Eurographics Symposium on Geometry Processing(2003)
L. Kobbelt, P. Schröder, H. Hoppe (Editors)

Multi-Chart Geometry Images

P. V. Sander[1], Z. J. Wood[2], S. J. Gortler[1], J. Snyder[3], and H. Hoppe[3]

[1] Harvard University, [2] Caltech, [3] Microsoft Research

Abstract

We introduce multi-chart geometry images, a new representation for arbitrary surfaces. It is created by resampling a surface onto a regular 2D grid. Whereas the original scheme of Gu et al. maps the entire surface onto a single square, we use an atlas construction to map the surface piecewise onto charts of arbitrary shape. We demonstrate that this added flexibility reduces parametrization distortion and thus provides greater geometric fidelity, particularly for shapes with long extremities, high genus, or disconnected components. Traditional atlas constructions suffer from discontinuous reconstruction across chart boundaries, which in our context create unacceptable surface cracks. Our solution is a novel zippering algorithm that creates a watertight surface. In addition, we present a new atlas chartification scheme based on clustering optimization.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Surface representations.

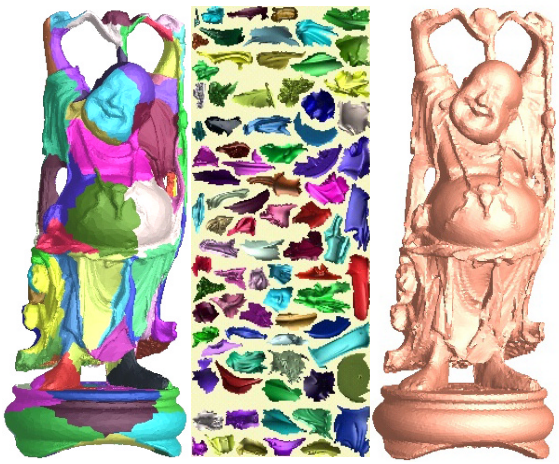
1. Introduction

Regular remeshing is the process whereby an irregular mesh is approximated by a mesh with (semi-)regular connectivity [Eck et al. 1995]. The simplicity of a regularly remeshed representation has many benefits. In particular it eliminates the indirection and storage of vertex indices and texture coordinates. This will allow graphics hardware to perform rendering more efficiently, by removing random memory accesses and thus improving memory access performance.

Geometry images. The most extreme such method, which creates the most regular remeshed representation, is the *geometry image* (GIM) introduced by Gu et al. [2002]. Their construction converts the surface into a topological disk using a network of cuts and parametrizes the resulting disk onto a square domain. Using this parametrization, the surface geometry is resampled onto the pixels of an image. As an added benefit, techniques such as image compression can be directly applied to the remesh.

However, this extreme approach of mapping an entire surface to a single square has limitations. Models having disconnected components require a separate geometry image per component, and complicated shapes with many extremities or topological handles have distorted parametrizations.

Semi-regular remeshing. A less extreme approach is to create a remesh with the connectivity of a subdivided base mesh. Examples of this approach include the methods of Eck et al. [1995], Lee et al. [1998; 2000], Kobbelt et al. [1999], Guskov et al. [2000], and Wood et al. [2000]. For these representations, special kinds of continuous multiresolution basis functions can be derived that allow multiresolution editing [Zorin et al. 1997] and compression [Khodakovsky et al. 2000].



chartified orig. mesh | image 400x160 | remesh PSNR=71.7 dB

Figure 1: *Example of a multi-chart geometry image.*

This more flexible representation still has constraints that can negatively impact parametrization efficiency. In particular, each chart (each surface region associated with a triangle of the base mesh) is effectively parametrized onto an equilateral triangle, which is then evenly sampled. Charts with non-triangular shapes are thus distorted by the parametrization; charts that are long and skinny are invariably sampled anisotropically. In addition, all charts, regardless of their size or

© The Eurographics Association 2003.

information content, must be allotted the same number of samples.[1]

Multi-chart geometry images. We describe a new atlas-based parametrization to define *multi-chart geometry images* (MCGIMs). Motivated by the atlas approach for texture mapping (e.g. Maillot et al. [1993]), we partition the surface into a geometrically natural set of charts, parametrize the charts individually onto irregular polygons, and pack the polygons into a geometry image (Figure 1). Such an atlas parameterization reduces distortion because the smaller charts are easier to flatten and because their parametrized boundaries can assume more natural shapes. Low-distortion parametrizations distribute samples more uniformly over the surface and therefore better capture surface signals. Each chart can be allotted an appropriate number of "defined" samples in the geometry image, separated by "undefined" samples. Our representation can be viewed as *piecewise regular* since it is composed of sub-regions of a regular GIM.

MCGIMs retain the key advantage of the original GIMs – rendering involves a simple scan of the samples in stored order. And, MCGIMs overcome the distortion present in GIMs, at the small expense of assigning some samples special "undefined" values.

A serious drawback of a general atlas parametrization is that its reconstructed signal is discontinuous across chart boundaries. Because irregular chart outlines do not align with the sampling grid, boundaries between neighboring charts are generally sampled differently. For geometry images, such signal discontinuities create unacceptable cracks in the surface. Even a few erroneous pixels become glaring artifacts. To prevent cracks, we present a novel zippering scheme that unifies boundaries of the discretized MCGIM charts to create a continuous (watertight) model.

Our main contribution is the MCGIM representation obtained through this zippering scheme. To create accurate MCGIMs, we also introduce several improvements to existing atlas parametrization methods. We develop a new atlas chartification scheme, based on general clustering optimization inspired by the work of Lloyd [1957] and Max [1960]. Our scheme creates compact, roughly flat charts whose boundaries align with the model's creases. We apply the mesh optimization framework of Hoppe et al. [1993] to refit the MCGIM samples to the original surface, thereby improving its accuracy considerably. And, we extend the "Tetris" packing algorithm of Lévy et al. [2002] to optimize both chart rotations and overall domain dimensions. With these improvements, we demonstrate that MCGIMs outperform both single-chart geometry images and semi-regular remeshes on example models.

[1] This assumes that one wants a remesh free of T-vertices, while avoiding the indirection-based representations that would result from adaptive subdivision refinement.

2. Previous work

Atlas parametrizations. Some atlas schemes map individual triangles or pairs of triangles separately into a texture [e.g. Cignoni et al. 1998, Carr and Hart 2002]. In this section, we review more general chart-based atlas constructions.

Maillot et al. [1993] partition a mesh into charts based on bucketing of face normals. Their parametrization method optimizes edge springs of non-zero rest length.

Piponi and Borshukov [2000] manually cut a subdivision surface using a network of edges. They parametrize the resulting single chart using a "pelting" analogy, by stretching out the surface boundary using a collection of springs.

Sander et al. [2001] partition a mesh using greedy face clustering (also done independently by Garland et al. [2001]). They parametrize the resulting charts onto convex polygons using geometric stretch. The charts are packed into a square using a greedy algorithm based on bounding rectangles.

Lévy et al. [2002] align chart boundaries with high-curvature features of the surface. After locating a set of seed faces farthest from sharp features, they grow charts about these seeds, and merge some resulting charts. They parametrize each chart using least-squares conformal maps with free boundaries. They use a Tetris-like packing algorithm that searches for best fit over the horizon of pieces packed so far.

Sorkine et al. [2002] grow charts while simultaneously parametrizing them. The chart growth stops when a distortion bound is reached or if self-overlap is detected, and a new chart is started. Their parametrization uses a stretch-based metric that penalizes both undersampling and oversampling.

Sheffer and Hart [2002] cut a surface into a single chart by cutting through high-distortion, less-visible surface regions.

None of these atlas constructions address the problem of inter-chart cracks when resampling geometry, as our zippering scheme does. Our chartification method is less greedy than previous methods, thus yielding better results.

Model decomposition. Shlafman et al. [2002] also develop a clustering-based approach to mesh partitioning. Because their application is morphing and not parametrization, their clustering distance metric does not account for overall chart planarity.

Zippering. Turk and Levoy [1994] reconstruct watertight surfaces from scanned height meshes by zippering mesh boundaries together. Because the desired surface is unknown, zippering is a challenging operation. In contrast, our zippering task (Section 4.5) is given a reference surface as input, and so can be made simple and robust.

Several algorithms consider the problem of reconstructing surfaces from contours [e.g. Fuchs et al. 1977]. The simplest case they consider is that of building a surface ribbon that spans two parallel contour polygons. This is quite similar to our zippering algorithm. The difference is that we seek to unify the vertices across the gap rather than to construct a ribbon of new triangles between them.

© The Eurographics Association 2003.

3. MCGIM representation

A multi-chart geometry image (MCGIM) is a rectangular 2D grid whose entries store 3D points sampled from a given surface. Since the MCGIM consists of multiple, arbitrarily shaped charts packed together into a grid, there are some wasted samples that lie outside the charts. We refer to these as *undefined samples*. The remaining are called *defined samples*, and can be further distinguished into *boundary samples* (with at least one undefined sample among its four immediate neighbors) and *interior samples*. The defined region of the image is specified either using a *bitmask* or by storing *NaN* (Not-a-Number) into the coordinate values of undefined samples.

In order to reconstruct a surface from the MCGIM, we must specify how polygon faces are formed. For each 2-by-2 quad of samples, we examine how many samples are defined. If fewer than three are defined, no polygon is formed. If exactly three samples are defined, one triangle is created. And if all four samples are defined, two triangles are created, with the diagonal chosen to be the shorter one in 3D. Applying this method to all quads in the image creates a triangle mesh. Although some triangles are in fact degenerate due to our zippering, such degenerate triangles can still be fed to the graphics pipeline since they have no effect on rendering.

Our goal is MCGIM construction is to create a *watertight* mesh, i.e. a 2-dimensional manifold where every edge is adjacent to exactly two faces and every vertex is adjacent to exactly one ring of faces. Note that MCGIMs can represent meshes with several connected components, as well as meshes with boundaries (where some edges are adjacent to only one face).

For rendering, normals are computed at each sample point using the normalized cross product of central differences. When an adjacent sample is undefined, it is replaced by the current sample in the central difference. (Opposite neighbors cannot both be undefined if the mesh is manifold.)

4. MCGIM creation

4.1 Mesh chartification

The first step in producing a multi-chart geometry image is to partition the mesh into charts. Charts that have compact boundaries and are mostly flat yield better parametrization and packing results, thereby reducing the error of the geometry image.

We first tried the chart merging approach of Sander et al. [2001]. They apply a greedy face clustering algorithm. Because the resulting charts have irregular boundaries, they subsequently straighten the boundaries. While the method works well when the desired number of charts is high, for a smaller number of charts the boundaries do not usually follow the "creases" of the model. For instance, after the straightening step, the gargoyle wings of Figure 2a are each composed of a single chart.

This same observation motivated Lévy et al. [2002] to develop an algorithm that aligns chart boundaries with high-curvature features. They first estimate sharp features using local edge dihedral angles, then identify seed faces farthest from sharp features, and finally grow charts about these seeds. In a sense, our new method can be seen as a generalization of this approach, but uses a more general optimization and does not require feature detection.

Algorithm overview. Our algorithm is inspired by Lloyd-Max quantization, which partition sets of elements (e.g. colors), into a fixed number of clusters (e.g. quantized colors). The basic strategy is to iteratively optimize the clusters using two steps:

- Partition the elements according to current cluster models.
- Refit a model to each cluster.

In our context, elements are mesh faces, clusters are charts, and the model is a seed face for each chart. We call the first step "chart growth" and the second step "seed computation". We now describe these in detail. Later, we present bootstrapping and termination conditions.

Phase 1: Chart growth. *Given n seeds representing the n charts, assign each face to a chart.*

In this phase we partition the elements by growing all the charts simultaneously using a Dijkstra search algorithm on the dual graph of the mesh. This graph connects each pair of adjacent mesh faces by an edge. We assign edge costs that encourage charts to be planar and to have compact boundaries. Whereas Shlafman et al. [2002] only consider local dihedral angles, our metric considers distance to a global chart normal. Specifically, the edge cost between a face F on a chart C and its adjacent face F' that is a candidate to join C is a measure of geometric distance between the two faces, and difference in normal between F' and the chart normal N_C:

$$cost(F, F') = (\lambda - (N_C \cdot N_{F'}))(|P_{F'} - P_F|) ,$$

where N_C is the normal of the chart containing F (the average normal of all faces already in the cluster), and $P_{F'}$ and P_F are the centroid points of F' and F. The parameter λ regulates the relative weight between the normal and geometric distance terms. For all of our experiments, we set λ to 1. Phase 1 terminates when all faces have been assigned to charts.

Phase 2: Seed computation. *Given a chartified mesh, update the seed for each chart.*

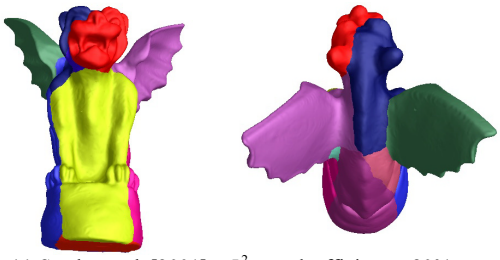
In this phase, we update the seed of each chart to be the "most interior" face within the chart. To find this face, we again perform a Dijkstra search on the dual graph of the mesh, but in the opposite direction. We start the search from all faces adjacent to a chart boundary. The edge cost is simply the geodesic distance

$$cost(F, F') = |P_{F'} - P_F| .$$

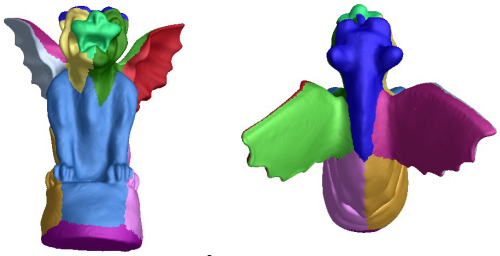
The last face reached within each chart becomes its new seed.

Convergence. Phases 1 and 2 are repeated until the new seeds are identical to those of the previous iteration. Occasionally, this discrete optimization fails to converge because the seeds begin to cycle between nearby faces (defining nearly identical chartifications). We detect these cycles by checking the set of seeds against the ones obtained in previous iterations. In practice, if a cycle is detected, any chartification in the cycle is acceptable.

© The Eurographics Association 2003.



(a) Sander et al. [2001]; L^2 stretch efficiency: 80%



(b) our method; L^2 stretch efficiency: 99%

Figure 2: *Comparison of the chartification algorithm from Sander et al. [2001] with our new approach on a 12-chart gargoyle. (Garland et al. [2001] would produce results comparable to (a).)*

Bootstrapping. In order to bootstrap the chartification process, we first assign a random face to be the first seed. Then we run successive iterations of phases 1 and 2, with one small modification: at the end of phase 1, if the desired number of seeds has not been reached, we add a new seed. The new seed is set to be the last face that was assigned to a chart during phase 1. This tends to place the new seed far away from the other seeds. Once the desired number of seeds has been reached, we repeat phases 1 and 2 until convergence without adding new seeds.

Chart topology constraint. To ensure that all charts are topological disks, we disallow the formation of annuli and boundaryless (closed) charts. This is a simple local check on the chart boundary performed during chart growth. There are situations where a face cannot be assigned to any chart due to the above constraint. When that happens, we add as a new seed the first face that failed this constraint during the chart growth. For instance, one cannot chartify a sphere with one chart. So even if one chart is requested, more seeds are inserted automatically.

Result example. As shown in Figure 2b, with our new method the front and back of the ears and wings are each assigned their own chart. Also note that the chartification is much more symmetric. We parametrized both models and calculated their stretch efficiencies, which is a measure of distortion of the parametrization. While the method of Sander et al. [2001] has a stretch efficiency of just 80% (due to high stretch on the gargoyle wings), the new method has the near optimal stretch efficiency of 99%. More chartification examples are shown throughout the paper.

4.2 Chart parametrization

To parametrize the charts, we minimize the L^2 geometric stretch metric of Sander et al. [2001]. This same metric is used by Gu et al. [2002]. It penalizes undersampling, and results in samples that are uniformly distributed over the surface. Sander et al. [2002] shows that the stretch metric is a predictor of the reconstruction error under the assumption of locally constant reconstruction. Instead of fixing the boundary to a square, we allow it to be free during the optimization [Sander et al. 2002].

We employ the hierarchical parametrization algorithm from Sander et al. [2002] to solve the minimization problem. This hierarchical algorithm finds better minima, and is orders of magnitude faster than uniresolution methods. Once all the charts are parametrized, we scale them relative to each other based on their stretch [Sander et al. 2002].

4.3 Chart packing

The algorithm in the previous section produces a set of charts parametrized on a continuous domain. The next step is to discretize the charts and pack them into an image.

Packing a set of 2D shapes is a provably hard problem [Milenkovic 1998], thus motivating heuristic approaches. Our approach is a simple extension of the "Tetris" packing algorithm by Lévy et al. [2002]. Their scheme iteratively adds charts to a square image while keeping track of a horizon that bounds all the charts packed so far. For each chart to insert, they locate the horizontal position that minimizes the wasted vertical space between the current horizon and the new chart.

Our method considers 16 orientations for each chart. Also, we let the domain be an arbitrary rectangle, rather than a fixed-sized square, since current graphics hardware can support textures with arbitrary dimensions. The user specifies a desired upper-bound on the total number of MCGIM samples, and we optimize for the best rectangle within this area bound. Figure 3 demonstrates the improvement.

Chart scales. A scaling factor determines how the continuous charts map to the discrete image grid. We initially set this scaling factor assuming 100% packing efficiency, and then successively reduce it by 1% until a successful packing is achieved. For each scaling factor, the charts are rasterized into a buffer to obtain their discrete shapes (see Section 4.4).

Domain dimensions. For a given chart scale, two nested loops consider MCGIM widths and heights whose product is less than the specified upper-bound. We limit the aspect ratio to be less than 4.

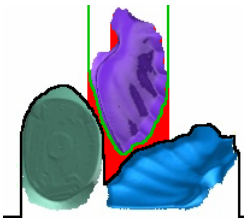
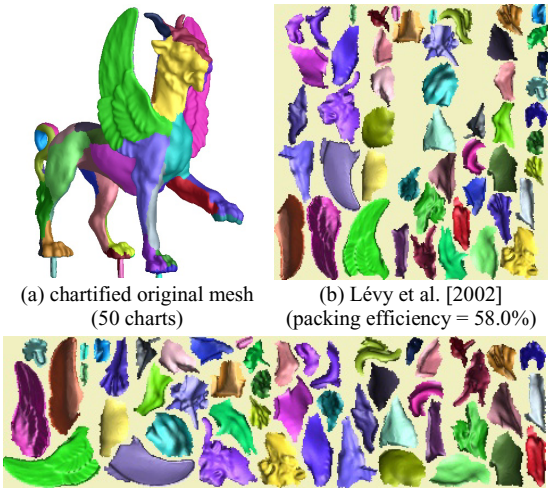


Chart placement. We place each chart in the position and orientation that minimizes "wasted space" (inset in red). In Lévy et al. [2002], the empty space is the region between the discretized upper horizon of the atlas (black) and the discretized lower horizon of the inserted chart (green). We also add any

© The Eurographics Association 2003.



(a) chartified original mesh (50 charts)

(b) Lévy et al. [2002] (packing efficiency = 58.0%)

(c) our method (packing efficiency = 75.6%)

Figure 3: *Our improvement on the packing of Lévy et al. [2002].*

unused space within the intersection of the upper and lower envelope of the new chart (red region in the upper-left), since a different rotation could avoid that wasted space.

4.4 Geometry image sampling

Given desired scale and orientations of the charts within the geometry image, we discretize the surface charts into the image. The image pixels will receive (x,y,z) positions sampled from the surface (Figure 4).

The set of defined image samples implicitly create a triangulation, using the rules described earlier in Section 3. We require that this triangulation maintain the same topology as the original surface. Satisfying this requirement involves two sub-problems:

(1) Creating discretized charts that are topological disks, since the surface charts are known to be disks.

(2) Connecting the charts together without cracks. The explicit chart boundaries on the original mesh will help in this zippering process, as explained in the next section.

We address the first sub-problem as follows. (The second problem is the subject of the next section.)

Interior rasterization. We begin by rasterizing each chart's 3D triangles into the 2D image domain using ordinary scan conversion. During rasterization, the value stored in the image domain sample is the corresponding (x,y,z) location from the 3D triangle. The unwritten samples remain undefined. This simple procedure does not guarantee that the rasterized charts will have disk topology. For example, undersampling a narrow chart "peninsula" leads to broken charts (see Figure 5a).

Boundary rasterization. To prevent charts from breaking into disconnected components, we rasterize the chart's boundary edges to add connecting samples (Figure 5b). In particular, for each boundary edge, we construct a 1-pixel wide rectangle centered along the boundary edge, and scan-convert it into the texture domain. During this boundary rasterization, undefined samples covered by the boundary-polygon are labeled as defined and filled in with the closest geometric point on the boundary. Previously defined samples are not overwritten.

Non-manifold dilation. Even with boundary rasterization, a discrete chart can have a non-manifold reconstruction at boundary samples (Figure 6). For instance, a single line peninsula of defined samples produces no triangles. Reconstructions can also erroneously produce two triangles touching at a single vertex. To correct these non-manifold neighborhoods, we extend (dilate) the chart to include additional samples. For each non-manifold sample p, we make it manifold by making "defined" a minimal number of its undefined neighbors. We assign these new samples the same position as p. We iterate until all samples are manifold.

The are cases when our algorithm can fail, particularly when the requested geometry image area is small. Some discretized charts could become annuli for coarse samplings. For practical image resolutions this is not a problem.

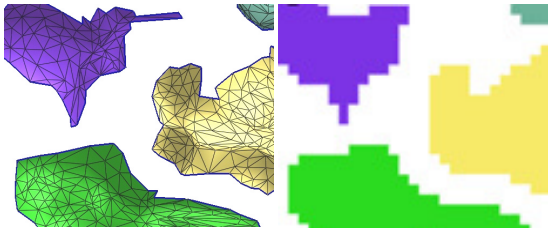
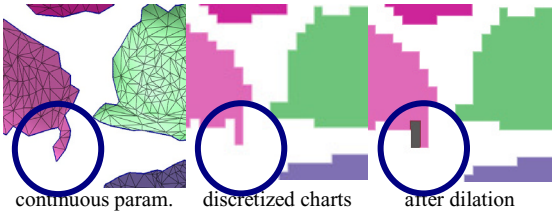


Figure 4: *Discretization of the bunny.*

Figure 5: *Rasterizing the boundary of the chart ensures that it is connected. (Samples in red are added by boundary rasterization.)*



continuous param. discretized charts after dilation

Figure 6: *The discretized charts may have non-manifold structures. We repair these by local chart dilation.*

© The Eurographics Association 2003.

4.5 Geometry image zippering

To prevent cracks in the reconstructed mesh, we "zipper" boundary samples along discrete boundaries of charts that adjoin on the 3D surface. The overall process is illustrated in Figure 7.

Cut-Nodes. The zippering algorithm first identifies all the *cut-nodes* in the mesh. A cut-node is a mesh vertex with more than two adjacent charts; in other words, a chart corner. For every chart adjacent to the chart corner, the cut-node must be mapped to a unique boundary *cut-node sample* in the image domain. We choose the closest boundary sample in terms of 3D distance. In addition, cut-node placement is constrained to preserve the clockwise-order of the cut-paths. In practice the GIM resolution is large enough to map each cut-node to a distinct cut-node sample. Cut-node samples corresponding to the same cut-node are all assigned its 3D geometric position.

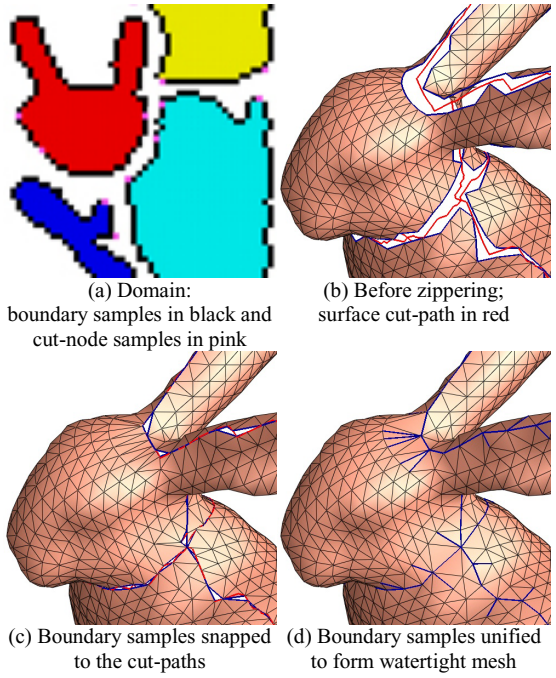
Cut-Paths. Cut-nodes form the end points of a set of edges comprising the *cut-path* or boundary between two adjacent charts. Every cut-path in the mesh maps to two *discrete cut-paths* (sets of boundary samples) in the image domain. We unify boundary samples along these pairs of discrete cut-paths. See Figure 7 for an example of the discrete cut-paths.

Snap. For each boundary sample in the discrete cut-path, we snap (overwrite) its position to the closest point along the continuous cut-path. This narrows the gap between the charts but the mesh is still not watertight because of sampling differences (Figure 7c).

Unify. To seal the boundaries completely, we then unify boundary samples. Two discrete cut-paths corresponding to the same continuous chart boundary generally contain different numbers of samples. So, simple 1-to-1 unification along the two cut-paths is insufficient: more than two samples must occasionally be unified.

A simple greedy algorithm produces high quality zipperings. It performs a one-pass, lock-step traversal of the two discrete cut-paths. Figure 8 shows two steps in this traversal. We begin the zipper at a cut-node sample (see Figure 8a), which has already been unified as described above. During the traversal, we can advance the zipper along either of the discrete cut-paths (see Figure 8b); in this case the newly visited sample is assigned the geometric position of the current unified cluster. Alternatively, we can advance the zipper along *both* of the discrete cut-paths (see Figure 8c); in this case we create a new cluster of samples that is assigned the geometric position of one of these samples. From these three alternatives, we select the zipper advancement that alters the updated sample by the least amount.

After unification, all samples along a cut-path are unified to at least one sample on the other cut-path. The mesh reconstructed from the discrete samples is watertight. See Figure 7 for a close-up image of the unified cut-paths in the bunny reconstruction. When two adjacent boundary samples in the same cut-path are unified together, the reconstruction forms a degenerate triangle, which is in fact just a line. These degenerate faces do not affect the manifold property of the mesh and can be removed during reconstruction or harmlessly rendered as is. Even with texture mapping, degenerate faces do not introduce artifacts, since the texture samples are unified just like the geometric ones.



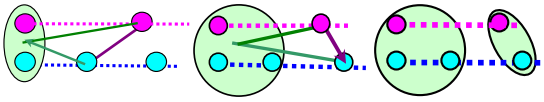
(a) Domain: boundary samples in black and cut-node samples in pink

(b) Before zippering; surface cut-path in red

(c) Boundary samples snapped to the cut-paths

(d) Boundary samples unified to form watertight mesh

Figure 7: *Illustration of the zippering process.*



(a) (b) (c)

Figure 8: *Three stages of the unification algorithm. Green bubbles represent unified clusters.*

4.6 Geometry image optimization

Having created a zippered, watertight MCGIM, we optimize its defined samples to improve geometric fidelity to the original mesh. As in feature-sensitive remeshing [Vorsatz et al. 2001], we seek to more accurately represent sharp features such as creases and corners. In addition, we want to smooth and regularize the sample positions across the zippered chart boundaries.

We extend the mesh optimization technique of Hoppe et al. [1993]. In that work, both the connectivity and vertex positions of the approximating mesh are optimized to fit the reference surface. The fit accuracy is measured by sampling a dense set of points from the reference surface and minimizing the squared distanced of those points from the approximating mesh.

© The Eurographics Association 2003.

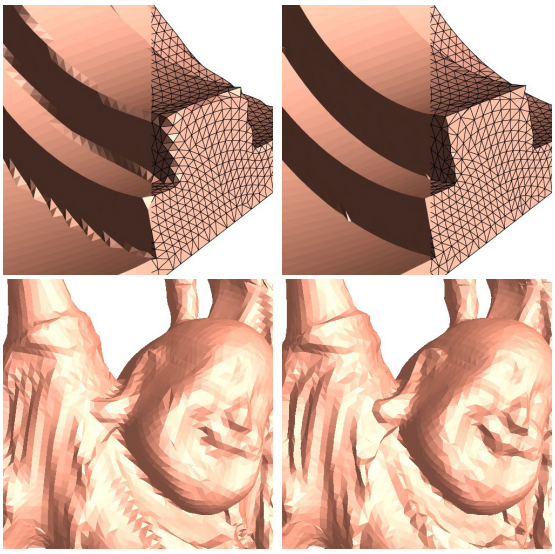


Figure 9: *Comparison of two multi-chart geometry images before and after mesh optimization.*

In our setting, the mesh connectivity is completely determined by the grid structure and the vertex positions. Recall from Section 3 that each quad of the grid is split into two triangle along its shorter diagonal. Thus, we only need to optimize vertex positions, noting that the grid diagonals may change as vertices are moved. We optimize using a non-linear conjugate-gradient solver, which converges to reasonably good answers even though the fit measure lacks continuity due to diagonal flipping.

To keep the mesh from folding over during the fit, we add a mesh smoothness term to the energy function being minimized. This term is set to the L^p norm of the dihedral angles of the mesh edges. Setting p=4 has produced good results. Figure 9 shows examples of MCGIM models with sharp features that are improved using geometry image optimization. The results are surprisingly good considering the restricted mesh connectivities.

5. Results

We have run our completely automatic pipeline on a large number of models. The process takes approximately 1-2 hours per model. Chartification takes no more than 5 minutes, parameterization no more than 15 minutes, sampling and zippering less than a minute, and mesh optimization about 10 minutes. The remainder of the time is needed for the discrete optimization used for packing. One could easily substitute other packing schemes – packing is not one of our main contributions.

We use two cost metrics to compare our results with other approaches. *Memory cost* is the total number of samples in the domain grid (i.e., domain rectangle area). *Rendering cost* is the number of samples that must be processed (i.e., defined samples). *Packing efficiency* is the ratio of defined samples to total number of samples. Thus,

*rendering cost = memory cost * packing efficiency .*

We express geometric accuracy as Peak Signal to Noise Ratio PSNR = 20 $\log_{10}(peak/d)$, where *peak* is the bounding box diagonal and *d* is the symmetric rms Hausdorff error (geometric distance) between the original mesh and the approximation. We similarly measure PSNR for a normal signal derived from the geometry (Section 3). A large normal error suggests parametrization artifacts like anisotropy.

Geometry image comparison. We created MCGIMs for a variety of models and compared them with GIMs from Gu et al. [2002]. To produce a fair comparison between single-chart and multi-chart geometry images, we also applied our geometry image optimization (Section 4.6) to the prior GIM results.

As shown in Table 1, we obtain significantly better PSNR results using MCGIMs for the models in Figure 14. For the same memory cost (i.e., including storage cost for our undefined samples), the geometry PSNR for our horse and feline meshes was more than 10dB higher than those of Gu et al. To make a comparison for equivalent rendering cost, we created new MCGIMs with the same number of defined vertices as GIMs. The additional samples yield a further improvement of about 2dB. For the Buddha model of Figure 1, the optimized GIM has a PSNR of 65.5dB, whereas our equivalent-memory-cost MCGIM has a PSNR of 71.7 dB.

models	gargoyle	horse	dragon	feline
genus	0	0	1	2
257x257 **GIM** from Gu et al. [2002]				
stretch efficiency	67.8%	32.4%	42.2%	33.3%
# defined vertices	66,049	66,049	66,049	66,049
# unique vertices	65,537	65,537	65,537	65,537
geometry PSNR	79.4	73.4	70.2	68.0
normal PSNR	30.3	19.4	20.1	17.6
257x257 **GIM** from Gu et al. + geometry image **optimization**				
geometry PSNR	83.5	71.9	73.6	68.1
normal PSNR	31.5	18.9	19.6	17.0
MCGIM (equivalent **memory** cost as Gu et al.)				
# charts	15	25	40	50
stretch efficiency	98.7%	99.2%	92.7%	99.1%
packing efficiency	72.7%	75.6%	73.1%	75.6%
dimensions	466x138	281x228	369x174	478x133
rectangle area	64,308	64,068	64,206	63,574
# defined vertices	46,724	48,389	46,913	48,038
# unique vertices	41,961	41,857	38,516	35,956
geometry PSNR	83.8	84.6	77.9	79.5
normal PSNR	33.7	30.8	22.9	28.7
MCGIM (equivalent **rendering** cost as Gu et al.)				
# unique vertices	58,769	55,329	52,059	55,329
geometry PSNR	85.2	85.3	79.4	82.5
normal PSNR	35.5	30.6	23.1	30.2

Table 1: *Geometry image comparison results.*

© The Eurographics Association 2003.

Texture-mapping using the implicit MCGIM parametrization is demonstrated in Figure 10.

Unlike GIMs, MCGIMs can also represent meshes with multiple components. The teapot in Figure 11 is an example with four components. Note the excellent chartification and packing.

Semi-regular remeshing comparison. We also made comparisons with semi-regular remeshes created by Guskov et al. [2000] and Lee et al. [1998] (Figure 12). We created MCGIMs with rendering cost (i.e., number of vertices) no larger than those of the semi-regular meshes. As shown in Table 2, our PSNR results are approximately 3db higher than those of the semi-regular meshes, representing a 30% reduction in Hausdorff error.

	semi-regular meshes		our MCGIM approach	
	# unique vertices	geom. PSNR	# unique vertices	geom. PSNR
feline	258,046	84.4	256,862	88.3
horse	112,642	87.8	99,851	90.2

Table 2: *Semi-regular remeshing comparison results.*

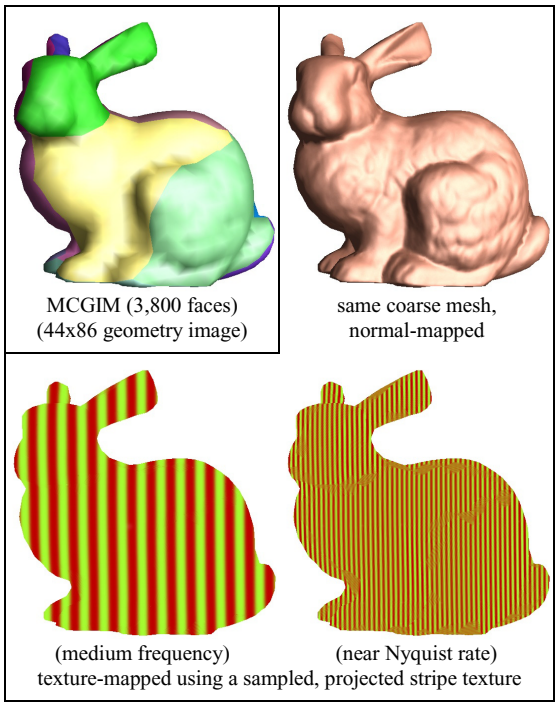


Figure 10: *An MCGIM texture-mapped with three 352x688 images. Texture coordinates are implicit, and each triangle is exactly associated with 9x9/2 texture samples. The resulting renderings are nearly seamless. The problem is that some zippered triangles are longer than interior ones, resulting in local undersampling.*

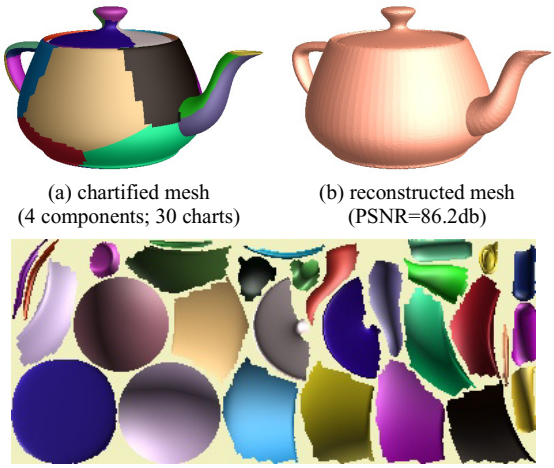


Figure 11: *As opposed to the GIM framework, MCGIMs can represent meshes with multiple components, such as this teapot.*

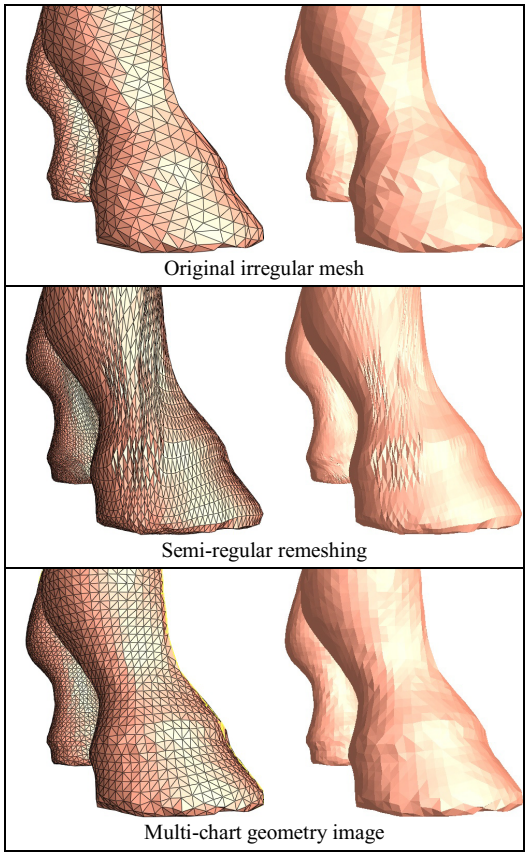


Figure 12: *Horse close-up showing improved sampling uniformity of MCGIM over semi-regular remeshing.*

© The Eurographics Association 2003.

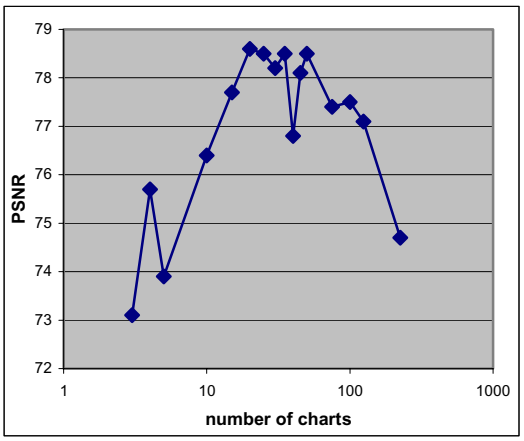


Figure 13: *MCGIM accuracy as a function of the number of charts on the horse model, for 128^2 samples.*

Selecting the number of charts. As shown by the graph in Figure 13, MCGIM accuracy exhibits a rough "hill" shape with increasing number of charts. Noise in the graph data results from optimization procedures involved in chartification, parameterization, zippering, and packing. The graph's shape comes from two countervailing trends: MCGIMs with fewer charts use less gutter space and unify fewer samples; MCGIMs with more charts produce less parametric distortion and more efficient packings. An optimal number of charts thus exists for a given model, but good accuracy is obtained over a broad range.

6. Summary and future work

We introduced multi-chart geometry images, a new surface representation that extends the geometry images of Gu et al. [2002]. MCGIMs have the same regular grid structure of GIMs, except that some of their image samples are undefined. This permits partitioning of the model into charts, and so obtains the flexibility needed to parametrize meshes of arbitrary complexity, high genus, and multiple components with less distortion.

We showed that MCGIMs significantly exceed the geometric accuracy of GIMs for equivalent memory cost. Obviously, this advantage increases if we neglect the cost of undefined samples, which are ignored in reconstruction and rendering. We also showed that MCGIMs yield better geometric approximations than those of the semi-regular remeshing of Guskov et al. [2000] and Lee et al. [1998].

Several areas of future work remain, especially MCGIM level-of-detail control and compression. We are also interested in rendering MCGIMs using graphics hardware, and exploiting their simplicity in future architectural designs.

References

CARR, N., AND HART, J. 2002. Meshed atlases for real-time procedural solid texturing. *ACM Transactions on Graphics. 21 (2)*, pp. 106-131.

CIGNONI, P., MONTANI, C., ROCCHINI, C., AND SCOPIGNO, R. 1998. A general method for recovering attribute values on simplified meshes. *IEEE Visualization 1998*, pp. 59-66.

ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. 1995. Multiresolution analysis of arbitrary meshes. *SIGGRAPH 1995*, pp. 173-182.

FUCHS, H., KEDEM, Z., AND USELTON, S. 1977. Optimal surface reconstruction from planar contours. *Comm. ACM* 20.

GARLAND, M., WILLMOTT, A., AND HECKBERT, P. S. 2001. Hierarchical face clustering on polygonal surfaces. *2001 ACM Symposium on Interactive 3D Graphics*, pp. 49-58.

GU, X., GORTLER, S. J., AND HOPPE, H. 2002. Geometry images. *SIGGRAPH 2002*, pp. 355-361.

GUSKOV, I., VIDIMCE, K., SWELDENS, W., AND SCHRÖDER, P. 2000. Normal meshes. *SIGGRAPH 2000*, pp. 95-102.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1993. Mesh optimization. *SIGGRAPH 1993*, pp. 19-26.

KOBBELT, L., VORSATZ, J., LABSIK, U., AND SEIDEL, H.-P. 1999. A shrink wrapping approach to remeshing polygonal surfaces. *Eurographics 1999*, pp. 119-130.

KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W. 2000. Progressive geometry compression. *SIGGRAPH 2000*, pp. 271-278.

LEE, A., MORETON, H., HOPPE, H. 2000. Displaced subdivision surfaces. *SIGGRAPH 2000*, pp. 85-94.

LEE, A., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. 1998. MAPS: Multiresolution adaptive parameterization of surfaces. *SIGGRAPH 1998*, pp. 95-104.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *SIGGRAPH 2002*, pp. 362-371.

LLOYD, S. P. 1957. Least squares quantization in PCM. Unpublished, reprinted as *IEEE Trans. Inform. Theory, IT-28(2)*:127-135, 1982.

MAILLOT, J., YAHIA, H., AND VERROUST, A. 1993. Interactive texture mapping. *SIGGRAPH 1993*, pp. 27-34.

MAX, J. 1960. Quantizing for minimum distortion. *IEEE Trans. Inform. Theory, IT-6(1)*: pages 7-12, March 1960.

MILENKOVIC, V. 1998. Rotational polygon containment and minimum enclosure. *Proc. of 14th Annual Symposium on Comp. geometry*, ACM.

PIPONI, D. AND BORSHUKOV, G. D. 2000. Seamless texture mapping of subdivision surfaces by model pelting and texture blending. *SIGGRAPH 2000*, pp. 471-478.

SANDER, P. V., GORTLER, S. J., SNYDER, J., AND HOPPE, H. 2002. Signal-specialized parametrization. *Eurographics Rendering Workshop 2002*.

SANDER, P. V., SNYDER, J., GORTLER, S. J., AND HOPPE, H. 2001. Texture mapping progressive meshes. *SIGGRAPH 2001*, pp. 409-416.

SHEFFER, A. AND HART, J. C. 2002. Seamster: Inconspicuous Low-Distortion Texture Seam Layout. *IEEE Visualization 2002*.

SHLAFMAN, S., TAL, A., AND KATZ, S. 2002. Metamorphosis of polyhedral surfaces using decomposition. *Eurographics 2002*, pp. 219-228.

SORKINE, O., COHEN-OR, D., GOLDENTHAL, R., AND LISCHINSKI, D. 2002. Bounded-distortion piecewise mesh parameterization. *IEEE Visualization 2002*.

TURK, G. AND LEVOY, M. 1994. Zippered polygon meshes from range images. *SIGGRAPH 1994*, pp. 311-318.

VORSATZ, J., RÖSSL, C., KOBBELT, L. P., AND SEIDEL, H. 2001. Feature-sensitive remeshing. *Computer Graphics Forum, 20(3)*:393--401, 2001.

WOOD, Z. J., DESBRUN, M., SCHRÖDER, P., AND BREEN, D. 2000. Semi-regular mesh extraction from volumes. *IEEE Visualization 2000*, pp. 275-282.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. *SIGGRAPH 1997*, pp. 259-268.

© The Eurographics Association 2003.

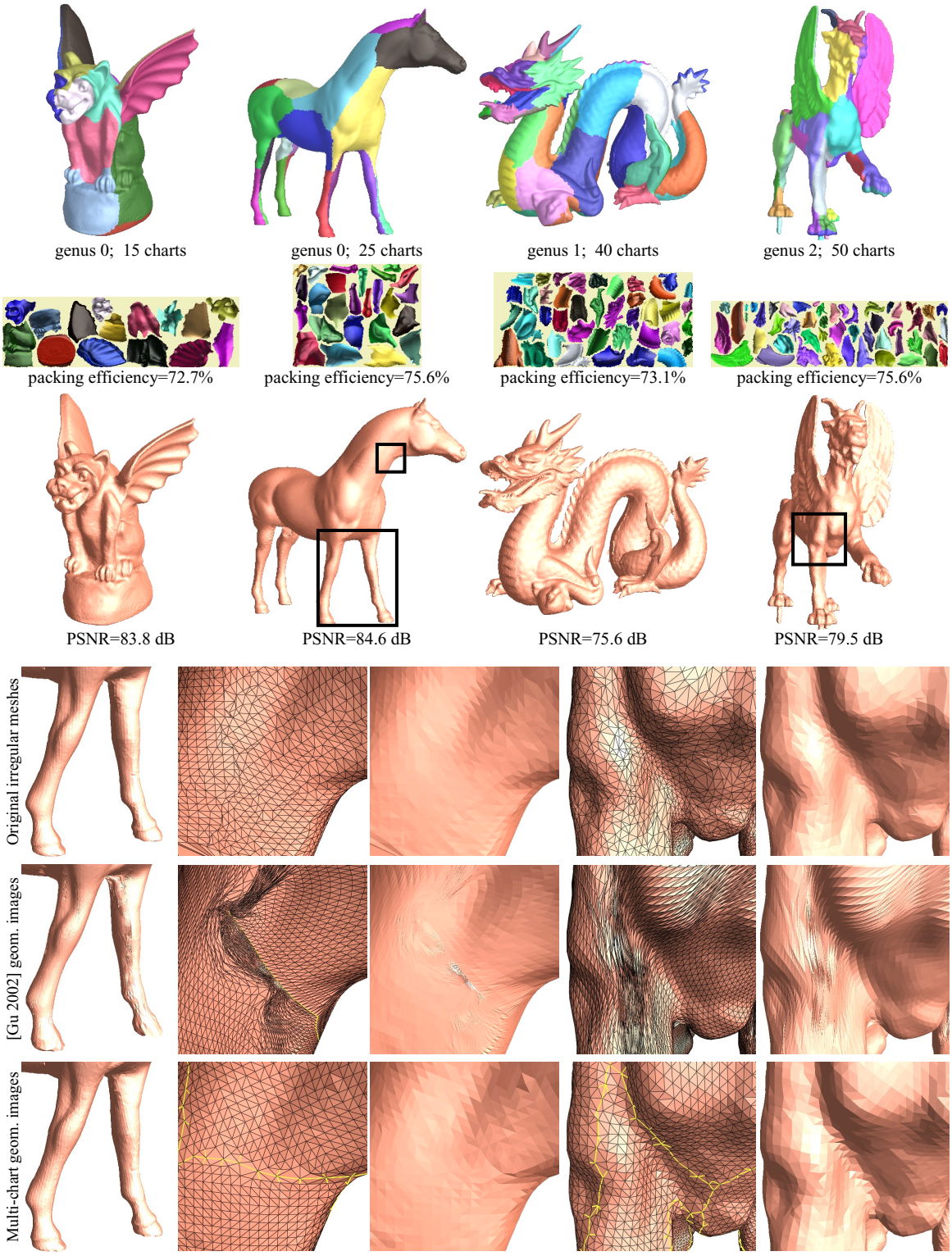


Figure 14: *Multi-chart geometry image results. Black squares indicate close-up views used in comparisons below.*

© The Eurographics Association 2003.