Surfacing By Numbers

Steve Zelinka* Michael Garland†

Department of Computer Science
University of Illinois at Urbana-Champaign

ABSTRACT

We present a novel technique for surface modelling by example called *surfacing by numbers*. Our system allows easy detail reuse from existing 3D models or images. The user selects a source region and a target region, and the system transfers detail from the source to the target. The source may be elsewhere on the target surface, on another surface altogether, or even part of an image. As transfer is formulated as synthesis with a novel surface-based adaptation of graph cuts, the source and target regions need not match in size or shape, and details can be geometric, textural or even user-defined in nature.

A major contribution of our work is our fast, graph cut-based interactive surface segmentation algorithm. Unlike approaches based on scissoring, the user loosely strokes within the *body* of each desired region, and the system computes optimal boundaries between regions via minimum-cost graph cut. Thus, less precision is required, the amount of interaction is unrelated to the complexity of the boundary, and users do not need to search for a view of the model in which a cut can be made.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture I.4.6 [Image Processing and Computer Vision]: Segmentation—Edge and feature detection I.4.7 [Image Processing and Computer Vision]: Feature Measurement—Size and shape

Keywords: texture synthesis, geometry synthesis, selection

1 INTRODUCTION

Most children have been exposed to the idea of "colouring by numbers". A drawing or cartoon is divided up into numbered regions, and a table maps numbers to suggested colours. The child colours each region of the drawing by looking up its number in the table and filling it in with the corresponding colour. Effectively, the original, complexly-coloured drawing is segmented such that each segment becomes easy to colour.

We apply this metaphor to the task of 3D surface modelling. In particular, we create or edit some *signal* over a 3D surface, such as fine-scale geometry, colours, or texture maps. With current tools, this task can be extremely tedious and repetitive, and reusing detail from existing models is difficult. In *surfacing by numbers*, the target surface is segmented into regions, and data-driven synthesis produces the detail within each region. Thus, the modelling task reduces to producing an appropriate segmentation, and finding an example of desired detail in each target region. This division of work is well-suited to both human and machine capabilities: humans excel at high-level semantic segmentation, and data-driven synthesis avoids the tedium tedium of producing the detail.

*szelinka@gmail.com
†garland@uiuc.edu

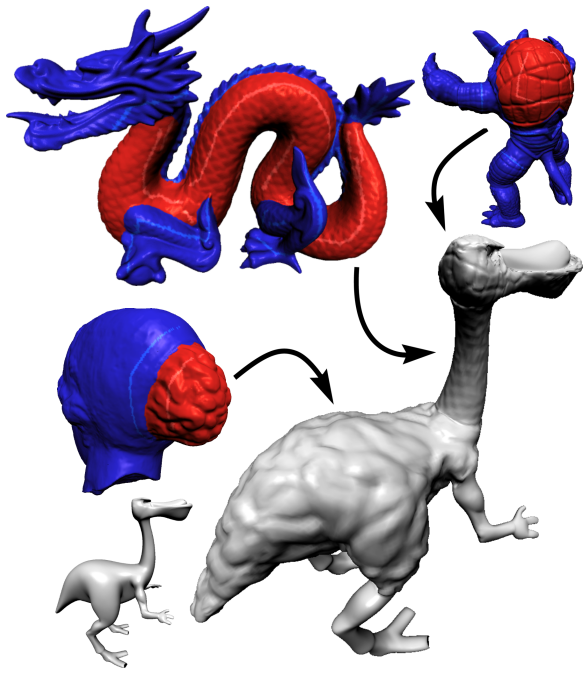

Figure 1: Reusing detail with surfacing by numbers.

Our system uses a novel interactive *region-centric* approach to surface segmentation. The user identifies small areas which belong to each desired region with a few mouse strokes, and the system discovers a good set of boundaries. Region-centric segmentation is very useful for *any* interactive part-of-surface selection task. As this is a fundamental operation within any conventional 3D modelling system, the approach is widely applicable.

Region-centric segmentation has a number of advantages over previous interactive segmentation techniques. Most previous work is based on scissoring, in which the user draws a cut through the model which is automatically completed or refined. Our approach requires less precision as users aim for the bodies of regions rather than their boundaries. The amount of user interaction depends on the number of regions, rather than the complexity of their boundaries, and we also avoid occlusion problems. With scissoring, it can be difficult to find a view in which a sufficiently close cut can be drawn due to self-occlusion. Region-centric selection naturally uses multiple strokes in different views as needed.

We effectively extend data-driven synthesis to sources defined over *manifold* domains. Thus, we can flexibly reuse signals from existing 3D models, such as fine-scale geometric detail. Users can pull different kinds of detail from a collection of existing 3D models and novelly combine them to produce new models. As 3D scanning technology matures, yielding ever more detailed models, the capa-

bility to reuse detail becomes increasingly important. Our region-centric segmentation algorithm further facilitates such reuse, providing fast and intuitive selection of complex regions from existing models.

2 Related Work

Surfacing by numbers complements existing methods for detail transfer. One class of method is based on common or cross-parameterization of models [25, 2] which allow easy transfer of attributes through the parameterization. A significant issue with these methods is parametric distortion. More recent work directly parameterizes models over each other [16, 28], and allows for transfer of mesh deformations [33]. These methods can lower the requisite distortion, but there is still no guarantee any good common or cross-parameterization exists.

Cut-and-paste surface editing [4] uses a local parameterization to transfer geometry among surfaces, avoiding much of the distortion required in globally consistent parameterization methods. Sorkine *et al.* [32] transfer Laplacian coordinates to better adapt the transferred geometry to the target shape. For large-scale geometry, surface stitching based on Poisson interpolation [38] yields excellent results, though good results can also be achieved with relatively unsophisticated blending methods [9]. These methods assume enough desired detail is available for cutting and pasting.

The texture synthesis community has extensively explored techniques for synthesizing detail over surfaces [24, 36, 35, 30, 40]. Our work is inspired by the "texture by numbers" system of Hertzmann *et al.* [12], which operates entirely in the image domain. All of these methods critically rely on the regular sampling and topology of the example data. Surfacing by numbers generalizes graph cut-based synthesis [18, 37] to allow 3D manifolds to serve as example data as well. Other work in the area includes Sharf *et al.* [29], who use implicit function matching to complete point-based surfaces, and synthesis over volumetric envelopes of a surface [23] or from volumetric sources [3].

Interactive surface segmentation has not received a great deal of attention from the computer graphics research community. Funkhouser *et al.* [9] and Lee *et al.* [19] use scissoring-based approaches, with all the drawbacks highlighted previously. Automatic segmentation methods have received much more attention [15, 7], but these methods' assumptions about a basis for segmentation are not compatible with our problem. In the image domain, there have been significant recent advances in segmentation using graph cuts [26, 20, 1], and our region-centric surface segmentation algorithm, described in detail next, is inspired by these image domain techniques.

3 Interactive Surface Segmentation

Region-centric surface segmentation augments a user's ability to select an arbitrary desired region of a surface. Our approach builds on the pioneering work of Boykov and Jolly [5] in the image domain. We generalize this work to 3D surfaces, and develop an efficient multi-resolution approach for high resolution meshes.

3.1 Background

Boykov and Jolly phrase interactive image segmentation as labelling each pixel of an image as either foreground or background. They transform this problem into a network flow problem, so the minimum cost cut of the network flow problem yields the segmentation.

The network uses a terminal for each of the two regions of the segmentation, and one node per pixel. The pixels are connected to their image neighbours and also to the terminals. The terminal

Figure 2: Interactive surface segmentation of bunny and sculpture models. Our system excels at scissoring tasks, such as with the bunny, but can easily handle much more complex selections. The bunny uses only two strokes, while the figure in the sculpture (shown front and back) was captured with only about a dozen.

to which a pixel is connected after the cut is computed gives its segmentation label.

Users interactively stroke pixels as fixed-foreground or fixed-background. Each fixed pixel's node gets an unbreakable edge to the appropriate terminal. Region-wide statistics are collected over each set of fixed pixels to assign predictive weights to each unfixed pixel's links to the terminals. Pixel to pixel edges are weighted by the perceptual cost of a boundary between the pixels, and are usuall inversely proportional to the image gradient. The solution can be updated on each brush stroke, so the approach degrades gracefully toward manual brushing.

3.2 Surface Segmentation

We follow a similar approach for selecting a region from a 3D surface. We make no assumptions about the input other than that it is an irregular triangle mesh. Users' fixed-foreground/background strokes are projected onto the mesh with a standard object ID buffer.

Our flow network uses two terminal nodes, a node for each vertex, connects neighbouring mesh vertices, and connects all vertices to each terminal. Fixed-foreground/background nodes are selected by projecting strokes on the mesh with a standard object buffer. Below, we detail our choices of *boundary* costs (weights for edges between mesh vertices) and *region-based* costs (for edges to terminal nodes). Note we must weight all costs by the area of the corresponding vertex neighbourhoods to account for irregular sampling.

3.3 Boundary-based Costs

Boundary-based costs guide the placement of the cut, directly penalizing cuts located in perceptually poor areas. For segmenting surfaces, a well-known perceptual result is the *minima rule* [13], which states that people tend to divide objects into parts at negative minima of curvature. Thus, the dihedral angle θ_{ij} between the faces

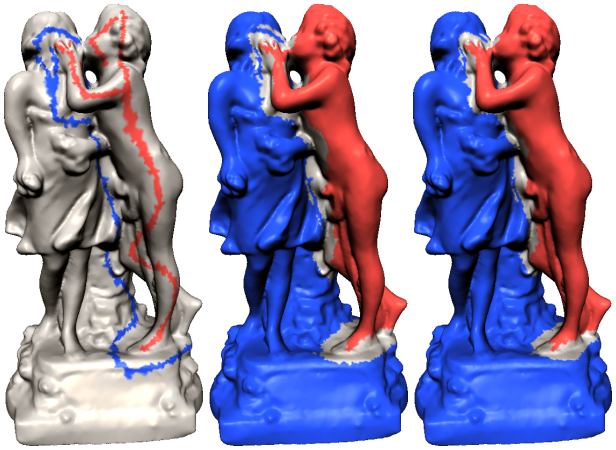

Figure 3: Multi-resolution segmentation, coarser to finer, left to right. Coloured faces are fixed at that resolution, so the graph cut problem is restricted to the gray areas (fixed coarse level faces correspond to the user's brush strokes.)

incident to vertices i and j is a key component. In addition, we generally prefer a short straight cut through a region to a convoluted one, so we try to minimize edge length d_{ij}. Our boundary-based cost e_{ij} is:

$$e_{ij} = \eta \frac{2\pi - \theta_{ij}}{2\pi} + \gamma \frac{d_{ij}}{\overline{d}} \quad (1)$$

where η and γ control the relative weight of the length and dihedral angle terms (we typically set both equal to one), $\overline{d}$ is the average edge length, and a dihedral angle π is flat, increasing with concavity.

3.4 Region-based Costs

Region-based costs assume the desired regions are measureably distinct. A statistical model of each is built using the fixed vertices in order to predict membership of unfixed vertices. In practice, *no* geometric region model is necessary for most selection tasks on surfaces. The boundary-based costs alone do a very good job, as shown in our results. We tested numerous models, such as histograms of local curvatures, position-based clustering, projections of Laplacian coordinates [32], and geodesic fans [39]. Only geodesic fans produce an qualitative improvement, and only when the desired selection is characterized by a particular kind of detail, but are too computationally expensive for interactive use. If the surface supports other signals salient for selection, such as colours or textures, they can be incorporated into a region-based model using k-means clustering [20, 26].

3.5 Multiresolution Segmentation

Computing the minimum cost graph cut can be expensive. Interactive response time can be maintained for graphs on the order of tens of thousands of nodes, but we would like to handle denser meshes.

To do so, we first cluster vertices into a static level of detail hierarchy with vertex correspondences, allowing us to map the fixed vertices at the finest level into the coarsest level. To avoid ambiguity, we prevent clusters containing fixed vertices from different regions. We solve the minimum cost graph cut as before but over the clusters instead of individual vertices. The segmentation is then refined to the next finer level of detail, and the process is repeated. However, the graph in this finer-level cut is limited to only those clusters near the boundaries of the coarser-level cut. Thus, the time complexity of the algorithm becomes proportional to the number of edges in the cut, rather than the number of edges in the entire mesh. This process is visualized in Figure 3, where only gray areas are unfixed.

In constructing the level of detail hierarchy, we wish the most salient details of the mesh to be preserved. We use quadric-based simplification [10] as it is extremely fast and produces perceptually reasonable results, especially given that we do not need to simplify to extreme levels. In practice, we simplify to a coarsest level on the order of thousands to tens of thousands of vertices, using two to four levels of detail. Response time is interactive for up to 100k vertices. On larger models, the solver can be invoked less often or only when prompted.

4 Surfacing By Numbers

Surfacing by numbers enables easy, flexible reuse of geometric or textural detail. User interaction amounts to iteratively selecting regions of the target surface and their corresponding source regions, and these selection tasks are made particularly easy by our fast, intuitive segmentation algorithm above. The main purpose of the user-driven segmentation is decompose the user's desired signal over the entire target surface into regions where it is relatively homogenous and suitable for data-driven synthesis. The key remaining question is how detail is transferred.

We assume detail is encapsulated within a *signal* ϕ defined over the source region, $\phi : M_S \rightarrow R^n$. Example signals used in this paper include texture maps, which evaluate to a colour at each point on the surface, and displacement fields, which evaluate to a local offset vector (not necessarily in the normal direction). While the signals we have used are typically piecewise linear, or mapped into a piecewise linear domain (e.g., a texture), it is not required.

As the user establishes each regional correspondence, the system synthesizes detail over the target region resembling the detail present in the source region. The synthesis process iteratively copies patches of the signal from the source region to the target region, and is similar to that used in Graphcut Textures [18]. For each face of the target region, we maintain a source patch ID and a geometric mapping between the face and corresponding part of the source region from which its signal is copied. We perform the following four steps iteratively:

1. **Target position selection**. A position for a new patch is selected on the target surface.

2. **Source patch selection**. A position within the source region is selected that closely matches existing detail around the selected target position.

3. **Mutual patch parameterization**. The source and target patches are parameterized in a common domain.

4. **Patch trimming**. Boundaries are computed between the proposed new patch and existing surface detail.

Finally, once the user is happy with the results, a final extraction step is performed. In the following sections, we explain each of these steps in detail. We then highlight some issues involved with geometry transfer, and finish our algorithm description with a discussion of some implementation issues.

4.1 Target Position Selection

We wish to find a position within the target region over which a new patch of source detail should be placed. We grade each face according to the sum of the cost of all *seams* (see below) within a fixed geodesic distance, and select randomly from the highest. In grading, seams between placed patches and uncovered regions

are given maximal signal difference. This policy encourages the algorithm to place new patches overlapping with as-yet-uncovered areas, as well as overtop areas with bad seams. In practice, the exact location of the target position does not seem critical to good results.

4.2 Source Patch Selection

Once a target position is selected, we wish to find a source patch that is a good match for the existing detail already transferred near that position. Facing a similar problem in their image domain synthesis algorithm, Kwatra *et al.* [18], depending on the characteristics of the source signal, choose either random patches, or search for the best matching position within the source image for the context of the target position. Choosing a random patch remains a viable option. However, matching patches across surfaces is an extremely difficult problem due to the irregular sampling of triangle meshes and inherent parametric distortion.

We solve this problem using matching based on geodesic fans [39]. Briefly, a geodesic fan is a surface analogue to a pixel neighbourhood, and is constructed by tracing a fan of geodesics across the surface, sampling a signal at equally spaced points along the geodesics. We preprocess the source region, sampling a geodesic fan at each face and building a search structure for them for efficient matching. To select a source patch, we sample a geodesic fan at the selected target position (note this may only be a partial geodesic fan if the signal doesn't completely cover the nearby surface), and find a good match using the search structure. As geodesic fan matching is orientation independent, the source signal may be reused in any orientation, which further increases perceived randomness (of course, the matching may also be trivially constrained to certain orientations, for example by placing a vector field over the source). The match also provides an *orientation* correspondence, so given a direction within the surface at one point, we know the corresponding direction on the other surface. This orientation correspondence is used to start the mutual patch parameterization.

4.3 Mutual Patch Parameterization

In order to copy detail from one surface to another, we must have a dense correspondence between the surfaces. The previous step provides one point correspondence between the surfaces, but this is clearly not sufficient. Given this initial correspondence, we grow a pair of corresponding parameterized patches across each surface. The main requirement of these parameterizations is that they must be low distortion. Indeed, in our case it is better to halt patch growth entirely than accept highly-distorted triangles, since those triangles can simply be covered by new patches in later iterations. In addition, to make sure that there is a valid signal to copy, we must ensure that the parameterized target patch is entirely contained within the parameterized source patch.

We adopt an iterative bounded-distortion flattening approach similar to that of Sorkine *et al.* [31], and also to similar to Lapped Textures [24]. The triangles containing the two points of correspondence are rigidly flattened into the plane while respecting the correspondence's orientation. The source patch is parameterized by flattening vertices in order of the amount of texture stretch [27] induced to their incident triangles, halting when flattened triangles violate a distortion bound or self-intersect, as in Sorkine *et al.*. The parameterization of the target patch is similar, except that when a vertex is selected to be added to the patch, we must first verify that each of its incident triangles being added to the patch fall within the source patch's parameterization. However, this is easily done using the same global self intersection test applied during the construction of the source parameterization. An example of the results achieved is in Figure 4.

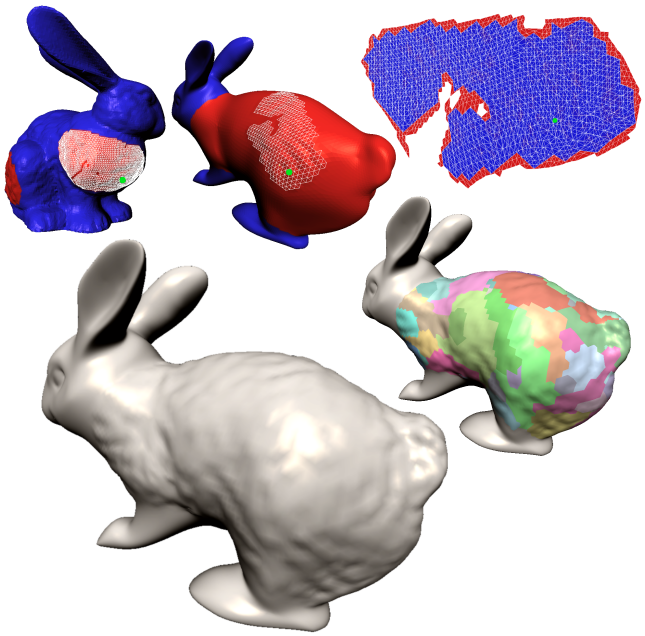

Figure 4: Mutual patch parameterization. The top row shows a parameterized source patch on the Stanford bunny model (left) and a target patch grown on a second bunny model. The parameterized patches are shown at right (target patch in blue). The data-driven synthesis result after a number of iterations is shown below, along with a visualization of the patches copied.

Once the mutual parameterization is complete, we have for each parameterized triangle in the target a mapping onto the source surface. Note that this mapping may be fairly complex. Each target triangle edge has been mapped to a piecewise linear curve on the source.

4.4 Patch Trimming

At this point, we have a set of faces from the target region, each with a low-distortion mapping onto a corresponding part of the source region. Each target face is also either uncovered, or has an existing mapping within the source region from a previous patch placement. The final step of our algorithm is to trim the borders of the new source patch so that they match well with the existing detail already copied.

We use another graph cut-based optimization procedure to solve this problem, with each face of the target region corresponding to a graph node. Again, there are two terminal nodes, an *existing* node T_E, for the existing patches on the surface, and a *proposed* node T_P, representing the new patch. Since nodes are faces, the basic topology of the graph is the dual of the target mesh. The node of any face not covered by the new patch is given an infinite weight link to T_E, and vice versa. Additionally, the face selected in Step 1 (§4.1) is given an infinite weight link to T_P. Edges connecting face nodes are weighted by the cost of the seam they would create.

A *seam* connects two faces with different patch IDs, and has a cost equal to the integral of the signal difference between the two patches along the edge connecting the faces. Each patch maps the connecting edge between the two target faces to a piecewise linear curve on the source surface. These two seam curves are arclength-parameterized as parametric distortion may give them different lengths. For colour or texture signals, we then simply uniformly sample both curves, and sum the corresponding sample differences to evaluate the path integral (we discuss its evaluation for displace-

Figure 5: Geometric detail from the armadillo model (Figure 1) is transferred to the hand model using our graph cut-based data-driven synthesis algorithm.

ment fields below). Note that area-based weighting is implicit here, as longer path integrals evaluate to higher differences.

The minimum cut of the resulting graph produces the trimmed patch. Faces still connected to T_E after the cut retain their current mappings to the source surface, while those connected to T_P are given new mappings from the new source patch. Note that as in Graphcut Textures, seam nodes must be used to augment the graph in order to encourage old seams to be replaced or overwritten.

4.5 Results Extraction

Once iteration is complete, a final transformed mesh can be extracted. Our extraction proceeds triangle by triangle over the target region. In many cases, especially with textured source data, or when the target triangles are small in comparison to source triangles, it is sufficient to simply copy texture coordinates or colours at the corners of each face from their corresponding position on the source mesh. More generally, we can use the direct mapping of each target face onto the source surface to straightforwardly produce a *metamesh* that is guaranteed to faithfully reproduce all of the detail from the source. As a metamesh can have an extremely high complexity, a better solution is to adaptively refine the target by sampling the difference between the piecewise linear reconstruction of the signal and the actual values from the corresponding source (e.g.,, at the center of each edge and center of the face); in many cases, the piecewise linear reconstruction will suffice (this approach mimics that used in ADF construction [8]).

4.6 Geometry Transfer

In order to transfer geometric detail from one surface to another, as demonstrated in Figure 5, we first construct a displacement field over the source surface by smoothing [11, 32]. We use geodesic fan-based bilateral smoothing [39], though any reasonable smoothing algorithm may be used. We copy displacement vectors (which needn't be in the normal direction) through a pair of corresponding local frames at the source and target positions, derived using the mutual parameterization. Also note that path integrals for computing graph edge weights may be calculated analytically for displacement fields as the integral of the displacement vector difference, since the field is piecewise linear by construction. In general, the target surface should be relatively smooth for good results by this method.

Figure 6: Surfacing by numbers from a single image. The image does not contain enough data to cover the entire model. With two source regions in the skin and hair, surfacing by numbers synthesizes detail over the whole 3D model.

4.7 Discussion

There are a number of potential extensions of our system. One possibility is to use texton differences [21], rather than straight signal differences, in the path integral evaluation. This would lead to a more accurate difference measurement, as textons better capture local area information, but they also require extra pre-processing. Similarly, higher quality may be achieved by applying deformations to a source patch as it is placed on the target, to match existing features [37]. High quality can also be achieved by computing a metamesh after each patch placement, since patches can then be trimmed to source features, but this very quickly explodes the size of the output. Also, note that our system is independent of the actual signal being measured or copied. It may thus be extended trivially to richer surface descriptions such as BTF's [34] or shell or volume textures [6, 23]. Another extension would be to support progressive scale variation [40] across the surface. Currently, we simply assume a global scale has been fixed between the source and target regions, but there is no reason this scale could not be varied across the target surface, since all scale-dependent calculations (e.g., flattening a triangle, computing a path integral) are local. Indeed, scale could even be varied across the *source* region, to account for signal distortion already present within it.

In our results, we have applied only simple blending kernels between patches copied to the target surface, and between the boundary of a target region and the rest of the model. Higher quality could likely be achieved by applying more sophisticated blending based on the Poisson equation [38]. In addition, it can be useful to resynthesize the boundary of two target regions if their source regions share a common boundary.

There is a non-obvious tradeoff between allowable distortion and the quality of results. As one would expect, larger allowable distortion in the mutual parameterization produces worse quality results due to the visible parametric distortion. However, a certain amount of distortion is required for good results, as lower distortion bounds lead to less regularly-shaped patches. Without regularly-shaped patches, it is difficult for data-driven synthesis to capture lower-frequency components of the source signal. In practice, we use distortion bounds on the order of 1.5 to 3 (i.e., stretching or shrinking in any one direction by at most a factor of 3).

5 Results

Our system runs interactively on a standard 1.5 Ghz PC for medium sized meshes (up to a hundred thousand vertices). Users can select

from meshes in real-time, the segmentation being updated in under a second. Once correspondences are established between target and source regions, detail synthesis is performed offline, usually taking a few minutes. The primary cost of detail synthesis is the mutual parameterization, followed by patch finding. Processing times per source mesh may not be trivial, but usually are on the order of minutes.

In terms of amount of interaction required, our segmentation algorithm performs comparably to scissoring, generally requiring a pair of strokes for simple scissoring operations (slicing an extremity off). However, typically only a handful of strokes is required even for relatively arbitrary selections, such as the back of the bunny model (Figure 7). The sculpture selection in Figure 2, made using only about a dozen strokes, is quite complex and would be difficult to achieve with scissoring methods, since there is no one view of the model in which a reasonably accurate cut can be stroked for this selection. Note that we have used segmentation boundaries limited to a subset of the edge set to simplify the processing for surfacing by numbers (no partially-covered triangles), but it would be straightforward to smooth the cut in a post-process to produce straighter boundary curves.

Surfacing by numbers is demonstrated using a variety of geometric source signals in Figures 1, 4, and 5. Surfacing by numbers can easily accomodate image sources as well (Figure 6). Using a synthesis-based approach in this case can be particularly useful, especially when the source image does not contain enough detail to usefully cover the target model (a common occurrence if one only has a single photograph of a subject). Note, however, that our approach is not appropriate for transferring specific image *features* to a target model, such as eyes or lips, in which small-scale features of the image must align to features of the geometry. Using our system for such tasks can become tedious, and an approach such as Matchmaker [17] is better. A hybrid system, using synthesis for textural regions and constrained texture mapping for specific features, would likely be very good for this application, as the synthesis process relieves distortion pressure on the constrained texture mapping.

As demonstrated in Figure 7, our data-driven synthesis algorithm may also be used in geometry completion [29], or as a part of context-aware hole-filling of scanned data. Given a smooth interpolatory surface over the hole (commonly generated as a "final" result in recent hole-filling methods [22, 14]), surfacing by numbers provides a fast and intuitive way to adapt the interpolatory surface to contain detail similar to that of the mesh.

6 Conclusions and Future Work

It has been well-demonstrated in the data-driven modelling literature that high-level semantic segmentation is difficult to do automatically, while high-quality low-level signal reproduction is relatively easy. Humans, however, are well-suited to high-level segmentation, but find reproduction tasks tedious and error-prone. Surfacing by numbers combines our high-level intuition with data-driven synthesis to provide a system leveraging the best of human machine capabilities. This frees the user to focus on the creative process, selecting elements for designing and detailing surfaces, rather than the mechanical, often tedious process of detail creation. Surfacing by numbers is essentially an "eyedropper" tool for surface detail, loading up detail from a source region, and applying it to a selected target region. The fast, intuitive surface segmentation algorithm we have presented is not only the basic interaction mechanism for our system, but can also improve the user experience for a vast range of conventional 3D modelling operations.

One drawback of the selection algorithm we have presented is its reliance on the minima rule. For artificial objects, this is often not sufficient, as salient features may be bounded by strictly convex areas, especially with right angles (consider the top of a table). Producing such selections with our method requires full brushing, since it always prefers to place a cut through a nearby flatter region rather than along a convex edge.

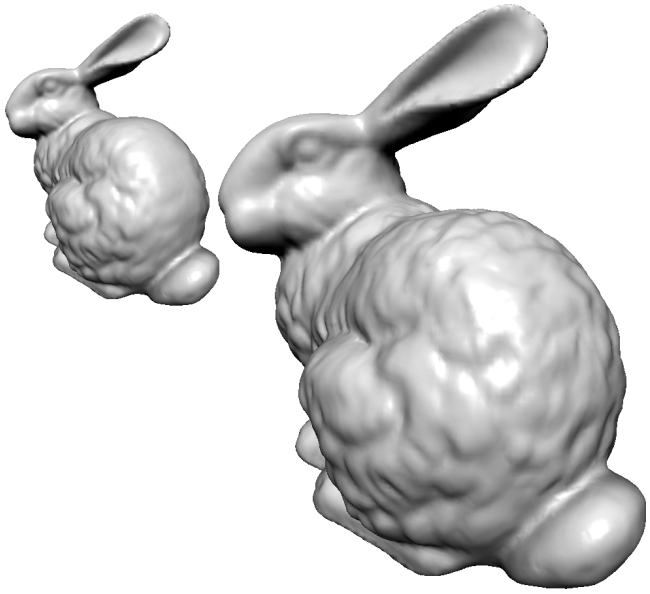

Figure 7: Geometry completion. The back of the bunny, mostly smooth and lacking detail, is enhanced with detail taken from other parts of the model.

We have explicitly avoided making any assumptions about correlations of signals with the geometry of the surface. In practice, colour detail often correlates with geometric structure (e.g., on faces). Exploiting these correlations is a fascinating area for future research. It may be possible to plausibly transfer highly complex signals that are typically expensive to compute, such as precomputed radiance transfer coefficients, dynamic deformation response modes, or simulation results such as erosion and weathering.

References

[1] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David H. Salesin, and Michael F. Cohen. Interactive digital photomontage. *ACM Transactions on Graphics*, 23(3):294–302, 2004.

[2] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes. *ACM Transactions on Graphics*, 22(3):587–594, 2003.

[3] Pravin Bhat, Stephen Ingram, and Greg Turk. Geometric texture synthesis by example. In *Proceedings of the Second Eurographics Symposium on Geometry P rocessing*, pages 43–46. Eurographics Association, July 2004.

[4] Henning Biermann, Ioana Martin, Fausto Bernardini, and Denis Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM Transactions on Graphics*, 21(3):312–321, 2002.

[5] Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Proceedings of the Eighth IEEE International Conference on Computer Vision*, volume 1, pages 105–112. IEEE Computer Society Press, 2001.

[6] Yanyun Chen, Xin Tong, Jiaping Wang, Stephen Lin, Baining Guo, and Heung-Yeung Shum. Shell texture functions. *ACM Transactions on Graphics*, 23(3):343–353, 2004.

[7] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–915, 2004.

[8] Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proceedings of SIGGRAPH 2000*, pages 249–254. ACM SIGGRAPH, July 2000.

[9] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. *ACM Transactions on Graphics*, 23(3):652–663, 2004.

[10] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 97*, pages 209–216. ACM SIGGRAPH, August 1997.

[11] Igor Guskov, Wim Sweldens, and Peter Schröeder. Multiresolution signal processing for meshes. In *Proceedings of SIGGRAPH 99*, pages 325–334, 1999.

[12] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of SIGGRAPH 2001*, pages 327–340, 2001.

[13] D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition*, 18(1–3):65–96, 1984.

[14] Tao Ju. Robust repair of polygonal models. *ACM Transactions on Graphics*, 23(3):888–895, 2004.

[15] Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, 2003.

[16] Vladislav Kraevoy and Alla Sheffer. Cross-parameterization and compatible remeshing of 3d models. *ACM Transactions on Graphics*, 23(3):861–869, 2004.

[17] Vladislav Kraevoy, Alla Sheffer, and Craig Gotsman. Matchmaker: Constructing constrained texture maps. *ACM Transactions on Graphics*, 22(3):326–333, 2003.

[18] Vivek Kwatra, Arno Schödl, Irfan Essa, Grek Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics*, 22(3):277–286, July 2003.

[19] Yunjin Lee, Seungyong Lee, Ariel Shamir, Daniel Cohen-Or, and Hans-Peter Seidel. Intelligent mesh scissoring using 3d snakes. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications*, pages 279–287. IEEE Computer Society Press, 2004.

[20] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM Transactions on Graphics*, 23(3):303–308, 2004.

[21] Sebastian Magda and David Kriegman. Fast texture synthesis on arbitrary meshes. In *Proceedings of the Eurographics Symposium on Rendering 2003*, pages 82–89. Eurographics Association, 2003.

[22] Fakir S. Nooruddin and Greg Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):191–205, 2003.

[23] Jianbo Peng, Daniel Kristjansson, and Denis Zorin. Interactive modeling of topologically complex geometric detail. *ACM Transactions on Graphics*, 23(3):635–643, 2004.

[24] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. In *Proceedings of SIGGRAPH 2000*, pages 465–470, New Orleans, LA, July 2000. ACM SIGGRAPH.

[25] Emil Praun, Wim Sweldens, and Peter Schröeder. Consistent mesh parameterizations. In *Proceedings of SIGGRAPH 2001*, pages 179–184. ACM SIGGRAPH, August 2001.

[26] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004.

[27] Pedro V. Sander, John Snyder, Stephen J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings of SIGGRAPH 2001*, pages 409–416. ACM SIGGRAPH, 2001.

[28] John Schreiner, Arul Asirvatham, Emil Praun, and Hugues Hoppe. Inter-surface mapping. *ACM Transactions on Graphics*, 23(3):870–877, 2004.

[29] Andrei Sharf, Marc Alexa, and Daniel Cohen-Or. Context-based surface completion. *ACM Transactions on Graphics*, 23(3):878–877, 2004.

[30] Cyril Soler, Marie-Paule Cani, and Alexis Angelidis. Hierarchical pattern mapping. *ACM Transactions on Graphics*, 21(3):673–680, 2002.

[31] Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization '02*, pages 355–362. IEEE Computer Society Press, 2002.

[32] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Proceedings of the Second Eurographics Symposium on Geometry P rocessing*, pages 179–188. Eurographics Association, July 2004.

[33] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics*, 23(3):399–405, 2004.

[34] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics*, 21(3):665–672, 2002.

[35] Greg Turk. Texture synthesis on surfaces. In *Proceedings of SIGGRAPH 2001*, pages 347–354. ACM SIGGRAPH, 2001.

[36] Li-Yi Wei and Mark Levoy. Texture synthesis over arbitrary manifold surfaces. In *Proceedings of SIGGRAPH 2001*, pages 355–360. ACM SIGGRAPH, 2001.

[37] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics*, 23(3):364–367, 2004.

[38] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Transactions on Graphics*, 23(3):644–651, 2004.

[39] Steve Zelinka and Michael Garland. Similarity-based surface modelling using geodesic fans. In *Proceedings of the Second Eurographics Symposium on Geometry P rocessing*, pages 209–218. Eurographics Association, July 2004.

[40] Jingdan Zhang, Kun Zhou, Luiz Velho, Baining Guo, and Heung-Yeung Shum. Synthesis of progressively variant texture on arbitrary surfaces. *ACM Transactions on Graphics*, 22(3):295–302, July 2003.