edge contraction that the cost function reports should introduce the least amount of error.

The quality of the algorithm depends on the choice of cost function, which in this case is a quadric error metric. The algorithm of Garland and Heckbert is based on the metric introduced by Ronfard and Rossignac [16], that can be understood to maintain a set of planes for each vertex on the surface. The error contribution of a vertex that has been moved to a new position is defined to be the sum of the squared distances from the vertex to its associated planes.

Initially, each vertex conceptually contains a list of the planes of all the triangles that are incident to the vertex. The error of the vertex starts out as being equal to 0, since the vertex is contained in all of its planes. The list of planes is then updated as edges are contracted, each contraction resulting in a vertex that has a list of planes equal to the union of the two sets of planes belonging to $\mathbf{v}_1$ and $\mathbf{v}_2$.

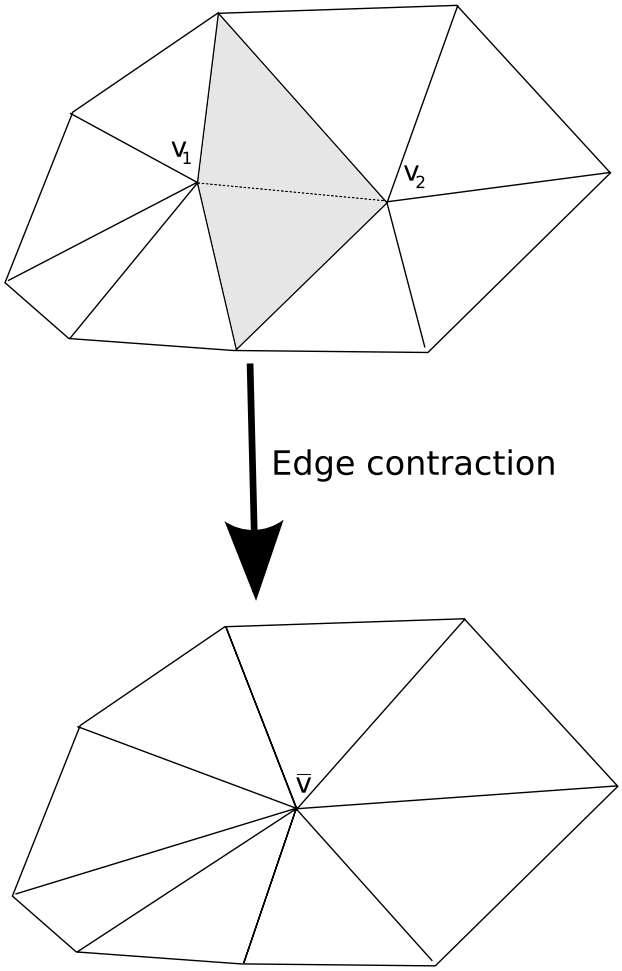


Figure 6. *The polygonal simplification algorithn iteratively performs a basic operation known as edge contraction. The edge connecting the two vertices (v_1, v_2) is replaced with a single point. The process eliminates the two shaded triangles and the connectivity of the neighboring triangles is updated*

The calculation of these sets can be done implicitly by representing the error as a quadric Q. Garland and Heckbert start by defining the plane equation of a face, $\mathbf{n}^T\mathbf{v}_0 + d = 0$, where $\mathbf{n}$ is the unit normal of the plane, $\mathbf{v}_0$ is any point in the plane, and d is a constant. The squared distance D^2 from this plane to the vertex $\mathbf{v}$ can then be rewritten as

$$\begin{aligned} D^2 &= (\mathbf{n}^T\mathbf{v} + d)^2 \\ &= (\mathbf{v}^T\mathbf{n} + d)(\mathbf{n}^T\mathbf{v} + d) \\ &= \mathbf{v}^T(\mathbf{n}\mathbf{n}^T)\mathbf{v} + 2d\mathbf{n}^T\mathbf{v} + d^2 \end{aligned} \quad (1)$$

D^2 can then be represented as the quadric Q

$$Q = (\mathbf{A}, \mathbf{b}, c) = (\mathbf{n}\mathbf{n}^T, d\mathbf{n}, d^2) \quad (2)$$

$$Q(\mathbf{v}) = \mathbf{v}^T\mathbf{A}\mathbf{v} + 2\mathbf{b}^T\mathbf{v} + c \quad (3)$$

The sum of errors for a set of planes as well as the union caused by edge contraction can be found by simple addition of two quadrics, defined as a componentwise operation. By using addition to represent the union of two sets of planes, some inaccuarcy is introduced due to several planes possibly being part of both sets. In addition, we must check the surface at each step to make sure that it does not fold over on itself [14].

We find the edge contraction which introduces the least amount of error to the surface by using the addition of quadrics. However, we still need to determine the position of the new vertex. Garland and Heckbert find the optimal placement of the vertex by minimizing the quadric of the vertex $\overline{\mathbf{v}}$. They define the new position as $\overline{\mathbf{v}} = -\mathbf{A}^{-1}\mathbf{b}$ if the minimization of the error results in a single point. If A is not invertible one of the end points of the edge is chosen as the new vertex position.

When the number of triangles in the structure has reached the desired level of detail, the algorithm stops the simplification process.

The ability to quickly and efficiently change the level of detail is especially useful when used for constructing gamuts in interactive tools, e.g., [17]. The simplification algorithm that we have evaluated is comparatively fast [7] while producing accurate surface approximations. We also propose that the algorithm can be used interactively using, e.g., a slider to change the number of triangles in the surface. By extending the original simplification algorithm to maintain a list of the changes that are done while simplifying the surface, the detail can easily be increased again after the simplification has been executed.

For undoing the simplifaction that has been applied to the surface, some information about the changes made at each iteration of the algorithm must be stored. We record the changes made to the vertices and the connectivity of the triangles, and instead of deleting redundant vertices and triangles we simply mark them as no longer being needed. When the user requests an increased level of detail, the simplification can then be reverted back to a larger number of triangles by performing the inverse operations in this list.

Results and Discussion

We have utilized the method previously described in [4] to compare this gamut boundary determination method against the USV method. The USV method has previously been shown [6] to perform better than the commonly used segment maxima algorithm, resulting in better accuracy for a given level of detail as determined by the number of segments used. We construct a reference gamut boundary by first using a device model for five different devices to generate surface points, and then utilize the modified convex hull algorithm to generate the boundaries.

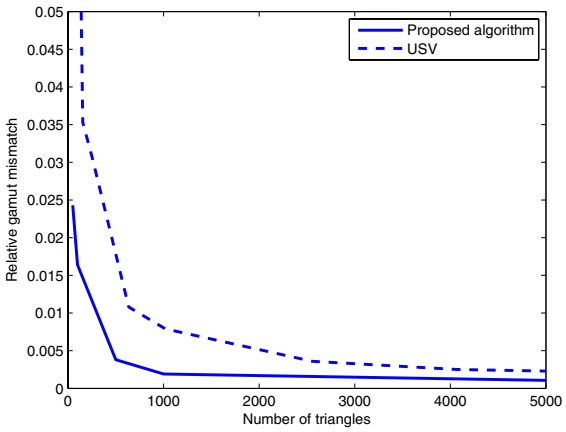


Figure 8. *The average relative gamut mismatch of our proposed method compared to the USV algorithm. The results show that our new algorithm performs better than USV for an equal number of triangles, and a high degree of accuracy can be achieved using a low number of triangles.*

The reference gamut boundaries consist of many small triangles of densely sampled surface points. We then use USV and our proposed algorithm to construct gamut boundaries that are made up of a much smaller number of triangles.

We compare these gamut boundaries to the reference and compute a relative volume mismatch using a voxel technique [4]. The average mismatch from these devices is plotted in Figure 8, where the average relative gamut mismatch of surfaces constructed using different numbers of triangles are shown. Our new algorithm performs better than USV for all tested choices of detail. The difference is particularly large for a small number of triangles, and our new algorithm requires very few triangles to give a highly accurate representation of the gamut boundary. As the number of triangles increases, the surfaces of both algorithms approaches that of our reference gamut.

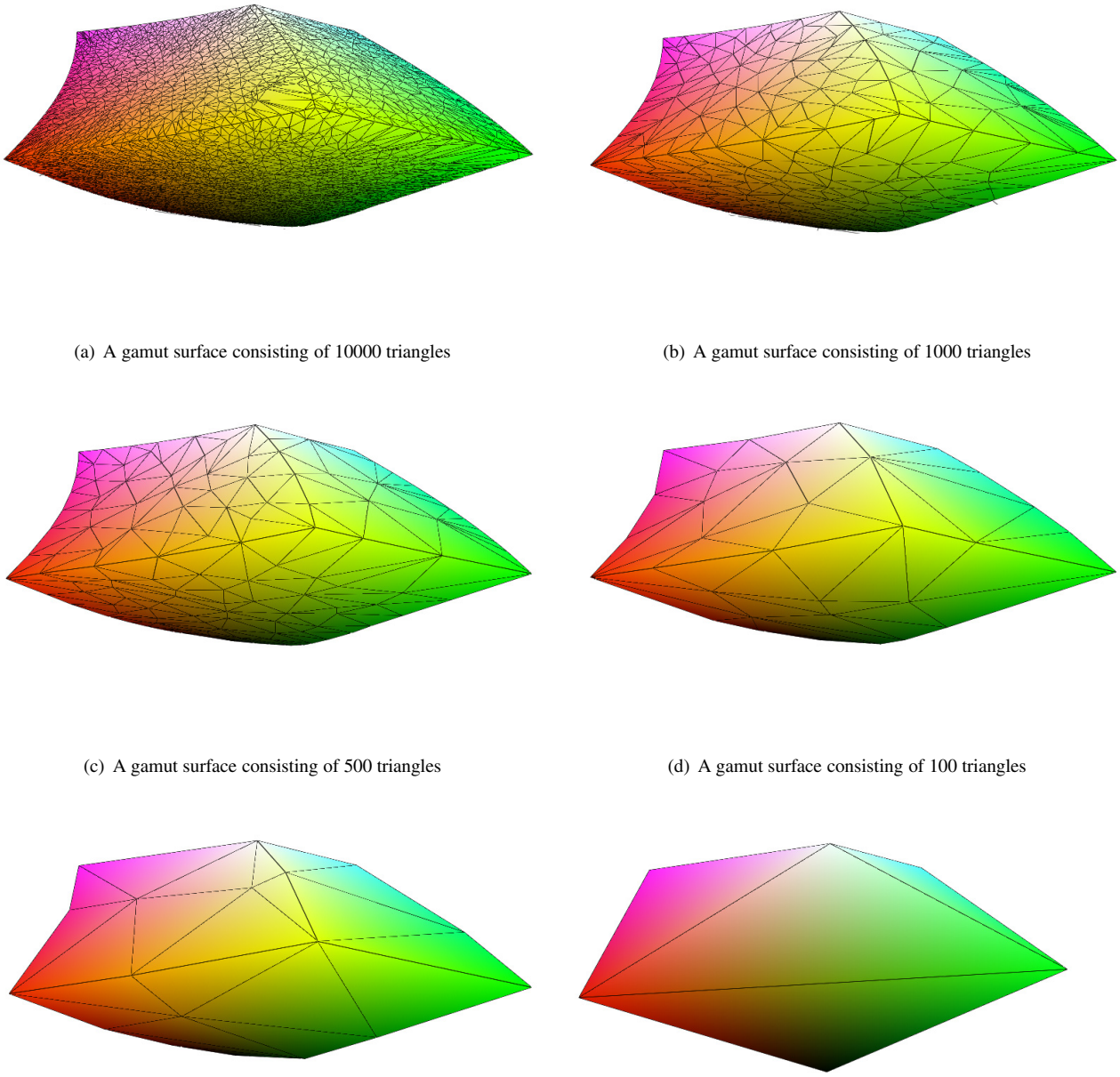
We can see 7 that the simplification maintains the sharp edges of the gamut quite well even when the number of triangles is quite low. The algorithm removes the almost coplanar triangles along the sides of the gamut, joining the small triangles and generating larger surface parts. The obvious alternative to simplifying a detailed surface is to generate a smaller number of points that are used as input to the surface construction algorithm. However, we can see that this would result in a less optimal surface.

There is no straightforward method that can be used to decide the optimal distribution of such points, leaving a uniform spacing of the points in the device color space as the logical starting point. This results in a gamut surface where the sides of the gamut consist of approximately the same number of points and triangles. This is clearly suboptimal, since sides where the points are nearly coplanar can be described using less triangles than sides that are more convex or concave.

The mesh simplification produces a more optimal selection of triangles and point distribution, since the cost function ensures that it generally will combine the triangles that are more coplanar before the others. The surfaces that it generates are therefore more accurate for a given number of triangles.

Conclusions

We have presented a new method for generating accurate gamut boundaries with a small number of surface triangles. Our method has been tested on data from five different devices, and



(a) A gamut surface consisting of 10000 triangles

(b) A gamut surface consisting of 1000 triangles

(c) A gamut surface consisting of 500 triangles

(d) A gamut surface consisting of 100 triangles

(e) A gamut surface consisting of 50 triangles

(f) A gamut surface consisting of 10 triangles

Figure 7. *An example of a gamut surface which is constructed using our algorithm. The initial surface in Figure 1, which is constructed using modified convex hull, consists of a high number of triangles. The structure is simplified by collapsing edges to reduce the number of triangles.*

results show that it performs better than an existing state-of-the-art algorithm in terms of accuracy for a given number of triangles. Our new algorithm produces visually pleasing gamut surfaces, and the edge contraction of the mesh decimation is performed in a sequence that can be seen to maintain the edges of the gamut.

In addition to increased accuracy, the proposed algorithm has the advantage that it can produce surfaces using an arbitrary number of triangles. For every iteration, the number of triangles is reduced by two. Given a desired number of triangles for the boundary, the number of triangles generated will therefore deviate from this by at most 1 triangle. By comparison, the USV algorithm has a limited selection of segments due to the technique used for subdivision of the triangles.

References

[1] Ján Morovič and M. Ronnier Luo. The fundamentals of gamut mapping: A survey. *Journal of Imaging Science and Technology*, 45(3):283–290, 2001.

[2] Ron Kimmel, Doron Shaked, Michael Elad, and Irwin Sobel. Space-dependent color gamut mapping: a variational approach. *IEEE Transactions on Image Processing*, 14(6):796–803, 2005.

[3] Ivar Farup, Carlo Gatta, and Alessandro Rizzi. A multiscale framework for spatial gamut mapping. *IEEE Transactions on Image Processing*, 16(10):2423–2435, 2007.

[4] Arne M. Bakke, Jon Y. Hardeberg, and Ivar Farup. Evaluation of gamut boundary descriptors. In *Proceedings of IS&T and SID's Fourteenth Color Imaging Conference*, pages 50–55, Scottsdale, Arizona, 2006.

[5] Raja Balasubramanian and Edul Dalal. A method for quantifying the color gamut of an output device. In *Color Imaging: Device-Independent Color, Color Hard Copy, and Graphic Arts II*, volume 3018 of *Proc. SPIE*, San Jose, CA, January 1997.

[6] A.M. Bakke, I. Farup, and J.Y. Hardeberg. Improved gamut boundary determination for color gamut mapping. In *IARIGAI*, Valencia, Spain, Sep 2008.

[7] P. Cignoni, C. Montani, and R. Scopigno R. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22:37–54(18), 25 February 1998.

[8] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The Quickhull Algorithm for Convex Hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[9] Gary W. Meyer, Linda S. Peting, and Ferenc Rakoczi. A color gamut visualization tool. In *Proceedings of IS&T and SID's Color Imaging Conference: Transforms and Transportability of Color*, pages 197–201, Scottsdale, Arizona, November 1993.

[10] Gary W. Meyer and Chad A. Robertson. A data flow approach to color gamut visualization. In *Proceedings of IS&T and SID's 5th Color Imaging Conference: Color Science, Systems and Applications*, pages 209–214, Scottsdale, Arizona, 1997.

[11] Ján Morovič and M. R. Luo. Gamut mapping algorithms based on psychophysical experiment. In *Proceedings of IS&T and SID's 5th Color Imaging Conference: Color Science, Systems and Applications*, pages 44–49, Scottsdale, Arizona, 1997.

[12] John R. Baumgardner and Paul O. Frederickson. Icosahedral discretization of the two-sphere. *SIAM Journal on Numerical Analysis*, 22(6):1107–1115, 1985.

[13] David P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001.

[14] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

[15] Muhammad Hussain, Yoshihiro Okada, and Koichi Niijima. Efficient and feature-preserving triangular mesh decimation. In *Journal of WSCG*, pages 167–174, 2004.

[16] Rémi Ronfard and Jarek Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum*, 15(3):67–76, 1996.

[17] Ivar Farup, Jon Y. Hardeberg, Arne M. Bakke, Ståle Kopperud, and Anders Rindal. Visualization and interactive manipulation of color gamuts. In *Proceedings of IS&T and SID's 10th Color Imaging Conference: Color Science and Engineering: Systems, Technologies, Applications*, pages 250–255, Scottsdale, Arizona, 2002.