SHAPE ANALYSIS WITH CROWDSOURCED DATA

XIAOBAI CHEN

A DISSERTATION

PRESENTED TO THE FACULTY

OF PRINCETON UNIVERSITY

IN CANDIDACY FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE

BY THE DEPARTMENT OF

COMPUTER SCIENCE

ADVISER: THOMAS FUNKHOUSER

JUNE 2014

Abstract

With explosion of 3D geometric data and its increasing usage in many applications, ranging from computer-aided design to medicine to paleontology, shape analysis is becoming an important research field. Common to many shape analysis tasks are two sub-problems: 1) segmenting a shape into meaningful parts and 2) identifying important/salient points on a shape. Both are easier for humans than for computers. In this thesis, we revisit these problems, from the angle of using crowdsourced data to learn from humans.

We first investigate the problem of mesh segmentation. By constructing a benchmark of 4300 manually generated segmentations for 380 surface meshes of 19 different object categories, and developing software to analyze 11 geometric properties of segmentations and to compute 4 quantitative metrics for comparison of segmentations, we are able to quantitatively answer "How do people decompose shapes into meaningful parts" and "How algorithms do in comparison with humans". The benchmark is widely adopted for evaluating new segmentation algorithms and prompts emergence of learning-based algorithms for mesh segmentation and labeling.

We then visit the problem of identifying salient points on meshes. Rather than defining saliency bottom-up from low-level geometric features, we start with the social/psychological essence of saliency by investigating Schelling points [1] on 3D meshes. We designed an online pure coordination game that asked people to select points on 3D surfaces that they expect will be selected by other people. We then analyzed properties of the selected points, finding that Schelling point sets are usually highly symmetric, that local curvature properties proposed in previous work are most helpful for identifying obvious Schelling points while global properties (e.g., segment centeredness, proximity to symmetry axis, etc.) are required to explain more subtle features. Based on these observations, we use regression to combine multiple properties into an analytical model that predicts where Schelling points are likely to be on new meshes.

Shared by these two problems are the desire to transfer properties and distributions collected for known meshes to unseen ones. Finally, we propose MeshMatch, a mesh-based analogy of the image-based PatchMatch [2] algorithm and a non-parametric multi-resolution approach to surface property transfer based on this algorithm. This method offers benefits of both parametrization and texture synthesis approaches to preserving both large- and fine-scale patterns of the source properties. We show its applications in texture transfer, detail transfer, texture painting, and transferring Schelling distributions.

Acknowledgements

This thesis is impossible without the support and mentoring of my adviser, Thomas Funkhouser. He sets a role model for me in conducting research and I am also motivated so often by his optimism and tenacity. I also thank David Dobkin, Adam Finkelstein, Dan Goldman and Szymon Rusinkiewicz to join my thesis committee.

I was fortunate to collaborate with Ingrid Daubechies, Aleksey Golovinskiy, Vladimir Kim, Yaron Lipman, Bill Pang, Abulhair Saparov, Eli Shechtman and Jian Sun. Doing projects together with you is both fun and a great learning experience for me.

I am indebted to many people who are always willing to help. Special thanks to Adam Finkelstein and Szymon Rusinkiewicz who offered advices on almost all my projects, to Fei-Fei Li and Jia Deng for getting me aware of crowdsourcing, to Doug DeCarlo and Manish Singh for helpful discussions in early stage of the Schelling Points project, to Daniela Giorgi and AIM@Shape whose mesh data is the backbone of many of my projects and to Tiggraph group for reviewing preliminary paper drafts. Princeton Graphics Group fosters a collegial and supportive environment and I will miss the camaraderie.

I want to thank my internship mentors - Dan Goldman and Eli Shechtman at Adobe, and Claus Brenner, Aleksey Golovinskiy, Tilman Reinhardt at Google for making all my intern experience wonderful. Witnessing so many cool things being done outside of school is very exciting. I also appreciate the funding support from Adobe, Google, NSF (CNFS-0406415, IIS-0612231, CCF-0702672, CCF-0937139, CNS-0831374) and Intel (ISTC-VC).

Finally, I am grateful to people who supported me outside of work and study. Graduate school becomes more joyful with the accompanying of my friends: Peng Jiang, Yiming Liu, Xiaojuan Ma, Lei Qi, Chong Wang, Ming Yang, Wenhan Zhu and others. I cannot be more thankful to my mother Lan Dai and father An Chen - their love encourages me in life.

To parents, to friends. To the past, to the future.

Contents

Chapter 1

Introduction

1.1 Motivation

Shape analysis is a field in computer science to provide annotations for 3D data so that 3D objects could be better represented, recognized, indexed and understood. The research is driven by the explosion of 3D geometric data due to advances in 3D scanning and 3D modeling technologies, as well as by many application domains that use such data, including computer-aided design, urban planning, paleontology, entertainment, medicine and many others.

Two problems fundamental to shape analysis are repetitively seen in many applications - mesh segmentation and salient feature detection. While there have been great research efforts developing automatic algorithms, computers are yet to match what humans do. Motivated by the fact that these tasks are much easier for humans, we try to 1) better understand what people do, and 2) apply what we learn from humans to advance algorithms in addressing these problems.

It is possible to formulate qualitative "rules" to mimic humans based on observations and/or psychological studies and there have been successful algorithms developed by encoding

the rules in a math formula. However, for the problems we consider, the rules can often be vague by themselves, not to mention the fuzziness in encoding them.

Our approach is data-driven, on assumption that large amount of data from humans could be a gold mine with "rules" encoded implicitly. Many problems in other fields are addressed successfully this way, such as natural language translation in machine learning and object recognition in computer vision. We expect the approach to be applicable to the shape analysis problems we investigate as well.

Pivotal to our approach is the gathering of large amount of meaningful data from humans. Traditional lab studies hardly scale to provide the amount of data needed for problems we consider. Fortunately, we see the rise of online crowd-sourcing markets such as Amazon's Mechanical Turk [3]. They provide both easy access to a large pool of human subjects working in parallel and handy tools to handle transaction details. As a result, we can afford to collect human inputs in large scale.

Challenges remain to apply crowdsourcing to solve shape analysis problems: 1) how to design online protocols, incentive structures and interfaces to collect unbiased and high-quality data; 2) how to analyze and mine the data to better understand what people do; 3) how to make use of the data.

The thesis demonstrates how we deal with these challenges in investigating problems of mesh segmentation (Chapter 2) and salient point detection on meshes (Chapter 3). To apply what is learned/gathered from humans to unseen meshes, we introduce a "transfer" algorithm in Chapter 4, and shows its application in transferring textures, details, Schelling distributions and others.

1.2 Related Work

Shape analysis has been increasing its capability in describing and differentiating 3D shapes. This capability can come from local, patch-scale properties on surfaces, such as local curvature estimations, discrete variants of the Laplacian-Beltrami operator [4], and spin images [5] shape descriptors. However, local properties are limited by the smaller support size to describe an object. Thus many work propose using global, large-scale properties to capture global structure of shapes, including symmetry analysis [6, 7], spectral methods [8], histograms of local properties [9, 10] etc.

Global properties are still not powerful enough, especially for problems that relate to higher-level shape understanding. One of these problems is to decompose shapes into meaningful parts. This task is often much easier for humans than for computers [11]. Another problem is detecting salient features on shapes - many local and global features matter but they hardly explain all of what humans will denote [1]. For both problems, we think it is worth learning more from humans and sourcing the wisdom of crowds.

1.2.1 Study of how people do X

Most related to our work are recent studies aiming at understanding how people perform tasks of interest in computer science. For example, Cole et al. [12, 13] studied where do people draw lines when making line drawings, Donovan et al. [14] tested color compatibility theories using color palettes configured by a large number of users, Secord et al. [15] optimized a general model for viewpoint selection by leveraging results of a large user study, and Eitz et al. [16] explored how do human sketch objects. In all cases, machine learning techniques were developed to make use of the valuable insights learned from humans by training on collected examples, including predicting line drawings for new surfaces [12], scoring quality of new color settings [14], selecting best view points for new

meshes [15], and recognizing objects from sketches [16]. This thesis specifically focuses on shape analysis problems: mesh segmentation and salient point detection on meshes.

1.2.2 Perceptual Psychology Studies

Also related is research in perceptual psychology to understand how the human cognitive system perceives things.

Several psychologists have put forth theories to explain how people perceive objects as collections of parts. Biederman [17] proposed the Recognition by Components theory. Hoffman and Richards [18, 19] introduced the minima rule, which states that humans tend to divide 3D shapes into parts along negative minima of the principal curvatures, along their associated lines of curvature. These theories are widely adopted by many algorithms [20, 21], but it is still ongoing research of how to quantitatively apply these theories.

Many perceptual studies aim to understand how people assign importance to regions of objects, especially on visual attention. In the image domain, Koch et al. [22] propose a model that salient points would be ones that are different from their surroundings. Attneave [23] studies which points are most important for approximation of a contour and his main finding was that most people select points where "the contour is most different from a straight line". Privitera et al. [24] use eye fixations measured from eye trackers to evaluate regions of interest in images. Similarly, Kim et al. [25] study how eye fixations relate to mesh saliency [26].

1.2.3 Crowdsourcing

Crowdsourcing is a process that involves outsourcing tasks to a distributed and undefined public. Because humans excel in many tasks that are yet hard for computers, and because Internet connects humans distributed around the world so they could work in parallel, there

has been a rapid growth of using crowdsourcing systems in research communities and in the industry [27, 28].

The massive parallelism of crowdsourcing enables large scale data collection from humans that are unimaginable before. For example, ImageNet [29] collects and organizes images according to the WordNet [30] hierarchy, and employs workers from Amazon's Mechanical Turk [3] to clean up hundreds of millions of images for all the noun nodes. reCAPTCHA [31] improves book digitization by sending words that cannot be read confidently by computers to the Web for humans to decipher in the form of CAPTCHAs [32]. These successes motivate our adoption of using crowdsourced data for shape analysis.

Crowdsourcing also motivates crowd-powered systems [33] and human computation techniques [34, 35] of using human efforts to perform tasks that computers can not yet perform well. Players of Foldit [36] unlock structures of proteins that puzzled academia for years. Ahn et al. [37] ask people to label images via the ESP game. Gingold et al. [38] demonstrate micro human computation in solving perception related tasks.

As crowd-powered systems prove to be useful, researchers start to study design principles for these systems. Mason et al.and Horton et al. [39, 40] studied incentives to humans in the systems. Huang et al. [41] explored how to design such a system more automatically. Bernstein et al. [33, 42, 43] developed the Find-Fix-Verify design pattern to improve work quality [33] and proposed a retainer model to reduce latency of crowd-powered systems for tasks requiring quick responses [42]. They also studied how to create crowds tailored for designer's needs [43].

1.3 Contributions

The thesis makes both research and algorithm contributions in shape analysis.

Chapter 2 describes a mesh segmentation benchmark to quantitatively answer "How do people decompose shapes into meaningful parts" and "How algorithms do in comparison with humans". The results suggest that people are remarkably consistent in the way that they segment most 3D surface meshes, that no one automatic segmentation algorithm is better than the others for all types of objects, and that algorithms based on non-local shape features seem to produce segmentations that most closely resemble ones made by humans. The work was previously published in [11], and has been widely used for evaluating new mesh segmentation algorithms [44, 45, 46], and has helped prompt learning based algorithms [47] for mesh segmentation and labeling.

Chapter 3 investigates "Schelling points" on 3D meshes, feature points selected by people in a pure coordination game due to their salience. To collect data for this investigation, we designed an online experiment that asked people to select points on 3D surfaces that they expect will be selected by other people. We then analyzed properties of the selected points, finding that: 1) Schelling point sets are usually highly symmetric, and 2) local curvature properties (e.g., Gauss curvature) are most helpful for identifying obvious Schelling points (tips of protrusion), but 3) global properties (e.g., segment centeredness, proximity to symmetry axis, etc.) are required to explain more subtle features. Based on these observations, we use regression analysis to combine multiple properties into an analytical model that predicts where Schelling points are likely to be on new meshes. We find that this model benefits from a variety of surface properties, particularly when training data comes from examples in the same object class. This work was also published in [1].

Finally, Chapter 4 describes a system to transfer surface properties between meshes. We propose MeshMatch [48], a mesh-based analogy of the PatchMatch [2] algorithm and a non-parametric multi-resolution approach to surface property transfer based on this algorithm. The system offers benefits of both parametrization and texture synthesis approaches, preserving both large- and fine-scale patterns of the source properties. Computation is ef-

ficient since we leverage the inherent locality and coherence of properties associated with the mesh geometry. We show a variety of applications using this system, including texture transfer, detail transfer, texture painting, and transferring Schelling distributions.

Chapter 2

Mesh Segmentation

2.1 Introduction

Automatic segmentation of 3D surface meshes into functional parts is a fundamental problem in computer graphics. A part decomposition not only provides semantic information about the underlying object, but also can be used to guide several types of mesh processing algorithms, including skeleton extraction [49, 20], modeling [50], morphing [51, 52], shape-based retrieval [53], and texture mapping [54]. All of these applications benefit from mesh segmentations that match what humans do.

Perceptual psychologists have proposed theories of how human decompose shapes into parts, e.g., the "minima rule" [18, 19] and the Recognition by Components theory [17], and many automatic mesh segmentation algorithms have been developed based on the theories in the last several years. However, there has been little work on quantitative evaluation of how well these algorithms match what people do. The main problem has been the lack of a "gold standard." There has been no benchmark set of 3D models and no agreed-upon metrics of success. Rather, most new algorithms are tested on their own set of 3D meshes, and results are presented only as images with different segments shown in different colors.

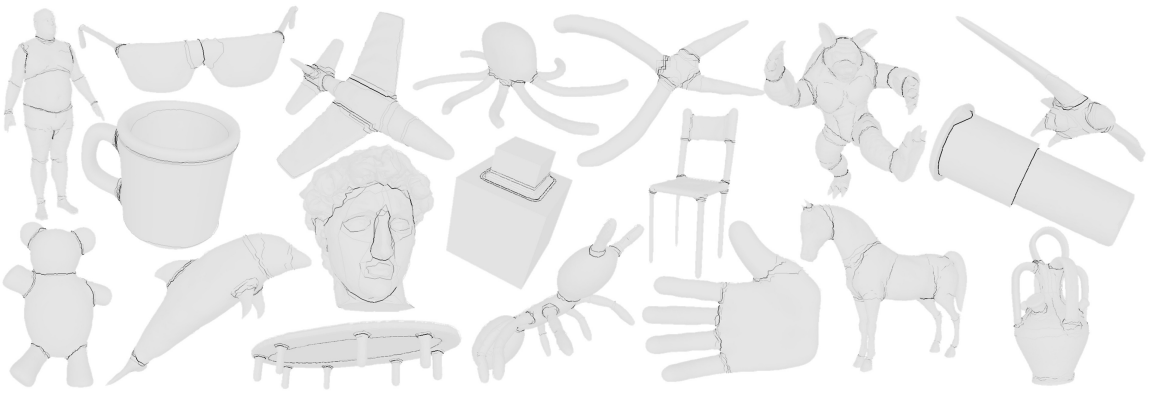


Figure 2.1: *Composite images of segment boundaries selected by different people (the darker the seam the more people have chosen a cut along that edge). One example is shown for each of the 19 object categories considered in this study.*

In a few cases, more than one algorithm have been compared on the same set of meshes (e.g., [55]), but the evaluation is qualitative (images shown side-by-side).

In this chapter, we present a benchmark to quantitatively study how human segment meshes and evaluate mesh segmentation algorithms. To build this benchmark, we recruited eighty people to manually segment surface meshes into functional parts, yielding an average of 11 human-generated segmentations for each of 380 meshes. This data set provides a sampled distribution over "how humans decompose each mesh into functional parts," which we treat as a probabilistic "ground truth" (Figure 2.1). Given this data set, we analyze properties of the human-generated segmentations to learn about what they have in common with each other (and with computer-generated segmentations); and we compute metrics that measure how well the human-generated segmentations match computer-generated ones for the same mesh, which provide a basis for quantitative comparisons of mesh segmentation algorithms.

The contributions of the chapter are five-fold. First, we describe a procedure for collecting a distribution of mesh segmentations from people around the world. Second, we investigate four metrics for comparing mesh segmentations, adapting ideas from computer vision to the domain of 3D meshes. Third, we perform quantitative comparison of seven automatic mesh segmentation algorithms according to these metrics. Fourth, we analyze properties of human-generated and computer-generated mesh segmentations, making ob-

servations about how they are similar and different with respect to one another. Finally, we provide a publicly available data set and software suite for future analysis and comparison of mesh segmentations (http://segeval.cs.princeton.edu/).

2.2 Related Work

Segmentation is a classical problem in processing of images, video, audio, surfaces, and other types of multimedia data. Accordingly, a large number of methods have been proposed for both computing and evaluating segmentations. As methods have matured for each data type, segmentation benchmarks have been proposed for evaluation and comparison of methods [56]. Examples include The Berkeley Segmentation Benchmark for images [57], The TREC Data Set for video [58], The IBNC Corpus for audio [59], and so on. However, to our knowledge, no analogous segmentation benchmarks have been developed for 3D surface meshes before us.

Over the last decade, many 3D mesh segmentation algorithms have been proposed, including ones based on K-means [60], graph cuts [61, 20], hierarchical clustering [62, 63, 61], primitive fitting [64], random walks [65], core extraction [66], tubular multi-scale analysis [67], spectral clustering [68], critical point analysis [69] and so on (recent surveys can be found in [70] and [71]). However, most of these methods have been evaluated only by visual inspection of results, and rarely are comparisons provided for more than one algorithm in the same study.

Perhaps the state-of-the-art in 3D mesh segmentation evaluation is the study of Attene et al. [55]. They compared five segmentation algorithms [64, 66, 20, 67, 72] on eleven 3D surface meshes. Evaluation and comparison was performed by showing side-by-side images of segmented meshes produced by different algorithms with segments shown in different colors. Since no "ground truth" was available, readers were asked to evaluate results visu-

ally, and discussion focused on qualitative characteristics of algorithms (e.g., segmentation type, suitable inputs, boundary smoothness, whether the method is hierarchical, sensitivity to pose, computational complexity, and control parameters).

The focus of our work is to provide quantitative metrics for evaluation of mesh segmentation algorithms. We would like to not only show images of example segmentations, but also provide statistical measures of "how close they are to what a human would do" for a large set of 3D models. In this sense, we follow the work of LabelMe [73], The Berkeley Segmentation Benchmark [57], and other image databases that have built sets of human-generated segmentations and then used them for evaluation of automatic algorithms [74, 75, 76, 77, 78, 79]. We leverage the ideas of this previous work in computer vision, using related metrics to evaluate the similarities in overall structure and boundaries of automatically-generated segmentations with respect to human-generated ones.

Concurrent with our work is a project by Benhabiles et al. [80], which has similar goals and methodology. Our study includes more human subjects, segmentation algorithms, mesh models, object categories, evaluation metrics, and analysis of human-generated segmentations.

2.3 Benchmark Design

The focus of this chapter is to investigate the design of a benchmark for 3D mesh segmentation algorithms. In building such a benchmark, several issues have to be addressed, including "which 3D models to study?," "how to acquire unbiased segmentations from humans?," "how to acquire data from a large variety of people?," "how to measure the similarity between two segmentations?," and "how to present aggregate statistics when comparing the results of different algorithms?." These topics are discussed in the following five subsections.

2.3.1 Selecting a 3D Model Data Set

The first issue is to select a set of 3D models to include in the segmentation benchmark. In making our selection, we consider four main criteria: 1) the data set should cover a broad set of object categories (e.g., furniture, animals, tools, etc.); 2) models should exhibit shape variability within each category (e.g., articulated figures in different poses); 3) models should be neither too simple, nor too complex, to segment; and, 4) models should be represented in a form that most segmentation algorithms are able to take as input. This last criterion suggests that the models should be represented by manifold triangle meshes with only one (or a small number of) connected component(s).

While several data sets fit these criteria, and any of them could have been selected for our study, we chose to work with the 3D meshes from the Watertight Track of the 2007 SHREC Shape-based Retrieval Contest provided generously by Daniela Giorgi [81]. This data set contains 400 models spread evenly amongst 20 object categories, including human bodies, chairs, mechanical CAD parts, hands, fish, pliers, etc. (one example from most categories is shown in Figure 2.1). It contains two categories where the same object appears in different poses (armadillo and pliers), another seven categories where different articulated objects of the same type appear in different poses (human, ant, octopus, teddy bear, hand, bird, four leg), and three categories where objects have non-zero genus (cup, vase, chair). Several object categories are fairly easy to segment (teddy, bearing), while others are very challenging (e.g., buste). Most importantly, every object is represented by a watertight triangle mesh with a single connected component, the representation supported most commonly by currently available segmentation algorithms.

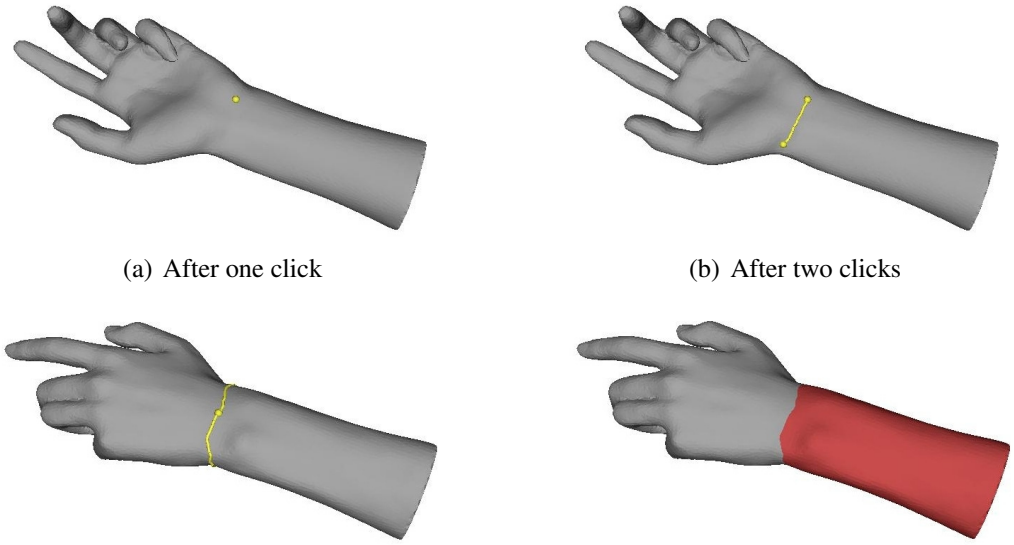
While this data set provides a good starting point for a benchmark, it also has some drawbacks. First, every model was remeshed during the process of making it watertight (e.g., the original polygonal model was voxelized, and then an isosurface was extracted). As a result, the sampling of vertices in the mesh is more uniform and noisier than is typical in

computer graphics models. Second, it contains a category called Spring, for which segmentation does not really make sense (each object is a single coil of wire), and thus we excluded it from our study. Over time, we expect the benchmark to include other data sets with different advantages and drawbacks that can be used to test different aspects of algorithms. For example, we have produced a segmented version of the Princeton Shape Benchmark [82]. However, many of its models contain mesh degeneracies, and thus it is not suitable for this study.

2.3.2 Designing an Interactive Segmentation Tool

The second issue is to design a system for collecting example segmentations from humans. Since the choice of segments/boundaries is subjective, and there are always variances among people's opinions, we needed a mechanism with which we can acquire segmentations from many different people for each polygonal model in the data set. The tool used to acquire segmentations should be easy to learn, to enable collection of data from untrained subjects; it should be quick to use (a couple of minutes per model), since thousands of examples are needed to cover the 380 models each with multiple examples; and, most importantly, it should not bias people towards particular types of segmentations – i.e., the results should reflect "what segmentation a person wants," not "what segmentation is possible with the tool."

There are many possible approaches to interactive mesh segmentation. For example, Shape Annotator [83] supports an interactive interface for combining segments generated by different automatic segmentation algorithms, and thus provides a solution that is easy to learn and quick to use. However, since the results are limited to contain only segments generated by automatic algorithms, they are highly constrained and biased towards those algorithms. Alternatively, several system are available that support intelligent scissoring of 3D meshes, using interfaces based on either strokes along cuts [50, 84] or sketches within



(a) After one click

(b) After two clicks

(c) After three clicks

(d) After making the cut

Figure 2.2: *Using the Interactive Segmentation Tool.*

segments [85]. However, then the results would be biased towards the cuts favored by the system's "intelligent" scissoring error function.

Our approach is to develop a simple interactive tool with which people can select points along segment boundaries (cuts) with a simple point and click interface, while the system connects them with shortest paths [52] (Figure 2.2). The user builds and adjusts the current cut incrementally by clicking on vertices through which the cut should traverse. For each click, the system updates the (approximately) shortest closed path through the selected vertices along edges of the mesh (shown in yellow in Figure 2.2). Vertices can be inserted into the middle of the cut or removed from it, and cuts can be defined with multiple contours (e.g., to cut off the handle of a cup). Whenever the user is satisfied with the current cut, he/she hits a key to partition the mesh, after which the faces on either side of the cut are shown in different colors. The segmentation process proceeds hierarchically as each cut makes a binary split.

This approach strikes a balance between user-control and ease-of-use. While the system provides some assistance to the user when defining cuts (it connects selected vertices with

shortest paths along edges), it is significantly less than the assistance provided by intelligent scissoring programs (which also encourage cuts to follow concave seams as defined by an error function) and significantly more than a brute-force interface where the user selects every vertex along every cut. In the former case, the system is very easy-to-use, but the results are biased. In the latter case, the system is impractically tedious to use, but gives the user absolute control over the placement of cuts. Our approach is somewhere in the middle. Specifying a desired cut often takes as few as three clicks, but can take more than ten for complicated cuts. Yet, we expect that connecting the user's input with shortest paths does not unfairly bias the results towards any particular segmentation algorithm.

2.3.3 Collecting Segmentations from Many People

The third issue is to design a procedure to acquire segmentations from multiple people for each 3D model in the data set.

The most obvious way to address this issue is to pay a group of people to come to our lab and run our interactive segmentation program under close supervision. However, this procedure is very costly, in both time and money. So, instead we recruited people to segment models through Amazon's Mechanical Turk [3]. The Mechanical Turk is an on-line platform that matches workers from around the world with paid tasks. Requesters (us) can post tasks (hits) to a web site (http://www.mturk.com/), where workers can select them from a list, follow the instructions, and submit the results on-line. The big advantage of this approach is that it taps the inexpensive and under-utilized resource of people with idle time around the world. The challenge is that we must provide instructions and quality assurance procedures that minimize the risk that acquired data is biased or poor quality.

To minimize bias, we distribute meshes to workers in random order, and we log the IP of people who access our task and combine it with the worker information from the Mechanical Turk to guarantee that each worker segments each model at most once. We also provide

very little information about how models should be segmented. Quoting from the web page, we tell the workers: "Our Motivation is to create a 3D mesh segmentation benchmark that helps researchers to better understand mesh segmentation algorithms. Our task is to use an interactive program to manually segment the 3D polygonal model into functional parts. We evaluate manually segmented 3D polygonal models by comparison to segmentations provided by experts. They should cut the surface along natural seams into functional parts with as much as detail as possible."

We also have a simple set of instructions to describe how to use the interactive segmentation tool: "An interactive tool for segmenting 3D meshes will start as soon as you accept our task. The program provides several commands (e.g., 'a') for creating and editing a path through vertices of the mesh and a command ('c') for cutting the mesh into two segments by separating along the current path. Each 'cutting path' partitions a single segment into two, and thus these simple commands can be applied in sequence to decompose a mesh into multiple segments. The segmentation result will be saved when the program exits (hit the ESCAPE key to quit the program). Full list of commands are listed in the tab called 'Tool Command Handbook.'"

Using this procedure, we received 4,365 submissions during a one month time period, of which 4,300 were accepted and 65 were rejected (for having no cuts at all). In looking at the results, we find that 25 are over-segmented in comparison to the others, and 353 have at least one cut that seems to be an outlier. However, we do not exclude this data from our analysis so as to avoid bias in the results. As shown in Figure 2.3, we accepted an average of 11 segmentations for each model. According to Amazon's Mechanical Turk statistics, the accepted submissions came from more than 80 participants, and each took an average of 3 minutes of worker time.

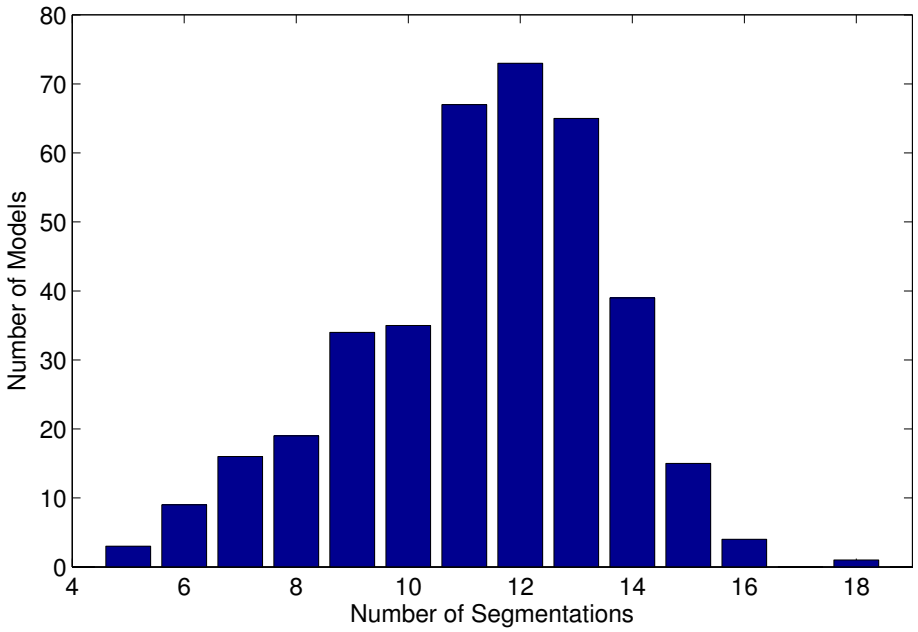


Figure 2.3: *Histogram of the number of segmentations per model.*

2.3.4 Computing Evaluation Metrics

The next design decision is to develop a set of metrics that can be used to evaluate how well computer-generated segmentations match the human-generated ones. Here, we follow prior work in computer vision, adapting four metrics that have previously been used to evaluate image segmentations. Generally speaking, the first one is boundary-based (i.e., measures how close segment boundaries are to one another), and the last three are region-based (i.e., measure the consistency of segment interiors). Among those, one explicitly considers segment correspondences in its error calculations, and another accounts for hierarchical nesting. Rather than selecting one of these metrics, we present results for all four, as each may provide different information that could be valuable in our study.

Cut Discrepancy

The first metric, *Cut Discrepancy*, sums the distances from points along the cuts in the computed segmentation to the closest cuts in the ground truth segmentation, and vice-versa. Intuitively, it is a boundary-based method that measures the distances between cuts [76].

Assuming C_1 and C_2 are sets of all points on the segment boundaries of segmentations S_1 and S_2, respectively, and $d_G(p_1, p_2)$ measures the geodesic distance between two points on a mesh, then the geodesic distance from a point $p_1 \in C_1$ to a set of cuts C_2 is defined as:

$$d_G(p_1, C_2) = \min\{d_G(p_1, p_2), \forall p_2 \in C_2\}$$

and the Directional Cut Discrepancy, $\mathrm{DCD}(S_1 \Rightarrow S_2)$, of S_1 with respect to S_2 is defined as the mean of the distribution of $d_G(p_1, C_2)$ for all points $p_1 \in C_1$:

$$\mathrm{DCD}(S_1 \Rightarrow S_2) = mean\{d_G(p_1, C_2), \forall p_1 \in C_1\}$$

We define the Cut Discrepancy, $\mathrm{CD}(S_1, S_2)$, to be the mean of the directional functions in both directions, divided by the average Euclidean distance from a point on the surface to centroid of the mesh (avgRadius) in order to ensure symmetry of the metric and to avoid effects due to scale:

$$\mathrm{CD}(S_1, S_2) = \frac{\mathrm{DCD}(S_1 \Rightarrow S_2) + \mathrm{DCD}(S_2 \Rightarrow S_1)}{\mathrm{avgRadius}}$$

The advantage of the Cut Discrepancy metric is that it provides a simple, intuitive measure of how well boundaries align. The disadvantage is that it is sensitive to segmentation granularity. In particular, it is undefined when either model has zero cuts, and it decreases to zero as more cuts are added to the ground truth segmentation.

Hamming Distance

A second metric uses *Hamming Distance* to measure the overall region-based difference between two segmentation results [76]. Given two mesh segmentation $S_1 = \{S_1^1, S_1^2, \ldots, S_1^m\}$ and $S_2 = \{S_2^1, S_2^2, \ldots, S_2^n\}$ with m and n segments, respectively, the *Directional Hamming Distance* is defined as

$$D_H(S_1 \Rightarrow S_2) = \sum_i \|S_2^i \backslash S_1^{i_t}\|$$

where "\" is the set difference operator, $\|x\|$ is a measure for set x (e.g., the size of set x, or the total area of all faces in a face set), and $i_t = \max_k \|S_2^i \cap S_1^k\|$. The general idea is to find a best corresponding segment in S_1 for each segment in S_2, and sum up the set difference. Since there are usually few segments per model, we use a brute force $O(N+mn)$ algorithm to find the correspondences.

If S_2 is regarded as the ground truth, then Directional Hamming Distance can be used to define the missing rate R_m and false alarm rate R_f as follows:

$$R_m(S_1, S_2) = \frac{D_H(S_1 \Rightarrow S_2)}{\|S\|}$$
$$R_f(S_1, S_2) = \frac{D_H(S_2 \Rightarrow S_1)}{\|S\|}$$

where $\|S\|$ is the total surface area of the polygonal model. The *Hamming Distance* is simply defined as the average of missing rate and false alarm rate:

$$\mathrm{HD}(S_1, S_2) = \frac{1}{2}(R_m(S_1, S_2) + R_f(S_1, S_2))$$

Since $R_m(S_1, S_2) = R_f(S_2, S_1)$, the Hamming Distance is symmetric. Its main advantage and disadvantage is that it relies upon finding correspondences between segments. This process provides a more meaningful evaluation metric when correspondences are "correct,"

but adds noise to the metric when they are not. It is also somewhat sensitive to differences in segmentation granularity.

Rand Index

A third metric measures the likelihood that a pair of faces are either in the same segment in two segmentations, or in different segments in both segmentations [86].

If we denote S_1 and S_2 as two segmentations, s_i^1 and s_i^2 as the segment IDs of face i in S_1 and S_2, and N as the number of faces in the polygonal mesh, $C_{ij} = 1$ iff $s_i^1 = s_j^1$, and $P_{ij} = 1$ iff $s_i^2 = s_j^2$, then we can define *Rand Index* as:

$$\mathrm{RI}(S_1, S_2) = \binom{2}{N}^{-1} \sum_{i,j,i<j} [C_{ij}P_{ij} + (1 - C_{ij})(1 - P_{ij})]$$

$C_{ij}P_{ij} = 1$ indicates that face i and j have the same id in both S_1 and S_2. $(1 - C_{ij})(1 - P_{ij}) = 1$ indicates that face i and j have different IDs in both S_1 and S_2. Thus $\mathrm{RI}(S_1, S_2)$ tells the proportion of face pairs that agree or disagree jointly on their segment group identities in segmentation S_1 and S_2. As a slight departure from standard practice, we report $1 - \mathrm{RI}(S_1, S_2)$ in our benchmark to be consistent with the our other metrics that report dissimilarities rather than similarities (the lower the number, the better the segmentation result).

The main advantage of this metric is that it models area overlaps of segments without having to find segment correspondences.

Consistency Error

The fourth metric, *Consistency Error*, tries to account for nested, hierarchical similarities and differences in segmentations [57]. Based on the theory that human's perceptual

organization imposes a hierarchical tree structure on objects, Martin et al. proposed a region-based consistency error metric that does not penalize differences in hierarchical granularity [57].

Denoting S_1 and S_2 as two segmentation results for a model, t_i as a mesh face, "\" as the set difference operator, and $\|x\|$ as a measure for set x (as in Section 2.3.4), $R(S, f_i)$ as the segment (a set of connected faces) in segmentation S that contains face f_i, and n as the number of faces in the polygonal model, the local refinement error is defined as:

$$E(S_1, S_2, f_i) = \frac{\|R(S_1, f_i) \backslash R(S_2, f_i)\|}{\|R(S_1, f_i)\|}$$

Given the refinement error for each face, two metrics are defined for the entire 3D mesh, *Global Consistency Error* (GCE) and *Local Consistency Error* (LCE), as follows:

$$GCE(S_1, S_2) = \frac{1}{n} \min\{\sum_i E(S_1, S_2, f_i), \sum_i E(S_2, S_1, f_i)\}$$
$$LCE(S_1, S_2) = \frac{1}{n} \sum_i \min\{E(S_1, S_2, f_i), E(S_2, S_1, f_i)\}$$

Both GCE and LCE are symmetric. The difference between them is that GCE forces all local refinements to be in the same direction, while LCE allows refinement in different directions in different parts of the 3D model. As a result, $GCE(S_1, S_2) \geq LCE(S_1, S_2)$. The advantage of these metrics are that they account for nested, hierarchical differences in segmentations. The disadvantage is that they tend to provide better scores when two models have different numbers of segmentations, and they can actually be misleading when either model is grossly under- or over-segmented with respect to the other. For example, the errors will always be zero if one of the meshes is not segmented at all (all faces in one segment) or if every face is in a different segment, since then one segmentation will always be a nested refinement of the other.

2.3.5 Comparing Segmentation Algorithms

The final design decision in defining the benchmark is to describe a protocol with which segmentation algorithms should be executed to produce segmentations suitable for fair comparisons.

The first issue is how to set parameters. Almost every algorithm provides tunable parameters (e.g., weights on error terms, spacing between seeds, thresholds on segment size, etc.), which sometimes affect the output segmentations dramatically. However, it would be unrealistic to search for an optimal set of parameter settings for each model; and, even if we could do that, the process would be unfair to the algorithms with fewer parameters. Hence, we utilize a single set of parameter settings for all runs of each algorithm, limiting our evaluation to the parameter settings recommended by the authors.

The one exception is for algorithms that require the target number of segments as an input parameter ($numSegments$). The problem is to select a value for this parameter that does not provide an unfair advantage or disadvantage for these algorithms with respect to algorithms that do not take such a parameter (i.e., ones that predict the number of segments automatically). One extreme solution would be to run each algorithm once for each model with the target number of segments set to some arbitrary value, say 7. Of course, this would unfairly penalize the algorithms that rely upon this parameter, since typical users would probably have a better way to select the target number of segments based on visual inspection of the model or its object category. Another extreme solution would be to run each algorithm repeatedly, once for each human-generated segmentation H to which it is being compared, each time setting *numSegments* to match the number of segments in H exactly. Of course, this would unfairly benefit the algorithms that take *numSegments* as a parameter – they would be given an oracle to predict the number of segments created by every individual person. Our solution is somewhere in the middle. We imagine that a typical user can look at a model and guess the number of target segments approximately. So, mod-

eling a scenario where a user looks at a model and then runs the segmentation algorithm, we choose a separate value for *numSegments* for each model, setting it to be the mode of the number of segments appearing in segmentations created by people for that model in the benchmark data set (this design decision is evaluated in Section 2.4.6).

The second issue is remeshing. Some segmentation algorithms change the topology of the mesh. Yet, most of our evaluation metrics rely upon meshes having the same topology. To overcome this problem, we provide a step where the segment labels of the modified mesh are mapped back to the input mesh (this mapping employs an area-weighted voting scheme to assign a face to a segment if it is split into two in the output, and it eliminates segments with fewer than 3 faces or less than 0.01% of the total surface area). This process avoids penalizing algorithms for changing the topology of the mesh.

The final issue we must address is how to aggregate evaluation metrics over multiple segmentations for the same model and multiple models for the same object category. For now, we simply report averages, both per category and over the entire data set for each algorithm (averages are computed first within each model, then those results are averaged within each object category, and then those results are averaged across the entire database to avoid biasing the results towards models and/or categories with more segmentations in the benchmark data set). However, more sophisticated aggregate statistics could be computed in the future.

2.4 Results

In order to investigate the utility of the proposed benchmark for evaluating segmentations, we performed six experiments. The first asks "How consistently do different people segment the same object?" The second two study how effectively the benchmark can be used to compare the performance of different segmentation algorithms, asking questions like: "Which algorithms produce outputs that most closely mimic human segmentations?," and

"Does any algorithm provide the best results for every type of object?" The next two investigate properties of human- and computer-generated segmentations, asking "which geometric properties are consistent in all segmentations by people" and "how do the properties of automatic segmentations compare to humans'?." The final study evaluates how our method for selecting the number of target segments for each model affects the comparison of algorithms.

While the results of these studies may be interesting in their own right, our main goal is to investigate whether the data set and evaluation procedures described in the preceding sections are useful for understanding similarities and differences between segmentations. It is our intention that other analyses and comparisons will be performed in the future as the benchmark grows as a public-domain resource for the research community.

2.4.1 Consistency of Human-Generated Segmentations

The benchmark provides segmentations made manually by multiple people for each of 380 models from a variety of different object categories, and thus it provides an opportunity to study how consistently different people segment the same objects. This question is not only important for its obvious implications in perceptual psychology, but also for its implications on our benchmark. Implicit in our choice to use human-generated segmentations as "ground truth" for our benchmark are the assumptions that people tend to segment objects in the same ways and that automatic algorithms ought to mimic them. We can validate the first of these assumptions with visual and statistical analysis of the benchmark data set.

Figures 2.1 and 2.4 show visualizations of the segmentations collected for a sampling of objects in the benchmark data set. These images show composites of cuts made by different people, and so darker lines represent edges where people placed segmentation boundaries more consistently. For example, in the image of the human body on the left of Figure 2.4, it can be seen that people cut very consistently on edges that separate the arms from the

body, but less consistently on edges that separate the chest from the stomach. In general, by visual inspection, it seems that people select the same types of functional parts, possibly cutting on edges that are parallel and slightly offset from one another (e.g., at the junctions of the fingers and the palm of the hands), and possibly decomposing objects with different levels of granularity (e.g., only some people cut at the knuckles in the two images of hands in Figure 2.4). Although a few exceptions exist (e.g., in the eyeglasses shown in the top-left image of Figure 2.4, people agree that the two lenses should be separated, but they disagree whether the bridge between two lenses should be segmented as an independent part), we conclude that people are remarkably consistent in their decompositions into parts.

This conclusion can be validated quantitatively using the evaluation metrics described in Section 2.3.4. For each segmentation in the benchmark, we can "hold it out" and then ask how well it matches the segmentations made by other people for the same model according to the four evaluation metrics. The results of this analysis are shown in Figure 2.5 – the bar labeled "Human" on the left side of each of the four plots shows the averages of the human segmentations evaluated with respect to all the other human segmentations. While it is difficult to make conclusions from the absolute heights of the bars in these plots, the bars labeled "Human" are clearly lower (better) than the corresponding bars for computer-generated segmentations, suggesting that people are more consistent with each other than algorithms are with people.

2.4.2 Comparison of Segmentation Algorithms

The main purpose of the benchmark is to evaluate and compare the results of automatic segmentation algorithms. To test its utility for this task, we followed the procedure described in Section 2.3.5 to compare seven algorithms recently published in the literature (listed below). In every case, except one (K-Means), the source code and/or executable was provided directly by the original inventors, and so each algorithm was tested with its

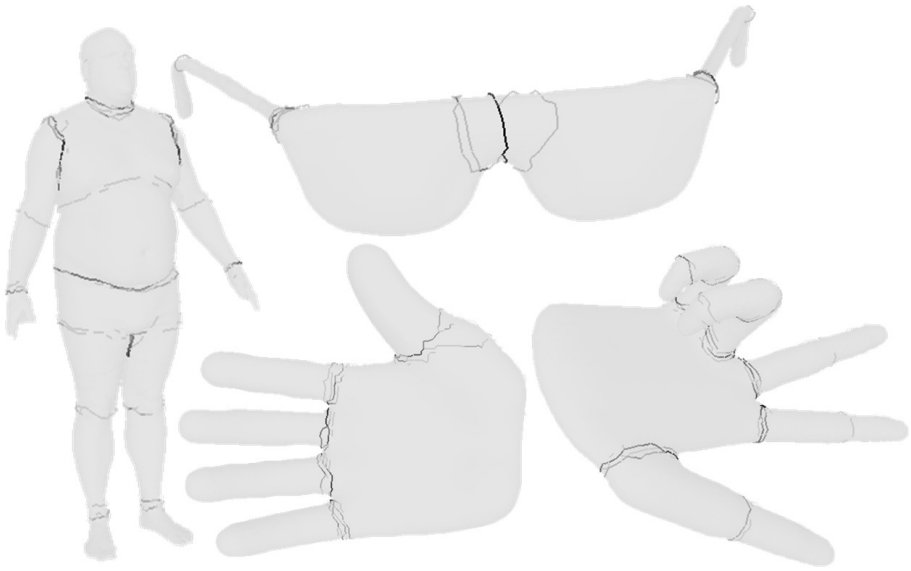
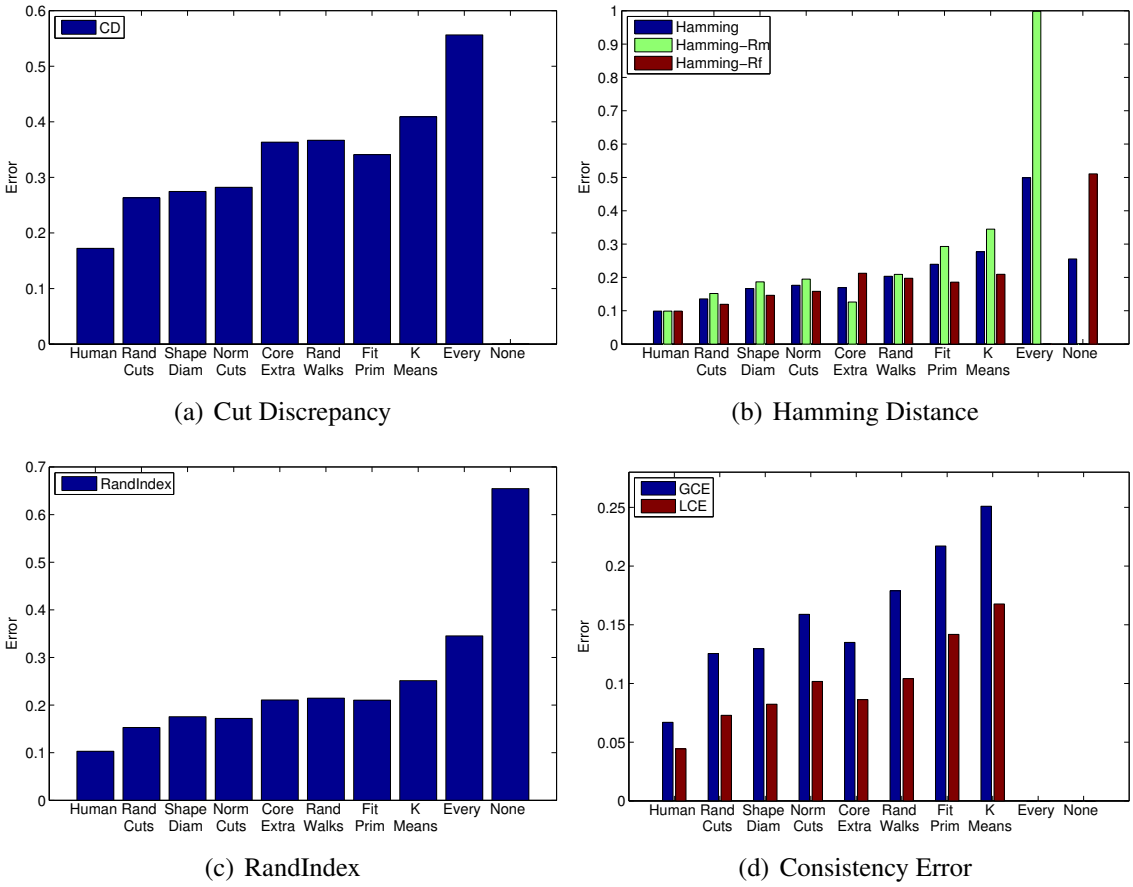


Figure 2.4: *Composited images of segmentation boundaries selected by multiple people for four example models (darker lines appear where a larger fraction of people placed a segmentation boundary).*

original implementation and with its default parameters. The following paragraphs provide a brief description of each algorithm, but please refer to the cited papers for details:

- **K-Means:** Shlafman et al. [60] describe an algorithm based on K-means clustering of faces. Given a user-specified number of segments, k, the algorithm first selects a set of k seed faces to represent clusters by continually selecting the further face from any previously selected. Then, it iterates between: 1) assigning all faces to the cluster with the closest representative seed, and 2) adjusting the seed of each cluster to lie at the center of the faces assigned to it. This iteration continues until the assignment of faces to clusters converges. In our implementation, which differs from the original by Shlafman et al., we compute distances on the dual graph of the mesh with a penalty related to the dihedral angle of each traversed edge using the method in [50].

- **Random walks:** Lai et al. [65] describe a procedure that proceeds in two phases. During the first phase, an over-segmentation is computed by assigning each face F to the segmented associated with the seed face that has highest probability of reaching F by a random walk on the dual graph of the mesh. During the second phase, segments



(a) Cut Discrepancy

(b) Hamming Distance

(c) RandIndex

(d) Consistency Error

Figure 2.5: *Comparison of segmentation algorithms with four evaluation metrics.*

are merged hierarchically in an order based on the relative lengths of the intersections and total perimeters of adjacent segments. The hierarchical clustering terminates when a user-specified number of segments has been reached.

- **Fitting Primitives:** Attene et al. [64] propose a hierarchical clustering algorithm based on fitting primitives. The algorithm starts with every face in a separate segment. At each iteration, the best fitting geometric primitive (plane, cylinder, or sphere) is computed to approximate the faces in every pair of adjacent segments, and the best fitting pair is chosen for merger. The algorithm proceeds bottom-up, merging segments until a user-specified number of segments has been reached.

- **Normalized cuts:** Golovinskiy et al. [61] describe a hierarchical clustering algorithm in which every face of the mesh starts in its own segment, and segments are

hierarchically merged in an order determined by the area-normalized cut cost: the sum of each segment's perimeter (weighed by concavity) divided by its area. The clustering terminates when a user-specified number of segments has been reached. This algorithm encourages segments to have small boundaries along concave seams while maintaining segments with roughly similar areas.

- **Randomized cuts:** Golovinskiy et al. [61] also propose a hierarchical decomposition procedure that uses a set of randomized minimum cuts to guide placement of segmentation boundaries. They first decimate the mesh (to 2,000 triangles in this study), and then proceed top-down hierarchically, starting with all faces in a single segment and iteratively making binary splits. For each split, they compute a set of randomized cuts for each segment, and then they identify for each segment which cut is most consistent with others in the randomized set. Amongst this set of candidate cuts, they choose the one that results in the minimal normalized cut cost. The algorithm terminates when a user-specified number of segments has been reached.

- **Core extraction:** Katz et al. [66] propose a hierarchical decomposition algorithm that performs four main steps at each stage: transformation of the mesh vertices into a pose insensitive representation using multidimensional scaling, extraction of prominent feature points, extraction of core components using spherical mirroring, and refinement of boundaries to follow the natural seams of the mesh. The algorithm partitions segments hierarchically, stopping automatically when the current segment S^i has no feature points or when the fraction of vertices contained in the convex hull is above a threshold.

- **Shape Diameter Function:** Shapira et al. [87] describe an algorithm based on the "Shape Diameter Function" (SDF), a measure of the diameter of an object's volume in the neighborhood of a point on the surface. The SDF is computed for the centroid of every face, and then the segmentation proceeds in two steps. First, a Gaussian

Mixture Model is used to fit k Gaussians to the histogram of all SDF values in order to produce a vector of length k for each face indicating its probability to be assigned to each of the SDF clusters. Second, the alpha expansion graph-cut algorithm is used to refine the segmentation to minimize an energy function that combines the probabilistic vectors from step one along with boundary smoothness and concaveness. The algorithm proceeds hierarchically for a given number of "partitioning candidates," which determines the output number of segments. We follow the authors' advice, setting the number of partitioning candidates to 5, and let the algorithm determine the number of segments automatically.

Figure 2.5 shows evaluations of these seven algorithms according to the proposed benchmark. Each of the four bar charts shows a different evaluation metric computed for all seven algorithms and averaged across the entire data set as described in Section 2.3.5. Added to each chart is a bar representing the human-generated segmentations evaluated with respect to one other (labeled "Human," on the left) and two bars representing trivial segmentation algorithms ("None" = all faces in one segment, and "Every"= every face in a separate segment). These three bars are added to the plots to provide sanity checks and to establish a working range for the values of each evaluation metric. In all cases, lower bars represent better results.

Segmentation Algorithm	Avg Compute Time (s)	Rand Index
Human	-	0.103
Randomized Cuts	83.8	0.152
Shape Diameter	8.9	0.175
Normalized Cuts	49.4	0.172
Core Extraction	19.5	0.210
Random Walks	1.4	0.214
Fitting Primitives	4.6	0.210
K-Means	2.5	0.251
None	0	0.345
Every	0	0.654

Table 2.1: *Analysis of trade-off between compute time and segmentation quality (compute time is in seconds measured on a 2Ghz PC).*

From this data, we see that all four evaluation metrics are remarkably consistent with one another. Although one of the metrics focuses on boundary errors (Cut Discrepancy), and the other three focus on region dissimilarities (Hamming Distance, Rand Index, and Consistency Error), all variants of all four metrics suggest almost the same relative performance of the seven algorithms.

Examining the average timing and performance results in Table 2.1, we see that the set of algorithms that perform best overall according to the Rand Index metric (Randomized cuts, Normalized cuts, Shape Diameter Function, and Core Extraction) are the ones that take the most time to compute.[1] This result is not surprising. They are the algorithms that consider both boundary and region properties of segments and utilize non-local shape properties to guide segmentation. For example, the algorithms based on Core Extraction and the Shape Diameter Function both consider volumetric properties of the object in the vicinity of a point, and the Randomized Cut and Normalized Cut algorithms both evaluate expensive objective functions that take into account large-scale properties of segment boundaries and areas. These four algorithms should probably be used for off-line computations, while the others are nearly fast enough for interactive applications.

[1] It is important to note that the Shape Diameter Function and Core Extraction algorithms automatically predict the number of segments to output, while the other algorithms require *numSegments* as an input parameter. So, these two algorithms are solving a more difficult problem than the others. The reader should keep this in mind when comparing results. This issue is investigated further in Section 2.4.6

2.4.3 Comparisons Within Object Categories

The results shown in the previous subsection provide only averages over all models in the benchmark, and thus paint broad strokes. Perhaps more important than ranking algorithmic performance overall is to understand for which types of objects each algorithm performs best. Then, users can leverage the results of the benchmark to select the algorithm most appropriate for specific types of objects.

Table 2.2 shows a comparison of the performance of the seven algorithms for each object category. The entries of this table contain the rank of the algorithm according the Rand Index evaluation metric averaged over models of each object category: 1 is the best (shown in red) and 7 is the worst. From these results, we see that no one segmentation algorithm is best for every category. Interestingly, it seems that some algorithms do not necessarily work best (relative to others) on the types of objects for which they were designed. For example, the method designed to fit geometric primitives is among the best for Humans, but not for the Mech and Bearing classes, which are CAD Models comprised of geometric primitives. These results suggest that either people do not segment objects in the expected way, or that there are other cues that reveal the part structures of these objects (e.g., concave seams).

2.4.4 Properties of Human-Generated Segmentations

The human-generated segmentations acquired for the benchmark are useful not only for comparisons, but also for studying how people decompose objects into parts. Several psychologists have put forth theories to explain how people perceive objects as collections of parts (e.g., the Recognition by Components theory of Biederman [17], the Minima Rule hypothesis of Hoffman and Richards [18, 19], etc.), and several algorithms incorporate assumptions about what type of segmentations people prefer (e.g., cuts along concave

Object Category	Rand Cuts	Shape Diam	Norm Cuts	Core Extra	Rand Walks	Fit Prim	K-Median
Human	1	5	2	7	6	3	4
Cup	1	5	2	3	4	6	7
Glasses	1	4	2	6	7	5	3
Airplane	2	1	4	7	6	3	5
Ant	2	1	3	4	5	6	7
Chair	4	2	1	5	3	6	7
Octopus	4	1	3	2	5	7	6
Table	7	4	1	5	2	3	6
Teddy	1	2	4	3	5	6	7
Hand	1	7	3	4	5	6	2
Plier	2	7	4	1	5	3	6
Fish	3	1	5	2	4	7	6
Bird	1	2	4	3	7	6	5
Armadillo	3	1	5	7	4	2	6
Bust	1	3	6	5	2	4	7
Mech	4	3	1	6	2	5	7
Bearing	2	1	3	7	5	4	6
Vase	1	4	3	2	5	6	7
FourLeg	2	1	6	4	7	3	5
Overall	1	3	2	5	6	4	7

Table 2.2: *Comparison of segmentation algorithms for each object category. Entries represent the rank of the algorithm according to the Rand Index evaluation metric (1 is the best, and 7 is the worst).*

seams [20], segments of nearly equal area [60], convex parts [88], etc.). We conjecture that our benchmark data set could be used to test these theories and assumptions.

As an example, the *left column* of Figure 2.7 shows five histograms of properties collected from the 4,300 human-generated segmentations of the benchmark set: a) minimum curvature at vertices on (blue) and off (green) segmentation boundaries; b) convexity, as measured by the average distance from a segment's surface to its convex hull [89] normalized by the average radius of the model; c) the length of a cut between segments normalized by the average of all cuts for its mesh; d) the area of a segment's surface normalized by the average segment area for its mesh; and e) the number of parts in each segmentation. We also have collected statistics for dihedral angles, maximum curvature, Gaussian curvature, mean curvature, minimum curvature derivative, and compactness [89], but do not show them in Figure 2.7 due to space limitations.

These statistics seem to support many of the shape segmentation theories and assumptions that are prevalent in the literature. For example, segment boundaries are indeed more likely along concavities in our data set – i.e., at vertices with negative minimum curvature (Figure 2.7a). Also, segment surfaces tend to be convex – i.e., having relatively small distances between the surface of a segment and its convex hull (Figure 2.7b).

However, some of the results are a bit surprising. For example, it seems that cuts between segments within the same mesh tend to have approximately the same lengths – i.e., the Normalized Cut Length (length of a cut divided by the average for the mesh) is often close to one (Figure 2.7c). It seems that this property (or related ones, like symmetry) could be included in a segmentation error function, but we are not aware of any prior segmentation algorithm that does so. In contrast, it seems that not all segments have the same area – i.e., the Normalized Segment Area (area of a segment divided by the average for the mesh) is usually not equal to one (Figure 2.7d). Rather, it seems that each mesh has a small number of large segments (where the Normalized Segment Area is much greater than 1) and a larger number of small segments (where the Normalized Segment Area is less than 1), probably corresponding to the "body" and "appendages" of an object. This observation suggests that algorithms that explicitly aim for equal segment areas (e.g., K-Means) are less likely to mimic what people do. Of course, these statistics and observations are specific to our data set, but we conjecture that they are representative because the benchmark includes a wide variety of object types.

2.4.5 Properties of Computer-Generated Segmentations

We can also study properties of computer-generated segmentations to better understand how algorithms decompose meshes and how they compare to humans'. Towards this end, we show histograms of the five properties discussed in the previous section for segmentations computed by the Normalized Cuts, Randomized Cuts, Shape Diameter Function,

Core Extraction, and K-Means algorithms in Figure 2.7 (from left to right starting in the second column). By comparing these histograms to the ones in the far left column (labeled "Human"), we aim to understand not only which computer-generated segmentations most closely resemble Humans', but also how are they similar and how are they different.

For example, looking at the histograms in the rightmost column of Figure 2.7 ("K-Means") and comparing them to the ones in the leftmost column ("Human"), we see that very few of the histograms are similar. It seems that the K-Means algorithm does *not* preferentially cut along vertices with negative minimum curvature (there is little separation between the blue and green lines in the top right histogram), but does preferentially decompose each mesh into segments with nearly equal areas (there is a large peak at '1' in the rightmost histogram of the bottom row). These properties do not match the Humans', which may explain why the evaluation metrics for K-Means were the poorest in Figure 2.5.

In contrast, comparing the first and second columns, we see that the histograms representing Normalized Cut segmentations (second column) very closely resemble those of the Humans' (first column) – i.e., they generally have the same shapes, maxima, and inflection points. These results corroborate the evaluation metrics presented in Section 2.4.2 – i.e., since many properties of the Normalized Cut segmentations match the Humans', it is not surprising that they also produce the best evaluation metrics. Generally speaking, the histograms further to the left in Figure 2.7 resemble the Humans' better than those that are further to the right, and similarly are ranked better by most evaluation metrics.

2.4.6 Sensitivity to Number of Segments

As a final experiment, we investigate how sensitive our evaluations are to the number of segments produced by the algorithms in our study. As discussed in Section 2.3.5, several algorithms (Randomized Cuts, Normalized Cuts, Fitting Primitives, Random Walks, and K-Means) require a parameter that determines the number of target segments produced by

the algorithm (*numSegments*). We set that parameter based on the most frequent number of segments found in human-generated segmentations of the same model in the benchmark (the mode for each model). This choice was made to ensure that the algorithms are being tested on typical input – i.e., it makes sense to evaluate how well an algorithm segments a coffee mug into two parts, but probably not seventeen. On the other hand, some algorithms do not take *numSegments* as a parameter and instead determine the number of segments to output automatically, and thus are at a disadvantage if the other algorithms are given a good estimate for the number of parts (note the differences for those two algorithms in the number of parts per model in the bottom row of Figure 2.7). To investigate the practical implications of this choice, we tested six alternative ways to set *numSegments* and study their effects on the relative performance of all seven segmentation algorithms:

1. **By Dataset:** *numSegments* could be set to the same number for all runs of every algorithm – i.e., to the average number of segments found amongst all examples in the benchmark data set (which is 7).

2. **By Category:** *numSegments* could be set separately for every *object category* – i.e., to the mode of the number of segments observed for that category in the benchmark data set.

3. **By Model:** *numSegments* could be set separately for every *model* – i.e., to the mode of the number of segments observed for that model in the benchmark data set.

4. **By Segmentation:** *numSegments* could be set separately for each human-generated segmentation. That is, every algorithm could be run once for every unique number of segments found in the benchmark, and then evaluations could be performed only between segmentations with exactly the same numbers of segments. This approach implies that the algorithm has an oracle that predicts exactly how many segments each person will produce, even if different people produce different numbers of seg-

ments – and thus provides a lower-bound on the evaluation metrics with respect to different selections of *numSegments*.

5. **By Shape Diameter Function:** *numSegments* could be set separately for every model according to the number of segments predicted by the Shape Diameter Function (SDF) algorithm. This choice allows investigation of how the relative performance of the SDF algorithm is affected by its choice in the number of segments – i.e., it puts the other algorithms on nearly equal footing.

6. **By Core Extraction:** *numSegments* could be set separately for every model according to the number of segments predicted by the Core Extraction algorithm. This choice allows investigation of how the relative performance of the Core Extraction algorithm is affected by its choice in the number of segments.

We implemented all six alternatives and ran an experiment to test how they affect the benchmark results. A plot of Rand Index for this study is shown in Figure 2.6. This plot is similar to the one in the bottom-left of Figure 2.5 (in fact, the dark blue bars are the same), except that each algorithm that takes *numSegments* as an input parameter is represented by six bars in this plot, one for each alternative for setting *numSegments*.

Looking at this plot, we observe that the comparisons of algorithms that take *numSegments* as input are largely insensitive to the method chosen to set *numSegments*. The main difference is that the algorithms that do not take *numSegments* appear relatively better or worse depending on how intelligently *numSegments* is set for the other algorithms. On the one hand, if the number of segments is chosen "By Dataset" (all models are decomposed into 7 segments), then the Shape Diameter Function algorithm (which chooses the number of segments adaptively) performs best. On the other hand, if the number of segments is chosen "By Segmentation" (green bars), "By Model" (dark blue bars), or "By Category" (yellow bars), then the Normalized Cuts and Randomized Cuts algorithms perform best.

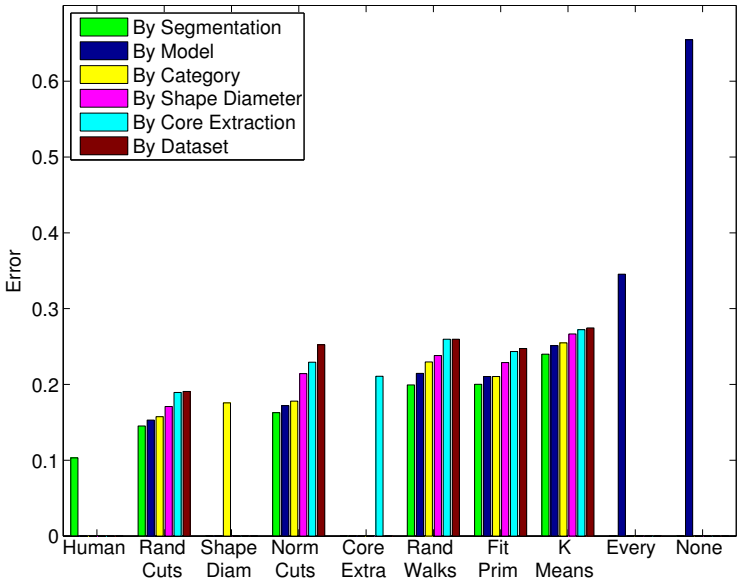


Figure 2.6: *Plot of Rand Index for six different methods of setting the target number of segments for algorithms that take it as input.*

Otherwise, we observe that the relative performance of algorithms remains remarkably consistent across different ways of controlling the number of segments.

Interestingly, when *numSegments* is chosen to match the number of segments predicted by the Shape Diameter Function algorithm (magenta bars), the performance of that algorithm is almost equal to the Normalized Cuts'. The same applies for the Core Extraction algorithm when *numSegments* is chosen to match its prediction (cyan bars). This result suggests that the performance difference between those two algorithms and the others may largely be due to differences in the numbers of segments generated, as opposed to differences in the placement of segments. However, this conjecture must be verified with an implementation for those two algorithms that allows setting *numSegments* (no such parameter is available currently).

Even more interesting is that fact that there is a significant difference between the performance of the best algorithms and the performance of the humans, even when the algorithms are given an oracle to set *numSegments* to match each human's exactly ("By Segmentation") – i.e., the green bar for "Human" on the far left is much smaller than the green bar

for any algorithm. This is in contrast to the relatively small differences in performance between setting *numSegments* "By Segmentation" vs. "By Model" vs. "By Category." This result suggests that the main differences between humans and algorithms are not due to differences in the number of segments produced by the algorithms.

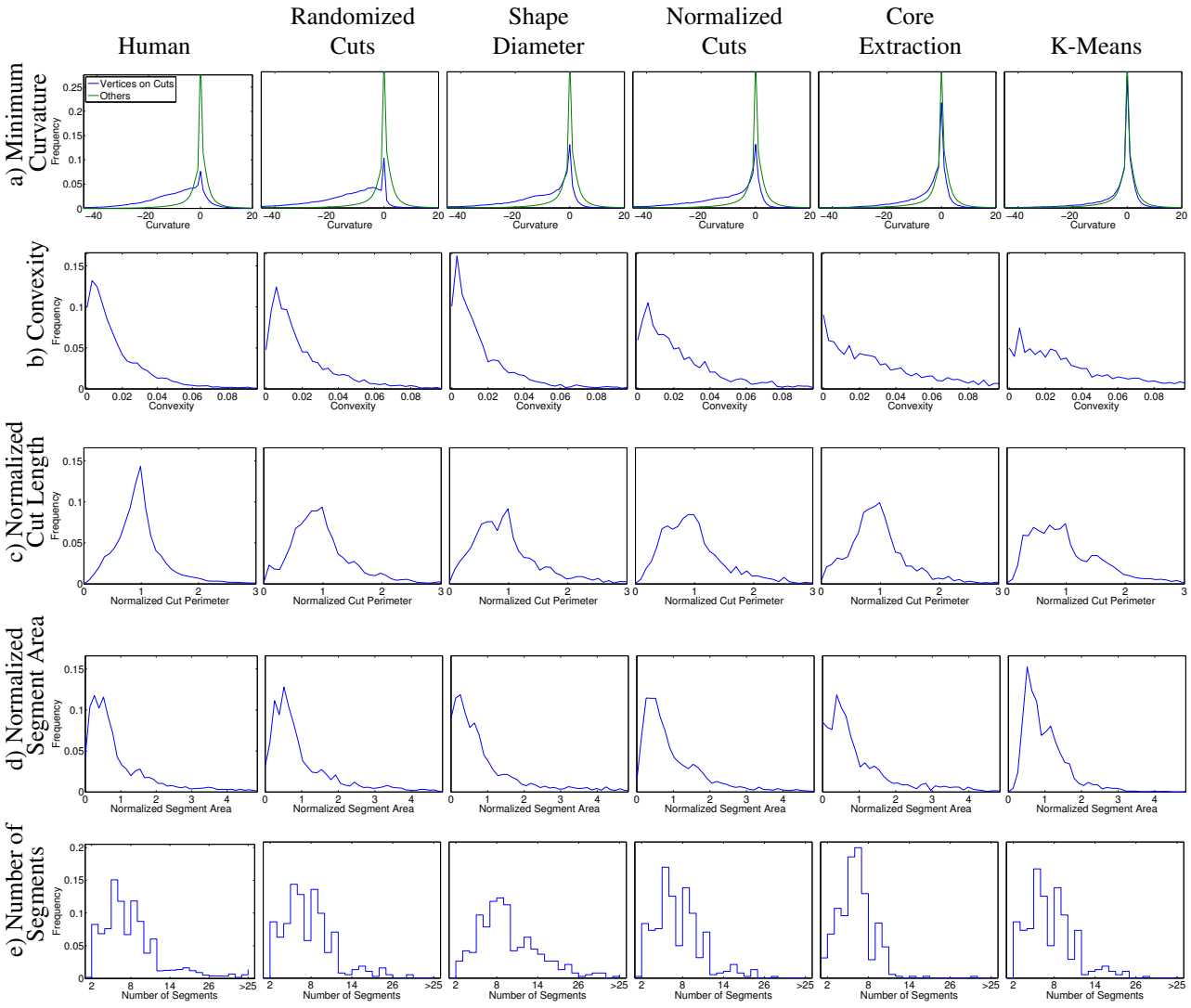


Figure 2.7: *Properties of human-generated segmentations (left column) compared with computer-generated segmentations (from second column to sixth) for five algorithms (properties of Fitting Primitives and Random Walks not shown due to space limitations).*

2.5 Conclusion

This chapter describes a benchmark for mesh segmentation. The benchmark contains 4,300 manual segmentations over 380 polygonal models of 19 object categories. Initial tests have been performed to study the consistency of human-generated segmentations and to compare seven different automatic segmentation algorithms recently proposed in the literature. The results suggest that segmentations generated by different people are indeed quite similar to one another and that segments boundaries do indeed tend to lie on concave seams and valleys, as predicted by the minima rule. We find that the benchmark data set is able to differentiate algorithms, but we do not find one algorithm that is best for all object categories. Generally speaking, it seems that algorithms that utilize non-local shape properties tend to out-perform the others. Since the benchmark data set and source code is publicly available (http://segeval.cs.princeton.edu/), we expect that these results will be amended and augmented by researchers as more data becomes available, more tests are performed, and more algorithms are developed in the future.

Chapter 3

Schelling Points

3.1 Introduction

Detection of "salient" feature points on 3D surfaces is also a fundamental problem in computer graphics with many applications in shape analysis and related fields, including object recognition [90], shape matching [91], shape-based retrieval [92], metamorphosis [93], cross-parameterization [94], texture mapping [95], deformation transfer [96], shape approximation [26], viewpoint selection [26], symmetry detection [97], and part-based segmentation [66, 98].

Although many definitions have been proposed for what constitutes "salient" feature points in the computer graphics literature (e.g., maxima of average geodesic distance, maxima of Gauss curvature, maxima of mean curvature differences at increasing scales, etc.), none of them captures what people perceive as salience in the social/psychological perspective, as defined in the Oxford English Dictionary: "the quality or fact of being more prominent in a person's awareness or in his memory of past experience" [99]. It is this definition that captures the semantic essence of a stable feature point, and therefore one that we believe is useful for applications in computer graphics.

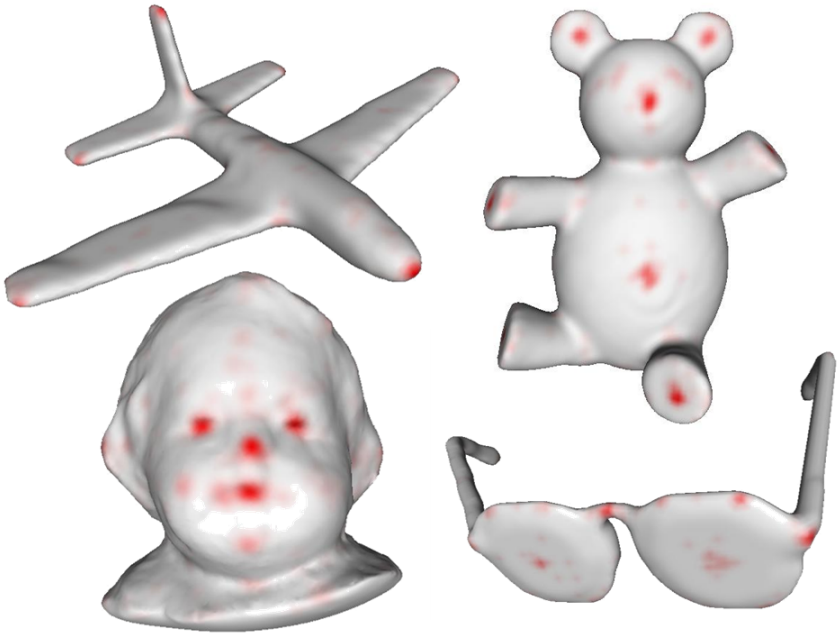


Figure 3.1: *Schelling points (red). Positions on a surface selected consistently by many people when trying to match each other without communication.*

The goal of this chapter is to develop a model of salience on 3D surface meshes based on this social/psychological definition. To achieve this goal, we leverage the concept of focal points as introduced by Schelling in his seminal paper on pure coordination games [100]. In game theory, "a focal point (also called a Schelling point) is a solution that people will tend to use in the absence of communication, because it seems natural, special or relevant to them" [101]. To discover focal points, Schelling performed user studies in which he asked people to make selections that they expect will match other people's selections. Since the people were not allowed to communicate in any way, they usually picked the most conspicuous point, often using semantic information not provided in the input. For example, if asked to select a time and place in New York City to meet someone without any prior communication, people tended to choose Grand Central Terminal at noon. This choice is clearly the result of prior semantic knowledge that makes that choice more prominent in a person's awareness or in his memory of past experience (i.e., social/psychological salience).

Inspired by Schelling's work, we designed a user study that asks people to select points on 3D surface meshes that they expect will be selected by other people and used the acquired data to build a model of mesh saliency. Our primary research contribution is the analysis of the collected data. We find that: 1) Schelling point sets are usually highly symmetric, and 2) local curvature properties (e.g., absolute value of minimum curvature) are most helpful for predicting obvious Schelling point features, but 3) global properties (e.g., symmetry and segment centeredness) are required to explain other features. Our secondary research contributions are the methods used for acquiring, analyzing, and predicting Schelling points on 3D meshes. They include: 1) the design of a user study, 2) the analysis of consistency and properties of Schelling points, 3) a learned model for predicting the distribution of Schelling points, and 4) an algorithm that uses the learned model to extract Schelling points on new meshes.

3.2 Related Work

Detection and matching of salient feature points are classical problems in computer vision, geometric modeling, computer-aided design, computer graphics, and several other fields [102]. In this section, we cover the most related work, focusing on methods designed for saliency estimation at a point.

Feature Point Detection: There has been many recent work on automatically extracting feature points on 3D meshes. Example methods include local maxima of average geodesic distance (AGD) to other points on the surface [98, 95, 91], local maxima of differences of Gaussians at multiple scales [103], local maxima of Gaussian curvature [104], properties of the Heat Kernel Signature [105], scale-space analysis of mean curvature flow [106], other multiscale analysis [107, 108, 109, 110, 111], points on the convex hull after MDS embedding [66], leaf nodes of a curve skeleton extraction [112], points with unlikely

local shape descriptors [113, 90], and points with local shape descriptors distinctive of an object class [114]. Most methods are based on differential properties of the surface (e.g., [103]), while others are based on global properties [91] and/or shape descriptor statistics (e.g., [90]). Previous work has analyzed which of these feature points is most stable under various models of perturbation [115], but none as studied how they relate to semantic salience in the Schelling sense.

Saliency Estimation: In related work, methods have been proposed to define continuous measures of "saliency" across a surface for mesh processing applications. Motivated by perceptual criteria [19], Lee et al. [26] used a center-surround filter of curvature across multiple scales to select salient regions for mesh simplification and viewpoint selection. Gal et al. [116] computed the saliency of a region based on its size relative to the whole object, its curvature, the variance of curvature, and the number of curvature changes within the region. Shilane et al. [114] defined the "distinction" of a surface region based on its similarity to objects within the same class and difference to objects in other classes. This chapter studies how these types of saliency measure relate to human-selected focal points and provides a new saliency measure learned from examples.

Saliency Evaluation and Comparison: Kim et al. [25] studied how eye fixations relate to mesh saliency [26], and other studies have evaluated and compared feature point detection algorithms for images in computer vision [117, 118, 24, 119, 120, 121]. For example, Schmid et al. [119] and Sebe et al. [122] compare interest point detectors on the basis of stability, repeatability, and information content. Privitera et al. [24] compare region of interest algorithms based on how well they match eye fixations measured with an eye tracker. Huang et al. [123] compare algorithms with respect to points collected in a game (Photoshoot) that asks people to select points on images that they expect will match a partner's selection, much like the ESP game proposed by Ahn et al. [124]. This work is similar to ours in that it also asks people to match point selections. However, the methodology of

that study can produce bias in selected points due to training effects of immediate feedback (it is not a pure coordination game), it considers only 2D points in natural images, it does not provide any analysis of how image properties correlate with selected points, it does not suggest new properties correlated with the selected points (e.g., symmetry, segmentations, etc.), and it does not provide a predictor of points learned from the collected examples.

Perceptual Psychology: There have been many studies in perceptual psychology to understand how people assign importance to regions of images. Most have been based on visual attention [22, 125, 126, 127, 128, 129]. For example, Koch et al. [22] proposed a model that salient points would be ones that are different from their surroundings. Other perceptual psychology studies have studied which points are most important for approximation of a contour. For example, Attneave [23] showed 80 subjects a series of 16 shapes drawn as 2D contours and then asked them to select "a pattern of 10 dots which would resemble the shape as closely as possible." His main finding was that most people select points where the "contour is most different from a straight line" – i.e., where the curvature has large magnitude. While these studies are related to ours, we aim to discover which points have semantic salience in 3D meshes.

Our work follows the line of research to understand how people perform tasks of interest in computer graphics [12, 11], focusing on feature point detection. It is novel not only because it considers a new question: "where do people select feature points?," but also because it asks the question in a different way: "where do other people select feature points?" and because it includes new algorithms for analyzing and extracting feature points based on the collected data.

3.3 Approach

The goal of this chapter is to develop a model of semantic salience for 3D surfaces. Our general approach is to gather a large collection of feature points from people and then to study what geometric properties distinguish them from others.

Although this general idea may sound straight-forward, it is surprisingly difficult to design a study to collect useful data on salience. We did not want to simply ask people to "please click on important points" because different people might have different ideas of what it means to be important (e.g., functional, structural, social, visual, etc.), and it would be difficult to ask the question in a way that reveals people's intentions without leading them to an answer.

During a pilot study, we tried an approach based on Attneave [23], where we asked people to select points on a 3D surface from which another person could recognize the object class by just viewing those points. This approach was a resounding failure. We found that most people selected points only on a 2D silhouette curve of the surface as seen from a single canonical view (e.g., the outline of a fish as seen from the side), which does not seem to match a notion of semantic salience that is useful for 3D applications.

Ultimately, we arrived at an approach based on Schelling's focal points: we ask people to select points they think will be selected by others. The concept of focal points was introduced in pure coordination game theory in the 1960's [100] (page 57) – Schelling found that people asked to make selections that match other people's selections amongst seemingly equivalent distinct options (segregated Nash equilibrium) often make the same choices without any communication or feedback. For example, if people were asked to choose "heads" or "tails" with no other information besides the goal of matching as many other people as possible, 86% chose heads [100]. If asked to match other people's selection on a map, most people agreed on just a few points (e.g., prominent intersections). It was

conjectured that the commonly selected focal points (later called Schelling points) arise from a strategy or outcome with properties of "prominence or conspicuousness" [100].

A decade later, Lewis used Schelling's ideas to introduce the term "salience," which he defined, first, as the property of an outcome of "standing out from the rest by its uniqueness in some conspicuous respect" and, second, as "being unique in some way everyone will notice, expect the others to notice" [130] (page 35). He used formal game models to characterize coordination strategies, hypothesizing that people use "common knowledge" to select amongst distinct Nash equilibrium. It is this "common knowledge" about feature points on 3D surfaces that we aim to capture and leverage in our work.

Inspired by the ideas of Schelling and Lewis, we have designed a method to study semantic salience on 3D meshes. Specifically, we first acquire a large number of Schelling points by asking people to select points on 3D surface meshes that they expect will be selected by other people (Section 3.4). We then analyze properties of the collected point sets, asking questions like: "how consistent are Schelling points selected on different meshes within the same object class?" and "how are the locations of Schelling points associated with geometric properties of the surface?" (Section 3.5). Next, we train a regression model that predicts the likelihood that a point on a surface is a Schelling point based on its surface properties (Section 3.6). Finally, we provide an algorithm to predict a set of Schelling points for new meshes (Section 3.7).

3.4 Study Design

The first and most difficult issue faced in our investigation is how to design a study to acquire Schelling points from many people for many types of 3D surfaces. We would like to collect enough data to analyze how Schelling points relate to surface properties across a wide variety of meshes and to train a predictive model that can be used to estimate Schelling

points for new 3D meshes. Of course, this is difficult because it requires recruiting and supervising many (possibly hundreds or thousands of) human subjects in a user study.

To address this issue, we performed our study on-line. Following the approach of Chen et al. [11] and Cole et al. [131], we recruited subjects for our study through Amazon's Mechanical Turk (AMT) [3], an on-line platform that matches people willing to work with paying tasks. Alternatively, we could have designed an on-line game to acquire input (as in [123, 124]), but that approach would have required attracting a player population, which is beyond the scope of this chapter. Since tasks on the AMT are typically short in duration (a minute or two), inexpensive (around 10 cents), and accessible on-line (in a web page), it is well-suited for studies like ours that require a large number of people to do simple, menial tasks (e.g., clicking points on a surface).

The challenge with any on-line study is to design a protocol that acquires useful information from a diverse population of subjects. Unlike a laboratory study, where a handful of screened subjects are trained, monitored, and employed for several hours, we have access to a much larger number of people, but less control over subject selection and less trust that every individual is motivated to do a good job. As such, our challenge is to design a study that motivates people to work responsibly and incorporates unbiased mechanisms to discard data from those that don't.

We designed an easy-to-learn task in which subjects that provide "better" input get paid more, and those that do not do a "good" job get paid nothing. Specifically, we presented each user with a 3D mesh shown in an interactive viewer with a crystal ball camera control and a simple method for clicking on points with the mouse (one key for adding a point at the selected position on the mesh and another key for removing a previously added point). Then, we provided the following instructions: " select points on the surface of a 3D object likely to be selected by other people. We will ask many people to do the same task and see how your selection matches others." Our reward structure ranked people according to

a scoring function that provides positive credit for each point also selected by at least 25% of other people, negative credit for other points, and zero credit if less than ten points were selected. Based on the ranking, we paid the top scoring 30% of people X, the next 60% X/2, and the bottom 10% were not paid. This incentive structure was chosen based on the results of a pilot study, which suggested that it is useful to provide motivation for people to select more than just the very obvious Schelling points (e.g., the ends of limbs on animals) and that it is better to acquire too many points from people than too few (if a surface has fewer Schelling points, then "extra" points will be distributed somewhat randomly).

Data Filtering: Even though our pay structure encourages people to do a good job, we employ three filters to discard point sets we expect to provide bad data: i.e., when the user: 1) clicks too few points (less than ten), 2) maintains approximately the same camera viewpoint for the entire interactive session (the cumulative camera rotation is less than 36 degrees), and 3) clicks too hastily (the average time per click is less than one second). These filters were chosen on the basis of a pilot study, which showed empirically that a significant fraction of people provided careful data and that these simple filters were effective at conservatively discarding the careless data while retaining much of the good data. In this regard, we designed our filters to favor false-negatives (discarding good data) over false-positives (accepting careless data), since it is easy to collect data on the AMT.

Mesh Selection: In order to cover a wide variety of object categories, and to leverage data collected in previous studies, we use the same mesh data set as described in 2.3.1. The data set is highly tessellated (an average of 10,223 vertices per mesh), so we can limit selected points to vertices of the mesh. In addition, since it has been used for previous studies of mesh matching [81] and segmentation [11], it provides an opportunity to investigate how Schelling points correspond with other types of data.

Protocol Implementation: To reduce bias in the acquired data, we implemented a scheduling program that records the AMT identifier and IP address of every subject doing our task.

That program ensures that no single AMT identifier or IP address can work on the same mesh twice, that meshes are distributed in randomized order, and that data is provided by approximately the same number of subjects for every mesh. The interactive Java Applet presented to the user for viewing meshes and selecting points is initialized with a random camera direction and two virtual lights, all that point at the object centroid from a distance relative to the object bounding box size. As the user rotates the meshes, the lights move with the camera, staying along the equator at 90 degree angles to the camera. The Applet records the position, time stamp, and camera parameters for every point added or deleted from the data set, and sends the data back to our server. No feedback or payments were made to any subject until all data was collected to avoid training effects.

Data Collection: Using this protocol, we used the AMT to acquire 24,124 point sets from 1,696 unique AMT accounts. Of this raw data, 9,965 point sets (44%) from 1,060 unique AMT accounts passed all three data filters – 16% had too few points, 14% were entered with too little camera motion, and 28% were clicked too hastily. The 9,965 point sets in our final data set contain 201,304 points in total, an average of approximately 10,000 points per object category, 500 points per mesh, and 20 points per point set. Each of the 380 meshes was represented by data from on average 25 people, and every mesh had at least 23 point sets.

Schelling Point Extraction: As a post-process, we extract a discrete set of Schelling Points from the collected data – i.e., the ones upon which many people agreed. Of course, since the data is discrete (on vertices of a mesh), and there is spatial noise σ in the point selection process, some aggregation is required to identify commonly selected points. To address this issue, we first construct a function on every mesh M that indicates the number of times every vertex V was selected by different people. Then, to reduce spatial noise, we blur that function geodesically with a Gaussian filter with maximal value 1 and $\sigma = 0.01R$, where R represents the radius of the mesh ($R = \sqrt{SurfaceArea(M)}$). The result is a smooth

Schelling distribution function, $S_D(V)$, roughly estimating the probability that each vertex V will be selected by a person in our study (shown in red in Figures 3.1 and 3.2).

Then, we extract a discrete set of *Schelling points*, S, from local maxima of $S_D(V)$ and build an indicator function, $S_P(V)$, that tells whether vertex V is a Schelling point ($S_P(V)$ is one at Schelling points and zero otherwise). To form this set, we select every vertex V with $S_D(V)$ greater than 12.5% and spatial separation by more than $0.02R$ from any other vertex V_k with higher $S_D(V_k)$. These choices include at least the vertices that were selected with spatial error less than σ by at least 25% of the people in our study (the positive payment threshold in our study). Some examples point sets are shown in blue in Figure 3.2.

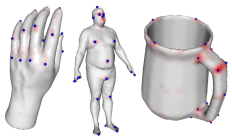
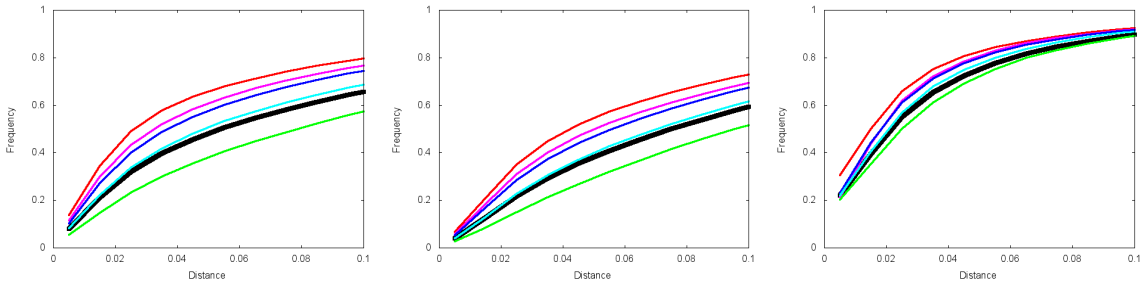


Figure 3.2: *Schelling points. The saturation of red depicts the estimated fraction of people selecting a mesh vertex in our study. Extracted Schelling points are shown in blue.*



(a) Consistency of points selected by different people on the same mesh

(b) Consistency of points selected by people on meshes of the same object class

c) Consistency of points selected by the same person on symmetric sides of a mesh

Figure 3.3: *Consistency of points collected in our study. The plots show cumulative distributions of normalized distance from each point to the closest point in another point set. Different colored curves depict averages for different subsets points. The thick black curve represents the overall average of all points. The thin colored ones separate points based on the order in which they were selected during an interactive session: the consistency of the first point selected is shown in the top red curve, points 2-3 are shown in purple, points 4-7 in blue, etc. Note that the average normalized edge length for these meshes is 0.013, and the normalized distance threshold used to call a point "consistent" in our analysis is 0.05.*

3.5 Analysis of Schelling Points

This large data set provides an opportunity to investigate a number of questions of potential interest in computer graphics and perceptual psychology, including "How consistently do people select points on the same mesh?," "How symmetric are the selected point sets?," "How are the selected points distributed on the surface?," and "What geometric properties of a 3D surface are prominent at selected points?." This section takes steps towards addressing these questions.

A. How consistently do people select points on the same mesh? The first and most basic question is whether people select points consistently in our study.

To address this question, we compute the geodesic distances between points selected by different people on the same mesh. For the k-th point $P^M_{i,k}$ in every point set P^M_i selected on every mesh M, we compute the geodesic distance $d_M(P^M_{i,k}, P^M_j)$ from that point to the closest point in every other point set P^M_j collected on the same mesh, normalizing for scale by dividing by square root of the surface area. We then analyze the consistency of the point

sets by plotting cumulative distributions of $d_M(P^M_{i,k}, P^M_j)$ indicating the fraction of point sets that are consistent with each selected point for a range of distance thresholds.

For example, Figure 3.3a shows cumulative distributions where the horizontal axis represents normalized distance thresholds ($d_M(P^M_{i,k}, P^M_j)$) and the vertical axis represents the fraction of points within that threshold of point sets selected by different people on the same mesh. The thick black curve represents the overall average of all points in the collected data set, aggregated first over all points within the same mesh, then over all meshes within the same object class, and finally over all classes in the data set to avoid over-weighting point sets or meshes with large numbers of points. The thin solid colored curves depict averages for different subsets of the points based on the order in which a user selected a point within his/her interactive session (e.g., the red curve shows distances from the first point selected in each session, the purple curve shows the same for the second and third point, the blue curve for points 4-7, etc. These curves tell a more detailed story of how consistency relates to the order that points were selected.

Looking at the curves in Figure 3.3a, we can readily make two observations. First, different people tend to pick points fairly consistently with one another on the same mesh. Using a normalized distance of $0.05R$ (approximately 3 inches on a human body) as a threshold to classify whether a point $P^M_{i,k}$ is "consistent" with a point set P^M_j, we find that 48.5% of the selected points are "consistent" with other point sets. This level of consistency is far greater than 12.8% that would be observed if points were selected randomly. Second, we observe that points selected earlier in an interactive session tend to be more consistent than ones selected later (there are higher densities near $d_M(p_i, P_j) = 0$ in the curves representing points selected earlier), which suggests that people choose the most "salient" points first, and others later. These results, combined with the visualizations in Figure 3.2, suggest that people selected points in a non-random, consistent way in our study.

B. How consistently do people select points on different meshes of the same object category? A second question of potential interest is whether people select semantically equivalent Schelling points on different meshes within the same object category. For example, if they select a particular point (e.g., the knee) when the object is in one pose (e.g., a running dog), do they also select that point when the same object is in a different pose (e.g., a sitting dog)? And, do they select a semantically equivalent point on a different object within the same general category (e.g., all four-legged animals)?

To address these questions, we produced a semantic mapping between all pairs of the twenty meshes within each of the twenty object categories and used those mappings to compute cumulative distributions of normalized distances between selected points. To produce the mappings, an expert user selected a set of landmark points for every mesh that is semantically consistent with every other mesh in the same category. For example, in the four legged animal category, every mesh was marked with landmark points representing the "tip of the nose," "tip of right ear," "middle of the back," etc. These landmark points provided a coarse point correspondence from which a dense one-way inter-surface mapping $A \rightarrow B$ was established from vertices of A to vertices of B using a simple procedure based on similarities of pairwise geodesic distances (following the strategy outlined in [132], except maintaining the explicit correspondences between landmark points).

Using these inter-surface mappings, we can study the consistency of points selected by people on different meshes within the same object class. Specifically, for each point $P_{i,k}^{M}$ selected by a person on mesh M, we use the inter-surface mapping $M \rightarrow M$ to transfer it to the domain of every other mesh M in the same class to form $P_{i,k}^{M \rightarrow M}$. Then, for every point set P_{j}^{M} collected on mesh M, we compute $d_M(P_{i,k}^{M \rightarrow M}, P_{j}^{M})$, the normalized geodesic distance from the point $P_{i,k}^{M \rightarrow M}$ to the closest point in P_{j}^{M}, and add it to the cumulative distributions using the same procedure as described in the previous subsection. The results are shown in Figures 3.3b.

Interestingly, we find that the consistency of points selected by different people on different meshes in the same class is almost, but not quite, as high as the consistency of points selected by different people on the same mesh. Overall, 39.4% of the points selected on one mesh are "consistent" with point sets selected on different meshes of the same object class. Since the local geometry of meshes within the same class are sometimes very different, this result suggests that the criteria people used for selecting points is not based only on absolute geometric properties (e.g., Gauss curvature), but rather on relative properties (e.g., extrema of Gauss curvature) or on semantic features that are consistent across different instances within the same class.

C. How symmetric are the selected point sets? A third question of interest regards symmetry: if a person selects a point on a symmetric object, how often do they also select the its symmetric correspondence(s)? For example, if a person selects the right ear on a head, how often do they also select the left ear?

To investigate this question, we manually produced a symmetry mapping from every mesh onto itself to establish dense symmetric point correspondences for each of the 319 meshes with intrinsic reflective symmetry. This was done by clicking on symmetric pairs of landmark points (e.g., right eye to left eye, left elbow to right elbow, etc.) and then interpolating those pairs to form a dense correspondence over the entire surface using an algorithm based on Bronstein et al. [132].

Given the symmetric mapping, we analyze the consistency of every point set with its symmetric correspondence using the methods described in the previous subsections – i.e., we build cumulative distributions of the normalized geodesic distance from every point to the closest point in the same point set after the symmetric mapping has been applied (Figure 3.3c). Our results reveal that point sets selected by people are highly symmetric – 76.0% of the points are "consistent" with the symmetric mapping. More specifically, comparing to the histograms in Figure 3.3, we find that point sets selected by people are far

more consistent with their symmetric mapping than they are with point sets selected by different people. As such, we conclude that symmetry is an important cue for selecting "salient" points in our study.

D. How are the selected points distributed on a surface? A fourth question of interest is to characterize the spatial distribution of points selected by people. To address this question, we show the histogram of distances from each point to other points in the same set for meshes of different object class in Figure 3.4. The plot shows a separate curve for each of the 19 object classes (dotted lines), along with the overall average (thick red line).

Three interesting observations can be made from these histograms. First, people tend to spread points fairly evenly on a mesh – i.e., there are few points that are either very close to or very far from other points. Second, the spacing for different object classes is different. For example, points on busts (statue heads) tend to be closer to one another (dotted light blue curve on left), while points on octopi are more widely spaced (light blue curve on right). Third, the spacing of points appears to be dictated more by the size of the mesh shown to the person than the physical size of the object in the real world. For example, the spacing of points on a pair of eyeglasses is the same as for a human body, and also the same as for an airplane. Similarly, the spacing of points on a human hand is different if shown alone or as part of an entire body. From these observations, we hypothesize that people think of the virtual objects at the scale they are shown and select the largest features available on the screen regardless of their size in the real world.

F. What geometric properties distinguish Schelling points? Finally, it is interesting to ask whether it is possible to characterize common geometric properties of Schelling points. For example, are Schelling points commonly at extrema of Gauss curvature? Can Schelling points be predicted by analysis of average geodesic distance or other geometric properties commonly used for feature extraction in computer graphics? And, are there other geometric

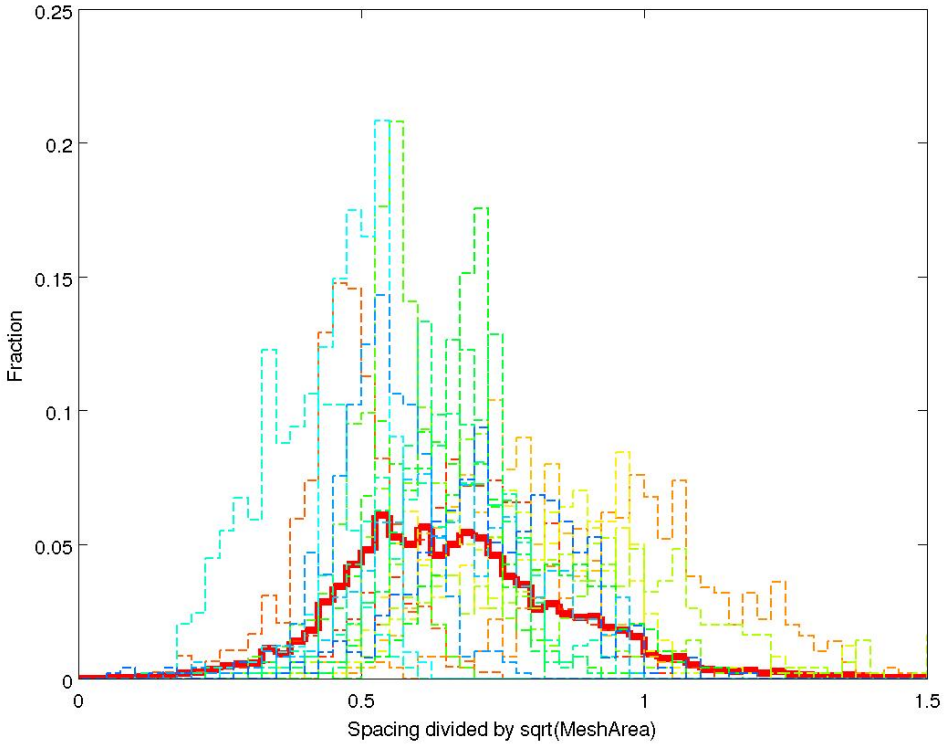


Figure 3.4: *Histogram of spacings between points collected for different mesh categories. Distances are listed as fractions the square root of the mesh area. Red line shows average.*

properties of meshes that are associated with Shelling points, but have not been used for point feature extraction before?

To investigate these questions, we compute a number of geometric properties for every vertex V of every mesh M in the data set, and then analyze how these properties explain the placement of Schelling points based on information theory statistics. The set of properties considered includes:

- **Curvatures:** the Gauss, mean, minimum, and maximum curvatures at V, as computed in Rusinkiewicz et al. [133].
- **Mesh Saliency:** the mesh saliency at V (at scales 0.1, 0.3, 0.5, and 0.7), as computed with code provided by Lee et al. [26].

- **Geodesic Distance:** the average (AGD), median (MGD), standard deviation (SDGD), tenth percentile (10GD), ninetieth percentile (90GD), and maximum of geodesic distance (GGD) from V to all other vertices in M, as estimated by Dijkstra's algorithm.
- **Shape Diameter Function (SDF):** the median length of rays traced from V through the interior of M as described in Shapira et al. [87] using the implementation provided by Kalogerakis et al. [47].
- **Heat Kernel Signature:** the amount of heat diffused from V to itself within time t, for five equally spaced time durations ranging from very small (HKS1) to very large (HKS101), as computed with the implementation provided by Sun et al. [105].
- **Up:** the Z coordinate (ZPosition) and normal direction (ZNormal), assuming it is prescribed by standards in the modeling language.
- **Symmetry:** the Intrinsic Reflective Intrinsic Symmetry Axis function proposed by Xu et al. [97], per our implementation.
- **Segment Centeredness:** the centeredness of the point within its part, as computed by first decomposing the mesh into segments automatically with the algorithm in Shapira et al. [87] and then computing the distance from V to the closest segmentation boundary divided by the maximal distance to the closest segmentation boundary for all vertices in the segment containing V.

Most of these properties have been used before for characterizing saliency and/or generating point sets on meshes. However, we consider two new ones that are specifically motivated by visual inspection of Schelling point sets collected in our study: symmetry and segment centeredness. We observe that people often select points on symmetry axes and centers of large convex parts in the absence of other more distinguishing features nearby (e.g., the center of the lens of the glasses and the belly button of the teddy bear in Figure 3.2. Thus, we expect that extrema of these functions (e.g., segment centeredness = 1) will be correlated with locations of selected points. This hypothesis is corroborated by the

histogram shown in Figure 3.5, which plots frequencies of selected points versus segment centeredness (for automatic SDF segmentations, as used throughout this chapter: and for manual segmentations provided by the Benchmark of 3D Mesh Segmentation [11], which are included only for comparison sake). We find that selected points appear most often at the center of extract parts – i.e., the rightmost bin of the histogram contains 25% of the distribution.

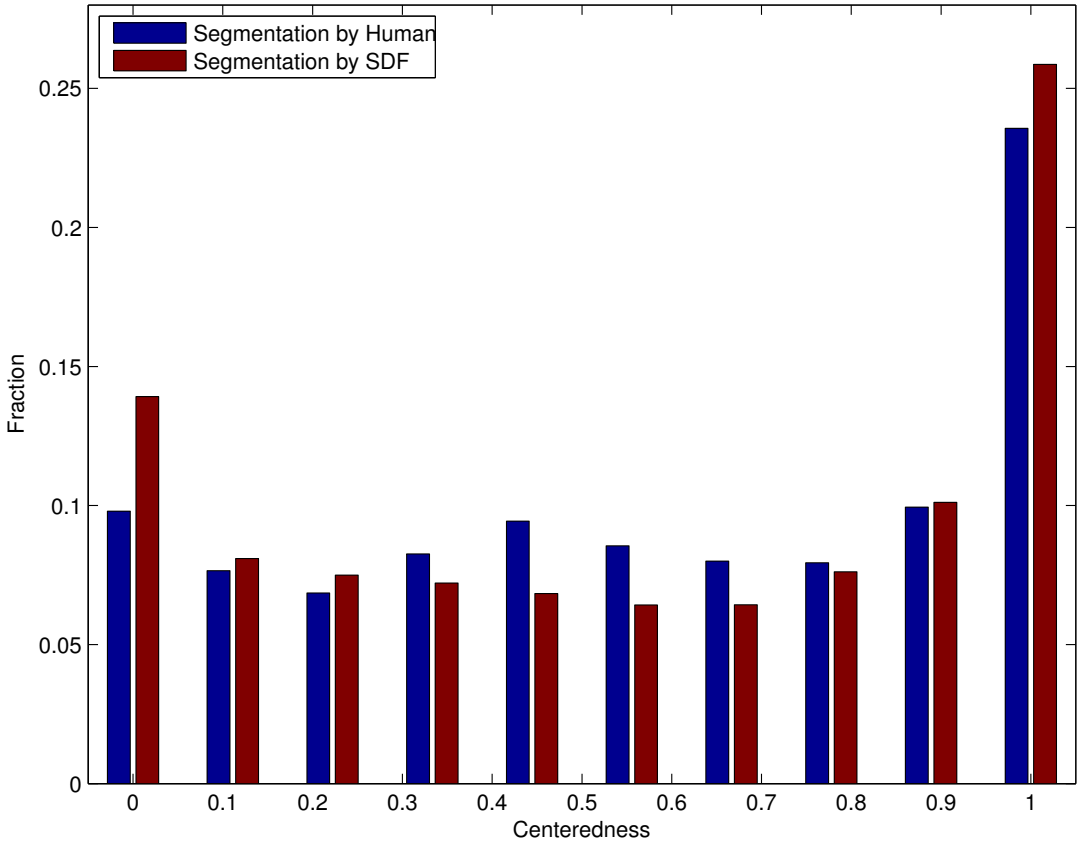


Figure 3.5: *Schelling points are most often at centers of segments (far right of plot).*

We also consider functions derived from these basic geometric properties. Specifically, we include the difference between minimum and maximum curvature (CurvDiff), which is smallest at centers of local rotational symmetry. We include the absolute value of Gauss, mean, minimum, and maximum curvatures, which are largest at critical points, peaks/divots, peaks/divots, and ridges/valleys, respectively. We also include blurs at four levels ($\sigma = 0.01R$, $\sigma = 0.02R$, $\sigma = 0.04R$, and $\sigma = 0.08R$) for all properties except HKS,

CurvRing, and Geodesic Distance, which are already very smooth. Finally, for every property, p, and function of that property, $f(p)$, we include a percentile transformation, %(p), that encodes the percentage of vertices that have a smaller value within the same mesh.

To analyze how Schelling points are associated with these properties, we calculate information theory statistics and coverage plots to estimate how well property distributions explain the Schelling distribution. Results for a representative set of properties and functions on them are shown in Table 3.1 and Figure 3.6.

Information Gain: The middle two columns of Table 3.1 show the Information Gain, $G(f)$, for functions f of several properties p. $G(f)$ is the difference between entropy of the Schelling point indicator function, $S_P(V)$ and its average entropy conditioned on discretized values of the property f – higher values of Information Gain indicate a stronger predictor of the Schelling point locations independent of other properties. From these results, we see that curvature is the strongest single cue for feature point selection – many of the Schelling points are at the tips of protrusions (fingers and toes), which have large positive minimum curvature, and a few are at conspicuous saddle points (e.g., between fingers), which have highly negative Gauss curvature. Other properties commonly used for feature point detection (e.g., Saliency, HKS, and AGD) also provide fairly good predictors.

Random Forest Importance: The rightmost two columns of Table 3.1 show the importance of f as computed by the Random Forests of [134], which estimate the importance of a feature in combinations with others by building a large number of decision trees trained with different sets of properties and measuring the differences between prediction errors with and without each property. It suggests that indeed minimum curvature and Gauss curvature are most useful properties for predicting Schelling points. It also suggests that the *added value* of AGD, HKS, mean curvature, and other common properties is limited, since they are redundant with minimum and Gauss curvatures. On the other hand, the importance of global properties, such as symmetry, SDF, and segment centeredness, is relatively high

Property (p)	Best Filter (f)	Info Gain		Importance	
		G(f)	G(%(f))	I(f)	I(%(f))
MinCurv	$\|(B(p,1))\|$	195	203	216	801
GaussCurv	$\|p\|$	201	211	59	267
Symmetry	$B(p,4)$	12	25	18	84
SDF	p	17	26	31	54
SegCenter	$B(p,3)$	24	43	15	33
HKS(101)	p	104	141	15	31
Saliency(0.3)	p	48	85	26	30
MaxCurv	$B(\|p\|,4)$	53	99	19	30
ZPosition	p	20	51	29	27
ZNormal	p	11	14	24	25
MeanCurv	$B(p,4)$	43	98	11	23
CurvDiff	p	29	46	22	21
AGD	p	25	85	18	18

Table 3.1: *Information gain (x1000) and random forest importance of mesh properties (rows) for predicting the Schelling point distribution. p represents the property value, f is the best filter found for p, $B(\cdot,\sigma)$ is Gaussian blur, $\%(f)$ is its percentile within a mesh, $G(\cdot)$ is information gain, and $I(\cdot)$ is an estimate of importance by analysis of Random Forests.*

(84, 54, and 33, respectively), which suggests they provide cues that are independent of other properties.

We note that percentiles (columns 4 and 6) appear more useful for predicting Schelling points than raw geometric property values: e.g., vertices having curvature higher than others on the same surface are more likely to be Schelling points than ones within any specific range of values.

Coverage plots: Figure 3.6 shows stack plots representing the fraction of Schelling points in S "explained" by extrema of different mesh properties. To produce this result, we extracted the K largest local maxima of each property ($K = 14$, matching the average size of a point set collected in our study), with maxima spread by at least geodesic distance $0.02R$, using the algorithm described in Section 3.7. Then, for each Schelling point, $s_i \in S$, we considered the surface properties in the order from bottom to top, marking the property with a vote if it is the first to have one of its K largest local maxima within $0.05R$ of s_i. After all votes are cast (one for each Schelling point on each mesh), we average the votes per mesh, and then average the fractions per class (shown in the first 19 bars in Figure 3.6),

and finally average those fractions to get a result for the entire data set (shown in the bar labeled "Overall" on the far right).

These results suggest that extrema of minimum curvature "explain" 56.7% of the Schelling points (bottom blue region on the rightmost bar). Of the remaining Schelling points, 6.7% are explained by Gauss curvature (red); and 9.1% of the ones still remaining are explained by symmetry (IRSA); and so on. Please note that there is variation in the results for different object classes: curvature measures are almost completely sufficient for explaining Schelling points on airplanes, for example; but symmetry and segment centeredness are more helpful properties for explaining Schelling points on teddy bears.

Interestingly, approximately 11% of Schelling points are not described by the strongest K local maxima of any property considered. Examining those points visually, we find that they tend to reside either at semantic features off-center on smooth surfaces (e.g., the eyes on the teddy bears) and/or at geometric features that did not appear among the strongest maxima (e.g., a knee or elbow when the appendage is not bent). Perhaps other geometric properties could be discovered to explain these remaining points in future work.

3.6 Prediction of Schelling Distributions

In this section, we investigate whether it is possible to predict mesh saliency using the data collected in our study. That is, for a mesh M never seen before, we aim to provide a method that estimates the probability that a person would have selected each vertex V of the mesh if it were included in our study – i.e., the predicted Schelling point distribution, $\hat{S_D}(V)$.

Transfer: A simple method to approach this problem would be to transfer the Schelling points from other similar meshes for which AMT data has been collected. That is, given a new mesh M, we could select a set of K similar meshes M_i from the collected Schelling point data set, compute a map $m_i(M \rightarrow M_i)$ from each vertex V_j of M to the corresponding

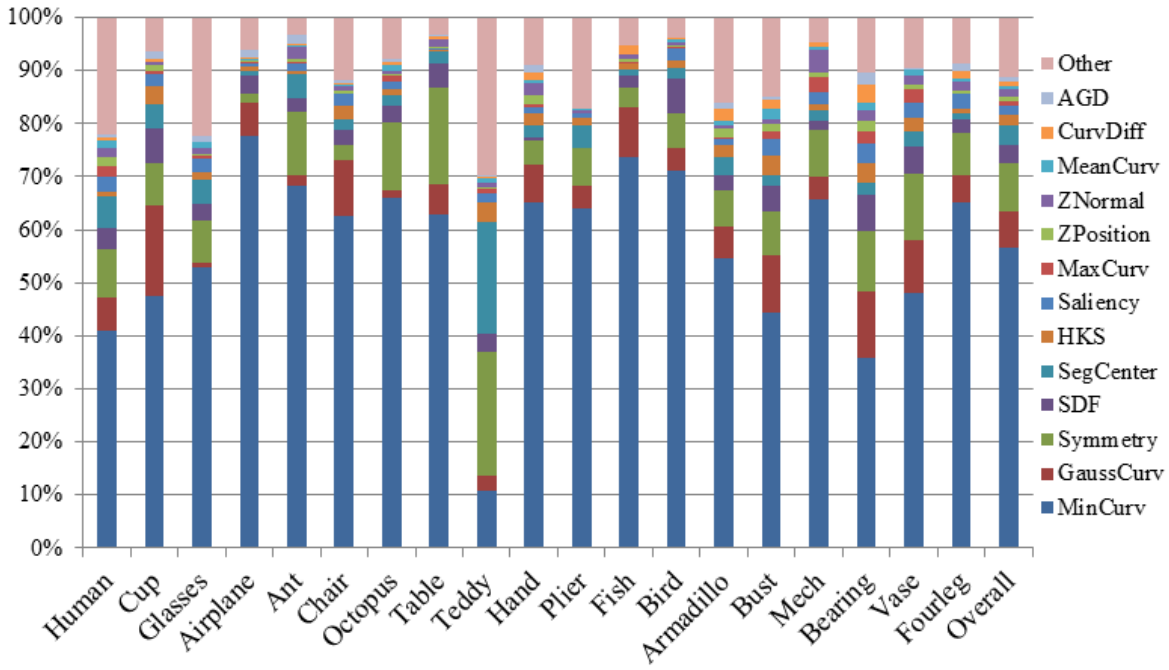


Figure 3.6: *Stackplot showing fractions of Schelling points first "explained" by extrema of each property considered in the order shown bottom to top.*

vertex $m_i(V_j)$ in each similar mesh M_i, and then predict the transferred Schelling point distribution $\hat{S_T}(V)$ for vertices of M by aggregating the Schelling points mapped from other meshes: $\hat{S_T}(V) = 1/K \sum_i^K S_P(m_i(V))$. This method is very simple. However, it will work well only when an accurate map can be computed, and when two meshes have the same number of Schelling points in the same exact arrangement, neither of which is often the case.

Regression: A second method is to build a regression model based on geometric properties of a surface. Given a "training" set of meshes M_i with collected Schelling points, we compute the geometric properties listed in Section 3.5 (Gauss curvature, etc.) for every vertex of every mesh M_i in a training set and then learn a model that relates those properties to the Schelling point indicator function $S_P(V)$. This model is simple to implement, since it considers each vertex of the mesh independently, and it can be applied to arbitrary new surfaces.

Our implementation for building the regression model is based on M5P regression trees as provided by Weka [135]). This model builds a decision tree that takes in a training set of vertices labeled with the properties described in the previous section along with the observed value of the Schelling point distribution at that vertex. It analyzes this data and builds a binary tree that splits feature space into distinct regions and then fits a linear regression model for $\hat{S}(V)$ for each region independently. It was chosen for our study because it can fit non-linear relationships between input and output variables (piecewise linear), and it provides an explanation for how the model operates (the decision tree), which is much easier to decipher than most other regression models.

As an example, Figure 3.7 shows the top nodes of a M5P regression tree learned by our system when trained on the sampled data for teddy bears. Examining the tree, it is interesting to note that the top few levels of splits take into account different properties: minimum curvature, symmetry, segmentation centeredness, shape diameter, saliency, etc. Also, different derived properties are utilized, with percentiles more often chosen for splits in the decision tree, and values or derivatives more often chosen for linear equations in the leaf nodes. Overall, this tree suggests that several types of properties can be combined to make better predictions than any one alone. Of course, nothing in our study is dependent on this particular choice of regression model, and we believe that several other alternatives could have been used just as effectively.

Results: To evaluate and compare how effectively the transfer and regression methods work, we performed a series of leave-one-out experiments where we utilized the Schelling points collected from people on all but one mesh to predict the Schelling point distribution on the one held out, and then we evaluated the quality of the prediction by comparison to the Schelling point data collected for that mesh. Evaluation was performed by computing the Information Gain between the predicted and actual point set distributions, as in Table 3.1.

```
%(Blur(MinCurv,4) <= 65.56
|   Symmetry >  35.669
|   |   SDF >  0.528
|   |   |   Symmetry <= 101.366
|   |   |   |   %(GeodesicTenPercentile) <= 44.399
|   |   |   Symmetry >  101.366
|   |   |   |   ZNormal <= 0.287
%(Blur(MinCurv4) >  65.56
|   SegCenter <= 0.772
|   |   %(MaxCurv) <= 37.769
|   |   |   %(Saliency0.7) <= 89.877
|   |   |   |   %(|Blur(MaxCurve)|,3) <= 33.576
|   |   |   %(Saliency0.7) >  89.877
|   |   |   |   %(Blur(|MaxCurv|),3) <= 77.394
|   |   %(MaxCurv) >  37.769
|   |   |   %(SDF) <= 43.417
|   |   |   |   Blur(Symmetry,4) <= 59.815
|   SegCenter >  0.772
|   |   MeanCurv >  5.973
|   |   |   %(ZNormal) <= 92.829
|   |   |   |   Blur(Symmetry,4) <= 63.682
```

Figure 3.7: *Top nodes of an M5P regression tree learned from Schelling points on teddy bears.*

For the first tests, we used vertex-to-vertex maps to transfer Schelling points to the held out mesh M from other meshes in the same object class. Maps were constructed with two different methods: 1) manually by interpolating human-selected landmark correspondences (TrueMap),[1] and 2) automatically using Blended Intrinsic Maps [136] (BlendedMap). In both cases, we considered cases where points are transferred from just the most similar mesh as computed with a geodesic D2 shape descriptor (Closest), and where they are transferred and aggregated from all meshes in the same object class (InClass). Results of these tests, averaged over all 380 meshes, are shown in the top four rows of Table 3.2. They suggest that transferring points does not work well, in general, For many classes of objects (e.g., fish, vase, chair, airplane, bird, etc.), different meshes have different arrangements of parts, accounting for the failures of even TrueMap. For other classes (e.g., ant, hand, etc.), computing semantically correct maps between meshes is difficult, accounting for the worse performance of BlendedMaps.

For the second tests, we used regression trees learned from Schelling points on a subset of meshes to predict $\hat{S_D}(V)$ for others. We executed two experiments that differ in how

[1] TrueMaps are considered only for didactic purposes. Since they are entered manually, they could not be used in an automatic prediction system.

the training sets were chosen. In the first experiment (InClass), regression trees were built in leave-one-out style using training data from meshes of the same class – i.e., the model was trained on 19 out of the 20 meshes in each object class and then tested on the 20th (as in Kalogerakis et al. [47]). In the second experiment (OutClass), the regression tree for each mesh was trained only using data from meshes of other object classes. Results of this study are shown in Table 3.2. They suggest that regression significantly outperforms transfer algorithms on this data set, even when training is performed only on examples from different object classes. They also show that the regression model combining many surface properties outperforms any single property – i.e., the highest Information Gain achieved for any single property (GaussCurvature) is 0.211 (second row of Table 3.1), in comparison to 0.257 (OutClass) and 0.341 (InClass) achieved with regression.

Prediction Method (p)	Source Data	Best Filter (f)	Info Gain
PredictMap	Closest	$\%(Blur(p,2))$	70
	InClass	$\%(Blur(p,2))$	105
TrueMap	Closest	$\%(Blur(p,2))$	160
	InClass	$\%(Blur(p,2))$	236
Regression	OutClass	$Blur(p,1)$	257
	InClass	$Blur(p,1)$	341

Table 3.2: *Information gain (x1000) of Schelling point predictions.*

Visualizations of Schelling point distributions predicted with a regression model trained InClass are shown in Figure 3.8. The meshes and color scheme are the same as in Figure 3.1 to facilitate direct visual comparison. From these images (and others provided in supplemental materials), we see that these learned models are not as precise as the actual Schelling distributions. However, they predict tips of protrusions well (corners in the airplane) as well as some subtle features (e.g., centers of eyeglasses, eyes of the bust, belly button of the teddy bear, etc.) that are difficult to recognize with a threshold on any single surface property.

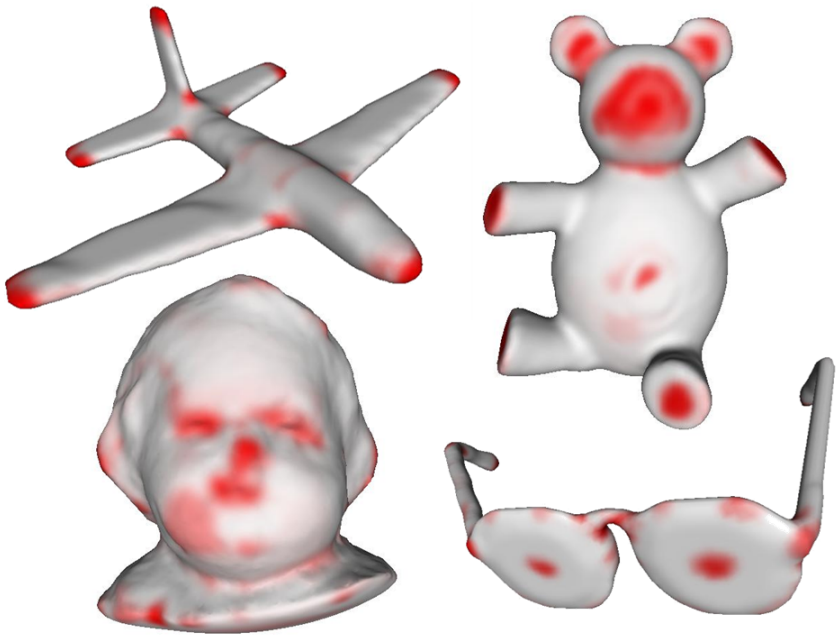


Figure 3.8: *Visualizations of Schelling point distributions predicted by our algorithm after training on properties of different meshes in the same object class (InClass).*

3.7 Prediction of Schelling Points

For many applications, it is important to not only estimate mesh saliency, but also to extract a discrete set of salient feature points. For example, surface matching algorithms often start by detecting a set of features and then searching for correspondences between them. To be successful, these algorithms require a set of feature points that have many of the properties of Schelling points: stability, spacing, symmetry, etc.

Many algorithms are possible to address this problem, including ones that perform a combinatorial optimization over possible candidate sets, choosing the one that maximizes some objective function. However, for the purposes of this study, we choose a simple greedy algorithm that has been used in several other saliency experiments (e.g., Shilane et al. [114]). The algorithm aims to select a set of vertices $\{V_i\}$ that maximizes the sum of $\hat{S}(V_i)$ at selected vertices, subject to the constraint that no two vertices in $\{V_i\}$ are too close to one another. It does so by sorting vertices from highest $\hat{S}(V)$ to lowest and then repeatedly selecting the next vertex remaining whose position is not closer than a normalized geodesic

distance threshold, D, to any previously selected vertex ($D = 0.05$). This process avoids selecting many points near one another on the mesh and provides an easy way to reduce the point set size by increasing the distance threshold D or terminating the greedy search after a given number has been selected.

Results: To investigate how effectively an algorithm can predict Schelling points, we perform an analysis of the similarities of the automatically generated point sets to the ones collected from people. For this analysis, we extract $k = 14$ points for each mesh (the even number closest to the median of point sets collected from people) and then follow the general methodology described in Section 3.5A. That is, we measure the distance from points in our automatically extracted point sets to the Schelling points S_P on the same mesh and plot the cumulative distribution with respect to increasing normalized distance thresholds.

For comparison sake, we consider point sets extracted with the same algorithm from maxima of every surface property considered in Section 3.5, and we consider point sets transferred from Schelling points on other meshes within the same class using the mapping algorithms described in Section 3.6.

Figure 3.9 shows the results. As in Figure 3.3a, the horizontal axis contains increasing distance thresholds, D, and the vertical axis shows consistency using the cumulative distribution of point pairs whose normalized geodesic distance is less than D. The red curve on top represents the consistency between Schelling points S_P collected from people with point sets extracted automatically from the distribution predicted with regression using InClass training. The lower curves represent the best point sets generated with other methods. These results suggest that the points selected by our algorithm from the distribution learned with regression using InClass training are closer to the Schelling points collected in our study than those predicted with other methods. We also find that they are slightly more consistent with the Schelling points than the average point set collected from peo-

ple, suggesting that the regression and point extraction algorithms for predicting Schelling point sets are effectively relating surface properties to Schelling point locations.

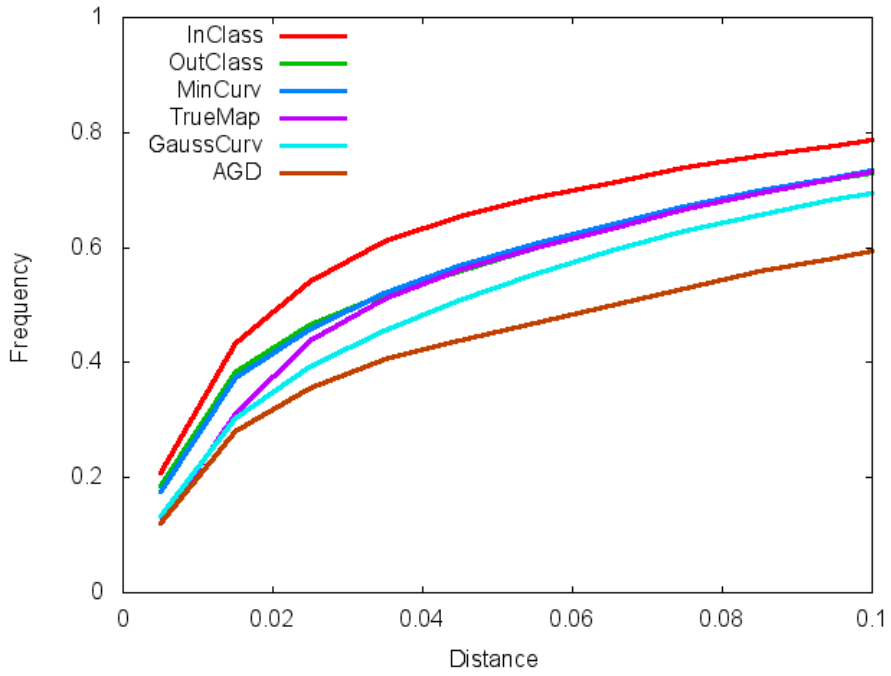


Figure 3.9: *Consistency between Schelling points and point sets extracted with different algorithms.*

3.8 Validation with Controlled User Study

As a final test, we investigated the impact of collecting data via the Amazon Mechanical Turk (AMT) on the main conclusions of our study. Although the AMT is becoming a common platform for perceptual experiments in computer graphics (e.g., Cole et al. [131]) and there are several studies validating AMT studies in the laboratory (e.g.,Heer et al. [137]), it is valuable to ask whether the data we gathered from the AMT is representative of that which would be collected from a controlled group of participants in a laboratory environment.

Of course, it is not practical to duplicate the AMT experiment exactly in the laboratory, since it would take hundreds (or possibly thousands) of hours to collect an equivalent amount of data. Instead, we ran a small study in which we recruited 30 volunteers through email inquiries to acquaintances and students unfamiliar with our project. Of these volunteers, 20 were male and 10 were female. They span a range of ages: 6 were ≤ 20 years old, 16 (21-30), 2 (31-40), 2 (41-50), 3 (51-60), and 1 ($>$60). All except four self-evaluate as having at least "plenty" of experience with computers, but only 9 report having "plenty" of experience with graphics. Every participant performed the study on a Windows computer with a 3-button mouse.

We asked each of the participants to select points on 19 meshes (one selected randomly from each object category) with the exact same instructions and user interface provided on the AMT (substituting the incentive to gain "points" for selections consistent with others' rather than higher monetary payment). After completion, participants completed an exit survey, in which 25 responded "yes" or "mostly" to a question asking whether they completed the task to their satisfaction. We rejected data from the other users, leaving 474 point sets from 25 participants. Of those, 354 (74.7%) passed all three data filters – 66 (13.9%) had too few points, 23 (4.9%) were entered with too little camera motion, and 31 (6.5%) were clicked too hastily. This is a significantly lower rejection rate than for the data collected on the AMT.

Analyzing the point sets passing all filters, we find that they closely matches the characteristics of data collected on the AMT. In particular, the distribution of distances indicating the consistency of points selected by different people on the same mesh (Figure 3.10a) and the reflective symmetry of points selected by the same user on the same mesh are almost identical to those collected on the AMT for the same set of meshes (Figure 3.10b). Of course, there is not enough data to study consistency across different meshes in the same class, or

to study variations within specific subsets of objects. However, all statistics computed for this small, controlled data set closely match those of the larger AMT data set.

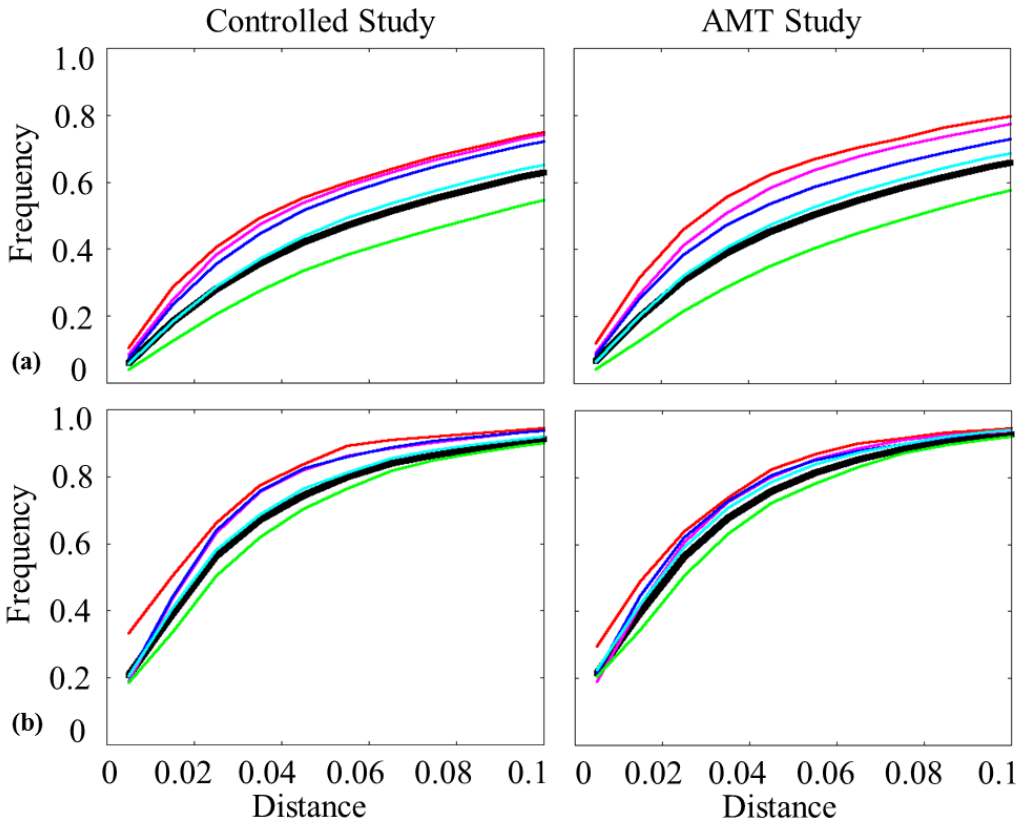


Figure 3.10: *Comparison of consistency and symmetry of point sets collected in a small, controlled study (left) versus ones collected from people on the Amazon Mechanical Turk (right). (a) The top row shows the consistency of points selected by different people on the same mesh (like Figure 3.3a). (b) The bottom row shows the consistency of points selected by the same person on symmetric sides of a mesh (like Figure 3.3c).*

3.9 Conclusion

This chapter described a study of Schelling points on 3D meshes – i.e., points that people expect to be selected by other people. With an on-line experiment, 9,965 points sets containing a total of 201,304 points are gathered. During both qualitative and quantitative analysis, we find that these points appear mainly at "semantically stable" positions on a 3D mesh – e.g., extrema of curvature, on axes of symmetry, centers of segments, etc. We also

find that they are selected fairly consistently by different people on the same mesh, and slightly less so on different meshes of the same object class. However, they are selected very consistently on symmetric parts within the same mesh. We have used this data to train an algorithm for predicting the distribution of Schelling point on new meshes and used it in algorithms for feature point detection.

The saliency measures produced with our methods could be used in a variety of applications in computer graphics (e.g., [93, 92, 90, 66, 26, 95]). For example, Figure 3.11 shows how a predicted Schelling distribution can guide a mesh simplification algorithm to preserve features with higher expected semantic salience (as in Lee et al. [26]). In this case, quadratic errors used by QSlim [138] were scaled by $\hat{S_D}^2$ learned with regression from InClass examples. Note how detail is better preserved in salient areas.

Our study is just a first step and thus has several limitations that suggest topics for future work. First, it considers only watertight meshes with relatively smooth features, and therefore some of the findings regarding local geometric features may not generalize to polygon-soup models commonly found in computer graphics repositories. Second, it considers only point features: studying line and region features would also be valuable. Third, it studies geometric surface properties of the collected data: future work could study other aspects of the data, including the time-dependent strategy people used to select feature points. Finally, it focuses only on applications in computer graphics (mesh saliency and feature point detection): future work might consider questions in perceptual psychology.

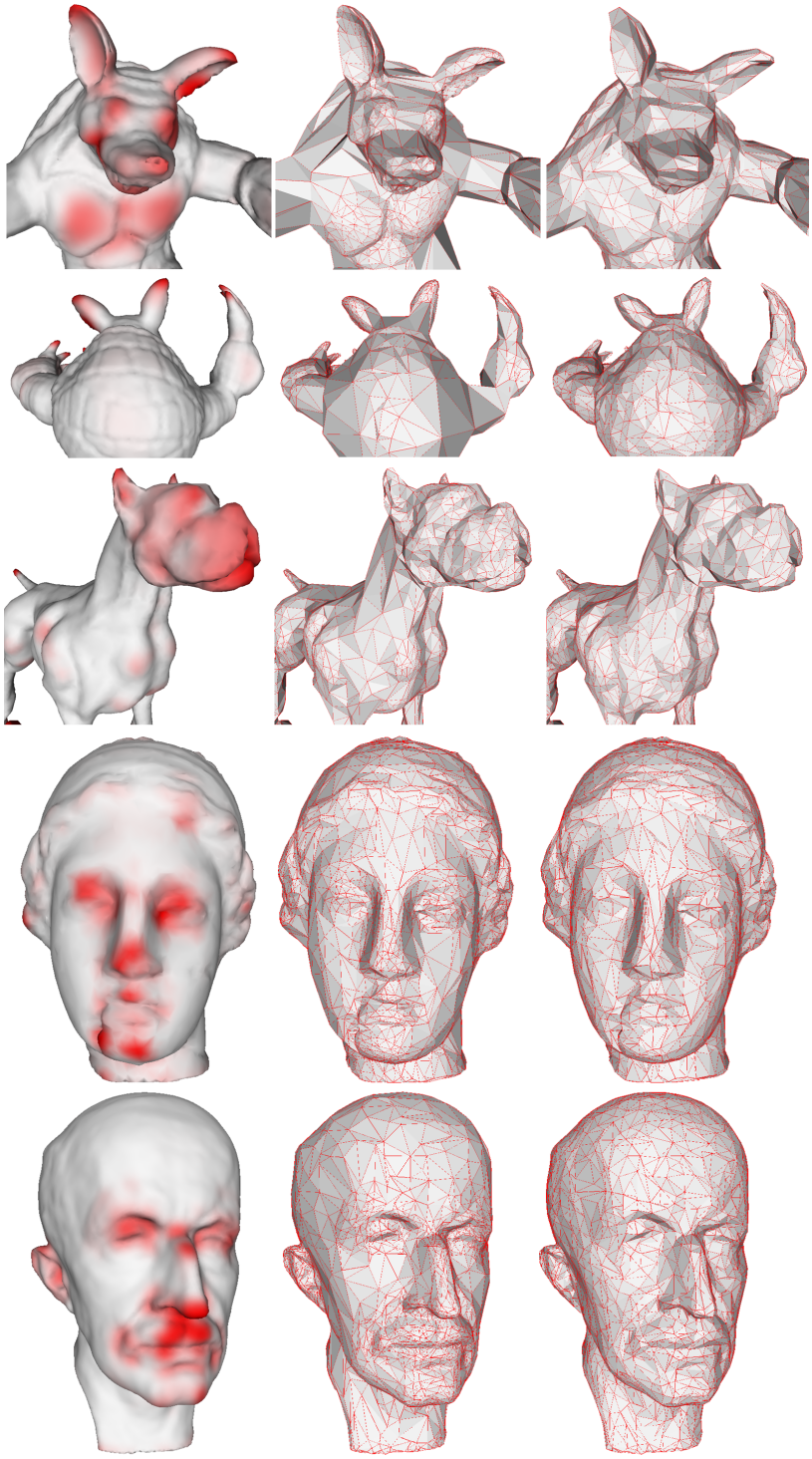


Figure 3.11: *The predicted Schelling distribution $\hat{S_D}$ (left) can be used to preserve salient detail during mesh simplification.*

Chapter 4

MeshMatch

4.1 Introduction

In both chapter 2 and 3, we collected data from humans and built valuable probabilistic distributions over mesh surfaces that reflect what people tend to do. Thus it is desirable to "transfer" the known distributions to an unseen mesh. Ideally, the "transfer" should not only keep the global distribution, but also respect local geometry and maintain local spatial structures of the original distribution.

In fact, transferring surface properties from one mesh to another is an important problem for many applications in computer graphics, including texture transfer [139], detail synthesis [140], shape analysis [141], hole filling [142], and surface editing [143].

Perhaps the most well-known instance of the problem is transferring color texture from one mesh to another. The goal is to produce a pattern on a target mesh that matches the spatially-varying pattern on a given source mesh, replicating both fine-scale textural details and large-scale structural elements of the pattern without visible distortion. This problem has become increasingly important as the demand for textured models in games and movies has exploded, while the availability of studio artists has not.

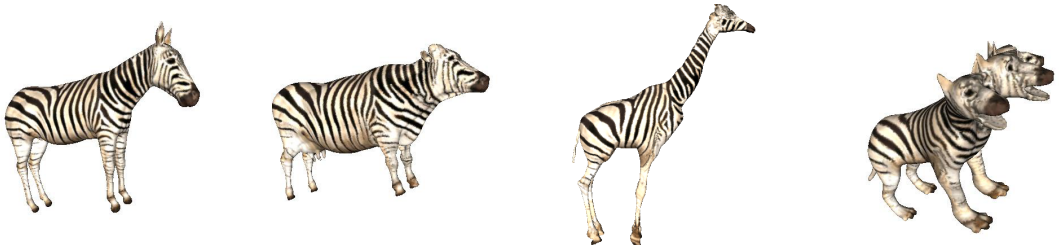


Figure 4.1: *Color texture transfer. From left to right: input zebra; output cow, giraffe, and Cerberus (a three-headed dog).*

Two general approaches have been proposed to address this problem. The first aims to produce a common parametrization that can be used as a mapping from one surface to the other, usually optimizing for a smooth map that minimizes local distortions as measured by deviations in lengths and/or angles (e.g., Alexa et al. [144]). This strategy is able to preserve large-scale patterns in textures, but suffers from noticeable distortion in small-scale details when the source and target surfaces have significantly different shapes (e.g., two animals with different length necks). The second approach uses texture synthesis to copy patches and/or statistics of texture from the source mesh onto the target (Merten et al. [139]). This strategy is able to reproduce small-scale details of textures, but has difficulty preserving large-scale patterns (e.g., different width stripes on different body parts of a zebra).

This chapter proposes a surface texture transfer framework that unifies the best features of surface parametrization and texture synthesis for 3D meshes. The heart of our framework is a surface correspondence algorithm based on PatchMatch [2] that quickly finds the surface patch in the source with shape and color properties most similar to each patch in the target. This algorithm is largely insensitive to the dimensionality of the patch representation and thus enables association of high-dimensional feature vectors representing shape and color descriptors with each patch. Using this algorithm in a multi-resolution synthesis strategy [145, 146] allows our system to efficiently transfer textures preserving both large- and fine-scale patterns of the source.

The key idea behind PatchMatch is to leverage random search and spatial coherence when finding patch correspondences. The law of large numbers suggests that some patches are

likely to find the correct nearest neighbor during a random search, and coherence suggests that good correspondences can be propagated to adjacent patches, yielding full nearest neighbor fields in high-dimensional feature spaces faster than is possible with spatial indexing structures like kd-trees. However, PatchMatch was developed for images, where there is a straightforward parametrization and distances are Euclidean, and the extensions of its key steps to work for 3D surface meshes are not obvious. Non-trivial methods must be developed for selecting appropriate shape and color features, representing surface patches, searching for patch matches at arbitrary orientations, transferring features across resolutions, determining efficient propagation orders, and identifying suitable locations for random jumps. Addressing these issues is our main research contribution.

We demonstrate our surface correspondence algorithm – "MeshMatch" – in the context of a "texture-by-shape" system that uses shape as a guiding layer in the spirit of Image Analogies [146]. Our approach supports several different shape features depending on the application: geodesic affinities generated from sparse manually-specified correspondences, as well as fully-automatic heat kernel signatures. We illustrate the utility of this system in a variety of applications, including texture transfer, detail transfer, and texture editing. In each case, our framework is able to automatically produce patterns on the target that resemble the source's at multiple scales, even when the source and target have relatively different shapes. We also apply our system in transferring Schelling distributions, which acts as an alternative method to predict Schelling distributions on a new mesh.

4.2 Related Work

Understanding how a pattern on one surface can be mapped to another is an important problem in many disciplines, including biology, paleontology, archaeology, etc. In this section, we focus on recent related work on texture transfer in computer graphics.

Parametrization. Perhaps the most obvious approach is consistent mesh parametrization. Recent work has proposed a number of ways of mapping arbitrary surfaces to canonical parametrization domains and then establishing a map with minimal distortion in those domains [144]. For example, consistent parametrization have been established on spheres [147, 148], planes [149], template meshes [141], and automatically computed base meshes [150]. These methods have the advantage that they can guarantee properties of the map (e.g., smoothness), but are suitable only for surfaces with similar topology and shape, as they would otherwise introduce distortions and/or seams in the transferred texture.

Mapping. Other methods compute a map between two surfaces directly via optimization with respect to an analytical distortion measure. For example, Generalized Multidimensional Scaling [151], Heat Kernel Maps [152], Mobius Voting [104], and Blended Intrinsic Maps [136] all aim to find a map minimizing deviations from isometry. Wang et al. [153] register human faces with a conformal map and use the dense correspondences for facial expression transfer. Dinh et al. [154] solve for a mapping between two 3D shapes using PDEs, but they assume that a smooth transformation between the shapes is already provided as input. Such methods are applicable only when the differences between two surfaces are well-approximated by an analytical model of distortion – they do not work well when different parts of the desired map require significantly different distortions (e.g., dragon $\leftrightarrow$ frog) and/or different numbers and arrangements of parts (e.g., insect with four legs $\leftrightarrow$ insect with six legs), which are typical problems of most real-world texture transfer problems.

Texture Synthesis. Other methods have transferred properties between surfaces by example-based texture synthesis. For example, Golovinskiy et al. [140] used the parametric synthesis method of Heeger and Bergen [155] to transfer displacement maps from one face to another. Breckon and Fisher [156] extended the non-parametric approach of Efros and Leung [157] for 3D texture transfer and surface detail in-painting. Bhat et al. [143] performed

similar synthesis operations in the volumetric domain. These methods can be applied to arbitrary surfaces, even when the source and target have different numbers of parts. However, they have only been demonstrated for stochastic textures with highly repetitive patterns. Our work builds on the patch-based optimization approaches introduced to texture synthesis by Kwatra et al. [158] and Wei et al. [159] and extended to the surface domain by Han et al. [160].

The works most closely related to ours are Mertens et al. [139] and Lu et al. [161]. Both papers describe "texture-by-shape" systems that use geometric surface features to guide example-based texture synthesis. They investigate methods based both on parametric and non-parametric models, finding that the parametric methods are able to reproduce weathering effects and other small-scale texture features strongly associated with local geometry. However, the non-parametric methods do not work as well, partially due to the difficulties of finding nearest neighbors in a high-dimensional search space. A goal of our work is to overcome this problem, thereby allowing synthesis of highly structured textures using nonparametric synthesis methods.

4.3 Overview

The objective of this work is to transfer texture/properties from one surface to another while retaining its geometry-aware characteristics, large-scale structural elements, and small-scale details.

To address this problem, we have developed a patch-based texture-by-shape synthesis algorithm based on Image Analogies [146] and PatchMatch [2]. As in other algorithms of this type, we simultaneously solve for the texture of the target mesh M_T and the correspondences to vertices of the source mesh M_S in a multi-resolution optimization. At each level of the optimization, the texture of the target mesh is first initialized, either by up-sampling

from the previous level or at random in the top level, and then an EM algorithm solves for the texture and correspondences for the next level via interleaved optimization. Specifically, the E-Step finds a correspondence from every vertex v_T of M_T to the most similar vertex v_S in the source mesh M_S, where similarity is determined by proximity of color and shape features within patches centered at v_T and v_S. The M-Step then synthesizes a new color for every vertex v_T by voting with colors found in patches of v_S corresponding to patches of M_T overlapping v_T. The iteration terminates when correspondences stabilize, or after a fixed number of iterations in each level.

A major challenge in implementing this patch-based synthesis approach is to find the source patch(es) most similar to each target patch quickly. If patch similarity is determined by the proximity of k features (e.g., $3p$ features for RGB color over a patch with p samples, plus q features for shape), then the problem is to find nearest neighbors in a k-dimensional feature space, which is well-known to be difficult when k is large, resulting in slow algorithms, even when approximate indexing structures are used [162]. In part to avoid this problem, many previous methods for geometry-guided texture synthesis have utilized parametric synthesis algorithms, which are faster but do not replicate large-scale structural elements in synthesized textures.

The key contribution of our work is to leverage the ideas proposed in PatchMatch [2] to accelerate the search for similar patches during non-parametric texture transfer between 3D meshes. The main idea is to leverage the law of random numbers and the spatial coherence of nearest neighbor field when searching for patch correspondences. Random guesses are interleaved with propagation to establish approximate correspondences quickly, without the need for a spatial indexing structure. As a result, finding similar patches is faster, less memory intensive, and less sensitive to the patch size and feature count. These differences enable the use of large patches with descriptive geometric features. In turn, such

informative descriptors help reproduce large-scale structures in transferred textures that are consistent with the model's shape.

The following section describes the implementation of our algorithm, paying special attention to the issues unique to 3D meshes. In Section 4.6, we illustrate results for several possible applications, and we conclude with a brief summary and discussion of topics for future work in Section 4.8. From now on, we will abuse terminology a bit and use texture to refer to color textures as well as other surface properties.

4.4 Algorithms

The input to our system is a source texture T_S stored on vertices of a source mesh M_S and a target mesh M_T; and, the output is a newly synthesized texture T_T stored on vertices of M_T, along with an approximate nearest neighbor field NNF that associates an orientation and vertex of M_S with every vertex of M_T.

As shown in Figure 4.2, the system starts by preprocessing the meshes to build efficient multi-resolution adjacency structures. Given a mesh M, we build L discrete levels via iterative mesh decimation using an algorithm similar in spirit to Q-Slim [138]. Edges are collapsed in shortest-first order, positioning new vertices at the midpoint of the collapsed edge to preserve good aspect ratio, and retaining the "vertex tree" of parent-child relationships to allow efficient traversal between multi-resolution levels. The mesh at each level, M^i, is decimated in this way until it contains approximately half as many vertices as the previous level, and the process stops when there are approximately 128 vertices in the coarsest mesh.

Then, the system performs a multi-resolution optimization of the nearest neighbor field (NNF) and target texture T_T. At each multi-resolution level, proceeding coarse to fine, it alternates between optimizing the NNF (via propagation and randomization) and optimiz-

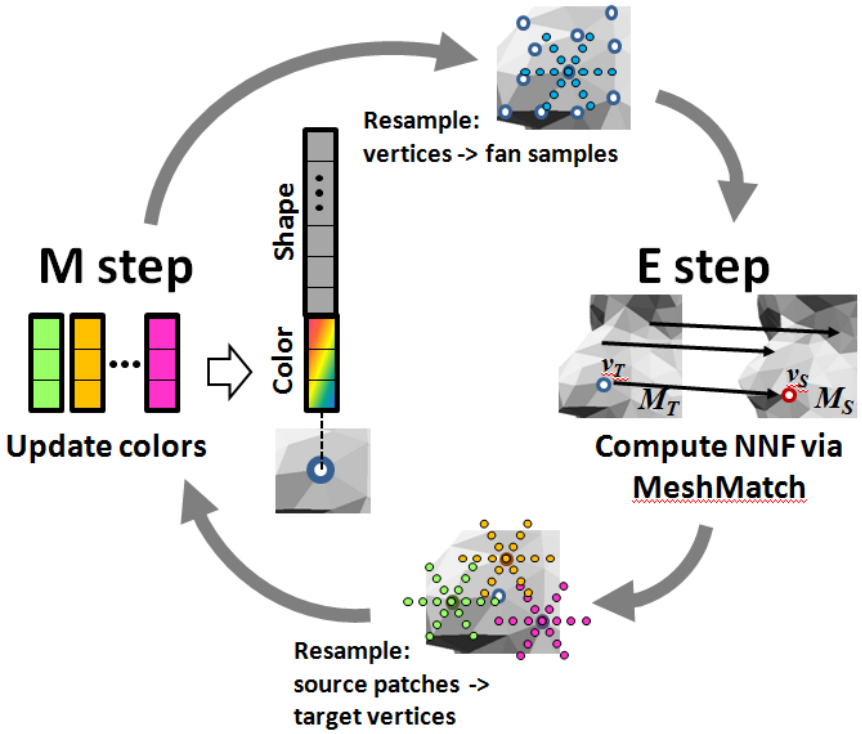


Figure 4.2: *Texture-by-shape algorithm overview.*

ing the texture (via voting) until they jointly converge. The result is up-sampled to the next multi-resolution level and the process iterates. The following subsections describe each step in detail.

4.4.1 Patch Similarity

Before describing steps of the algorithm, we first describe the method used to determine the similarity of patches centered at vertices v_S and v_T. On the one hand, this is an implementation detail, as any distance metric could be used in the context of MeshMatch. On the other hand, it is an important detail, since the goal of the algorithm is to maximize the patch similarity $S(NNF(v_T), v_T)$ for every vertex v_T of the target mesh M_T.

There are several questions that must be addressed: 1) what is the shape of each patch?, 2) what features are stored with patches?, 3) how are the features represented in patches?, 4) how are relative orientations determined?, and 5) how is similarity computed? For images,

answers to these questions are relatively straight-forward (e.g., compute L^2 differences between RGB colors in rectangular patches aligned with regularly sampled pixel locations) [163, 145, 164, 165]. However, they are significantly more difficult for 3D meshes, which have irregular vertex sampling and lack obvious local parametrization.

Surface Patches. For every vertex v of every mesh at level M, we find and store the set of vertices $P(v)$ of M within a given radius r of v via a geodesic walk on vertices of the mesh. We compute a local tangent polar coordinate frame for v and store vertices of $P(v)$ in that frame. We call this representation the "patch" associated with v.

Surface Features. For each patch, we store a set of features indicating the distribution of texture values and geometric properties within the patch. Similarities of these features will be used to determine the similarity of patches.

In this work, we aim to allow both "true" geometric correspondences between semantically corresponding locations, as well as "creative" correspondences provided by an artistic user, that do not necessarily match semantic correspondences — and which also enable manual override when automatic methods fail. Our system therefore supports both automatic and interactively-specified shape features depending on the application:

Our automatic shape features must capture both small-scale details of curvature and also large-scale attributes that place the patch within a broader context of its shape (e.g., features should distinguish the neck from the back of a zebra, since different width stripes are likely to appear in those different regions). To achieve this goal, we compute the Heat Kernel Signature (HKS) [105] at every vertex of the mesh. HKS is a smooth, orientation-invariant, and isometry-invariant shape descriptor that represents for each point p how much heat leaves p and then returns back to it in a given time t. At small time scales, it approximates the local Gauss curvature; and, at large time scales, it approximates the average diffusion distance to other points on the surface. Thus, it captures both small- and large-scale shape features in a common continuous framework. It has recently been used successfully for in-

trinsic symmetry detection and shape matching [166], showing excellent results for finding similarities between similar semantic regions for many types of objects. We augment the HKS vector with one more dimension: the dot product of the surface normal with a prescribed up direction. This up vector is helpful in many cases to distinguish similar shape regions with different semantics (e.g., the back and belly of an animal). Computing it is not onerous, since the up direction is prescribed in most modeling languages (e.g., positive Y in VRML), and automatic methods can be used to align the upright orientation of a mesh in most cases where it is not [167].

For interactive control, we provide an interface allowing a user to specify manual correspondences between points on different models, and compute features using the geodesic distances from each vertex to these constraint points. Unfortunately, geodesic distances alone are not very good shape features for two reasons: First, because the models have different global and local scales, the shape features do not match exactly even at correspondence points. Second, each manually-specified point has global influence across the model, making local control difficult. To address these problems we convert the geodesic distance vectors $G(v)$ to geodesic affinities $G'(v)$ using an affine transform and an exponential:

$$G'(v) = \exp(-M_{T \to S} G(v) / 2\sigma^2) \tag{4.1}$$

The affine transform $M_{T \to S}$ (applied only to feature vectors on the target model) is computed using all pairs of correspondences (v_S, v_T) between source and target models, such that the L_2 distance $M_{T \to S} G(v_T) = G(v_S)$ exactly at all such pairs. (The resulting linear system may be under-constrained, and we find the minimum-norm solution using a linear solver.) The affine transform addresses the first problem with geodesics – meeting the constraints exactly at correspondence points – and the exponential term reduces the influence of correspondences far from v. The choice of σ is not crucial: In all our results we used $\sigma = \max_{i,v} G_i(v)/3$, one third the maximum geodesic distance of any point on the mesh.

Patch Similarity. We define the dissimilarity, $D(v_T, v_S)$, of the patch centered at vertex v_T with the patch centered at v_S as the integral of the L^2 difference of all features at all points in the patch at the optimal aligning rotation:

$$D(v_S, v_T) = \text{argmin}_\theta \int_{p \in P(v_T)} \|F_T(p) - F_S(R(p, v_S, \theta))\| dp \tag{4.2}$$

where $P(v)$ is the surface patch centered at v, $F_T(p)$ is the feature vector associated with every surface point p, and $R(p, v_S, \theta)$ is rotation of p around v_S by θ in the tangent plane.

This definition requires a search over orientations and the evaluation of an integral for every patch comparison, which could be expensive without an appropriate patch representation. We choose it because it provides a robust dissimilarity measure and avoids the challenge of establishing a direction field on a surface.

To accelerate computation of Equation 4.2, we represent the distribution of features within a patch using a geodesic fan (Zelinka et al.[168]), a representation that stores a discrete sampling of surface features in a geodesically circular structure indexed by polar coordinates (r, θ) centered at v (Figure 4.3). This representation provides the advantages of pre-

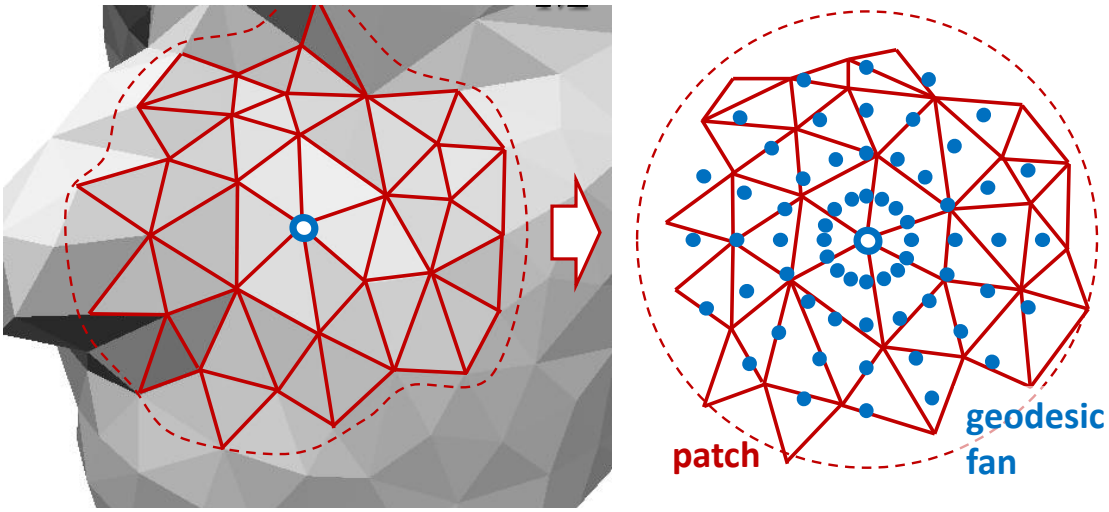


Figure 4.3: *Data structures. Left: 3D mesh (gray) and local patch region (red); Right: patch mapped into local 2D coordinate frame (red) with geodesic fan samples (blue).*

filtering features at discrete samples for rapid comparison and rapid search over relative orientations.

In our current implementation, geodesic fans store filtered estimates of the 3 (RGB) color features at fan samples located at $R = 3$ discrete radii and $A = 18$ discrete polar angles in each fan. The shape features vary more slowly across the surface, so we find it suffices to use only a single shape vector sampled at the center of the fan (8 dimensions for HKS or anywhere from 3 to 20 dimensions for geodesic affinities). Thus we typically have up to a 182-dimensional feature vector associated with each vertex. This is not too different from typical image patch search applications, which often use 147-dimensional color patches.

4.4.2 Optimization

Given this definition of patch dissimilarity, our main task is to find texture values and nearest neighbors that minimize the dissimilarity measure for every vertex of the target mesh. Since the patch dissimilarity depends on both the nearest neighbor field (NNF) and the texture values (T_T), this requires a joint optimization.

Our EM optimization algorithm starts at the coarsest level of the multi-resolution mesh, initializing the NNF randomly. Then, for each multi-resolution mesh level, from coarse to fine, it updates the NNF via propagation and randomization in the E-Step, and then reconstructs an estimate of the target texture from the NNF with a voting algorithm in the M-Step. After the EM iterations have converged, the NNF is up-sampled to the next finer resolution, the target texture is reconstructed at the finer level by re-voting, and the process is repeated until the finest multi-resolution level is reached.

NNF Propagation. During the NNF propagation phase of the E-Step, our goal is to utilize "good" correspondences found in the previous step (possibly by random chance) to find others for adjacent vertices. To do this, we visit each vertex v_T of M_T in order, updating

NNF(v_T) to be the vertex v_S of M_S with highest patch similarity to v_t amongst a set of candidates that includes its current nearest neighbor, NNF(v_T), and all vertices adjacent to nearest neighbors of vertices adjacent to v_T. This phase takes advantage of spatial correspondence in the nearest neighbor field to propagate good correspondences from one vertex to others in its connected component of the mesh.

An issue encountered for 3D meshes in this phase is how to choose a vertex traversal order. For maximal efficiency, we aim to find an ordering that favors visiting adjacent vertices in succession, so that recent updates can be leveraged as each vertex is visited. For images, the natural order is to visit pixels across successive scan-lines. However, no such order exists for vertices of a mesh. So, we select a start vertex randomly within each connected component of M_T and visit vertices in breadth-first order from there. In successive iterations we start from the last vertex in the most recent sequence; this is analogous to alternating forwards/backwards passes in the image-based PatchMatch algorithm.

Note that some care must be taken to propagate consistently under patch rotations. As described in Barnes et al. [169], when searching over rotations, propagation neighborhoods in the target domain must be transformed to the appropriate coordinate frame.

NNF Randomization. During the NNF randomization phase of the E-Step, our goal is to "explore" the space of possible nearest neighbors. To do this, we visit each vertex v_T of M_T, updating NNF(v_T) to be the vertex with highest patch similarity to v_t amongst a set of candidates that includes its current nearest neighbor, NNF($_T$), and vertices sampled from M_S at increasing distances from NNF(v_T). Following PatchMatch, the candidate set includes a random sample within radius r of v_T, another within radius $2r$, and so on. This random sample often contains a good correspondence for v_T, which is propagated to its adjacent vertices in the next iteration.

An interesting issue encountered for meshes in this phase is how to sample vertices matching a desired distribution of distances from a given vertex NNF(v_T). This is easy for images,

since distances are Euclidean. However, for meshes, distances are geodesic, and tracing geodesic paths for long distances on a mesh is too slow for the inner loop of our algorithm. To address this issue, we leverage the hierarchy of the multi-resolution mesh. Since every multi-resolution level covers the same area with half as many vertices as the next, they are spaced approximately $\sqrt{2}$ times as far apart. Thus, we can randomly sample vertices at prescribed distances of $\sqrt{2}^j$ from NNF(v_T) by traversing j levels up the multi-resolution hierarchy from NNF(v_T) and then traversing back down j levels moving to a random vertex in the next finer level at each step down (Figure 4.4). This provides a fast way to sample random vertices from uniform distributions over several region sizes.

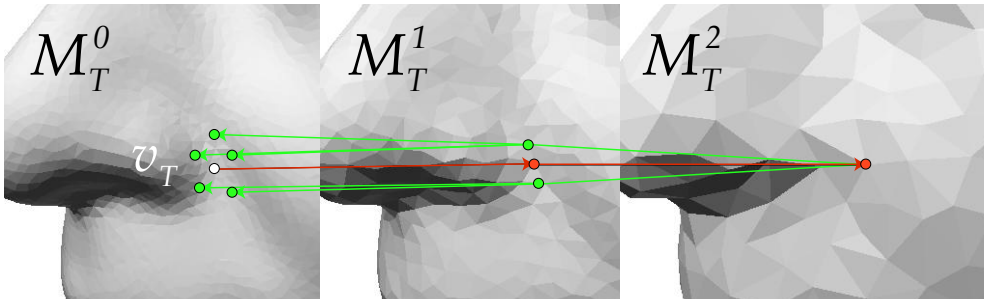


Figure 4.4: *Using the multi-resolution mesh hierarchy to sample random vertices at multiple distances from v_T (shown in white): Ancestor vertices (shown in red) are associated with many descendant vertices (shown in green), allowing for fast random sampling of a large region by hopping multiple levels in the multi-resolution mesh.*

Texture Reconstruction: The M-Step of the optimization updates the target texture based on the current estimate of the nearest neighbor field. For every vertex v_T of M_T, the algorithm performs a weighted averaging (voting) of values from the source texture T_S using the correspondences of nearby vertices (Figure 4.5). Specifically, for every vertex v_p in the neighborhood of v_T, the location of v_T is mapped to a location on M_S by NNF(v_p), and then a color estimate is accumulated from the source mesh vertices using a weighted average, with weights falling off as a Gaussian with the distance from v_T to produce the final reconstruction value.

This Gaussian filter is essential in the inner loop of the algorithm to avoid aliasing, as otherwise feedback loops can amplify spurious patterns. However, the resulting output

is somewhat blurrier than the input texture, so as an optional last step, we reconstruct per-vertex colors at the finest resolution by simply copying the nearest-neighbor color instead of filtering with a weighted Gaussian. The results in Figure 4.1 and Figure 4.8 were produced using this nearest-neighbor sampling.

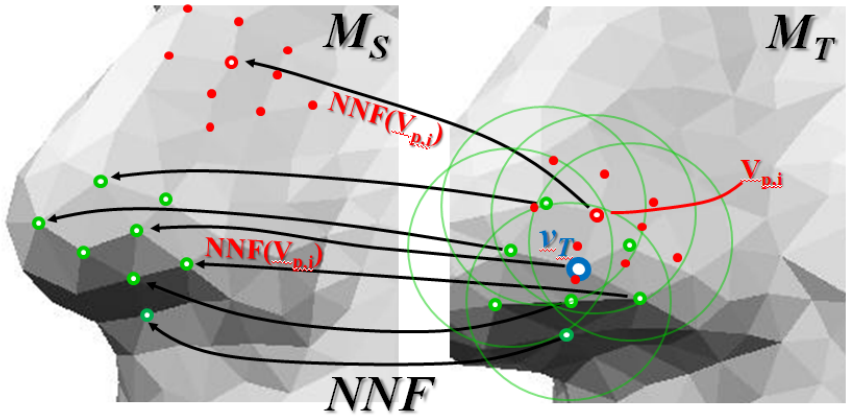


Figure 4.5: *Texture Reconstruction. Geodesic fan samples from the NNF vote for the texture values of each vertex* v_T.

Upsampling. Finally, after the EM optimization has converged for level i, we upsample the NNF to the next finer level $i+1$. If NNF(v_T^i) = v_S^i at level i, we must determine the corresponding vertex v_S^{i+1} and relative coordinate frame for v_T^{i+1} in the next finer level of the multi-resolution hierarchy. To do this, we project each vertex v_T^{i+1} onto M_T^i and determine the geodesic offset and relative orientation with respect to v_T^i. Then, we use NNF(v_T^i) to find v_S^i and reverse the geodesic offset and relative orientation to find the mapped position of vertex v_T^{i+1} on M_S^i. Finally, we map that position and orientation to the finer mesh M_S^{i+1} and find the closest vertex v_S^{i+1}. This provides the up-sampled value: NNF(v_T^{i+1}) $= v_S^{i+1}$.

4.5 Interactivity

The MeshMatch algorithm could be run in an automatic fashion. However, for applications that human interaction/editing is desired, we provide a user interface allowing the user to add and remove corresponding points. It supports creation and deletion of both one-to-one

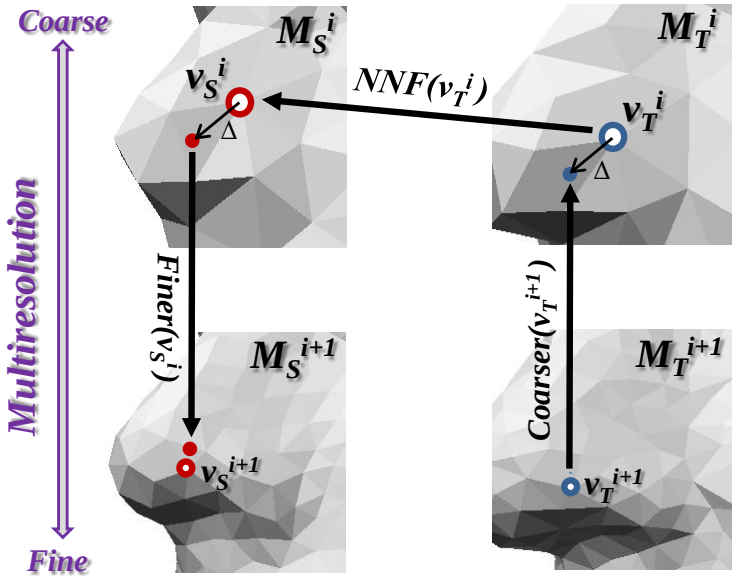


Figure 4.6: *Upsampling. At the end of each iteration, NNFs for vertices in multi-resolution level i are up-sampled from the NNF of level $i+1$.*

and one-to-many constraints, and allows camera motions of source and target views to be locked, facilitating navigation around the models while identifying corresponding points.

In addition, the computation of the texture transfer is fast enough that we can provide progressive updates of the transfer in progress in the target view, enabling the user to decide whether additional correspondences are required to improve the results. Coarse-resolution results are typically visible within a few seconds after adding new correspondences, and fine resolutions are computed in under 30 seconds.

4.6 Application and results

In this section, we demonstrate applications of the proposed algorithms.

4.6.1 Color texture transfer

The most straightforward application of our approach is the transfer of color texture from one 3D model to another. Figure 4.1 illustrates transfer of textures from a zebra to three other creatures. Note that when transferring the zebra's stripes to the other geometry, the resulting stripes are not uniformly or randomly distributed as one might expect using texture synthesis approaches. Instead they retain the same spatial variation as seen in the source model, with widely separated stripes near the hindquarters and thicker, more densely packed stripes along the neck. However, the texture correspondence is not one-to-one: The giraffe's neck has many more stripes than the zebra, and the Cerberus has three noses and six eyes.

This highlights an advantage of our method: Because it does not assume one-to-one mapping between source and destination geometry, it can adapt appropriately to variations in local shape, synthesizing more texture in stretched areas. Thus, our method demonstrates the preservation of high level structure like direct mapping techniques, while maintaining the flexibility of non-parametric texture synthesis approaches.

Even in cases where a one-to-one mapping is possible, it may be extremely deformed and thus unsuitable for direct texture mapping. In Figure 4.7 we compare our method to correspondences computed using blended intrinsic maps [136]. These correspondences – and indeed any one-to-one mapping for this model pair – produce severe distortions in order to match such dramatically different geometry. Our method can use any one-to-one or one-to-many mapping method as initialization and/or soft constraint, synthesizing locally consistent and distortion-free texture while retaining global semantic correspondences.

For color transfer we find that models such as the cow and zebra typically require only about 6 to 10 manual correspondences, and the weights of shape and color features are set to be equal (after normalizing for dimensionality and variance). In some cases we use lower shape weighting at finer scales to encourage more flexible transfer.

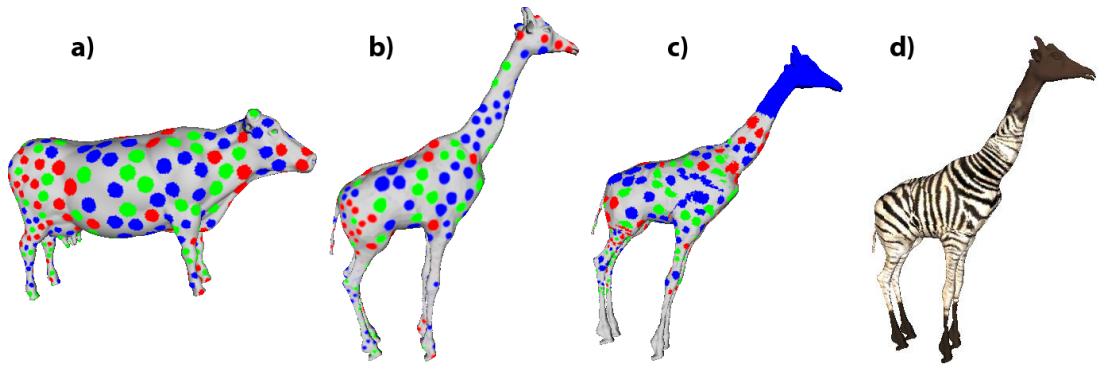


Figure 4.7: *Left to right: (a) input cow with polka-dot, transferred results of (b) ours, (c) of using Blended Intrinsic Maps (BIM), and (d) zebra stripes transferred also using BIM.*

Figure 4.8 illustrates transfer between more widely differing models. Here there are few geometric cues because there is no obvious correspondence between shapes. By adjusting a few correspondence points we can obtain a global correspondence and synthesize a new spatially-varying texture that is locally similar to the original at those points.

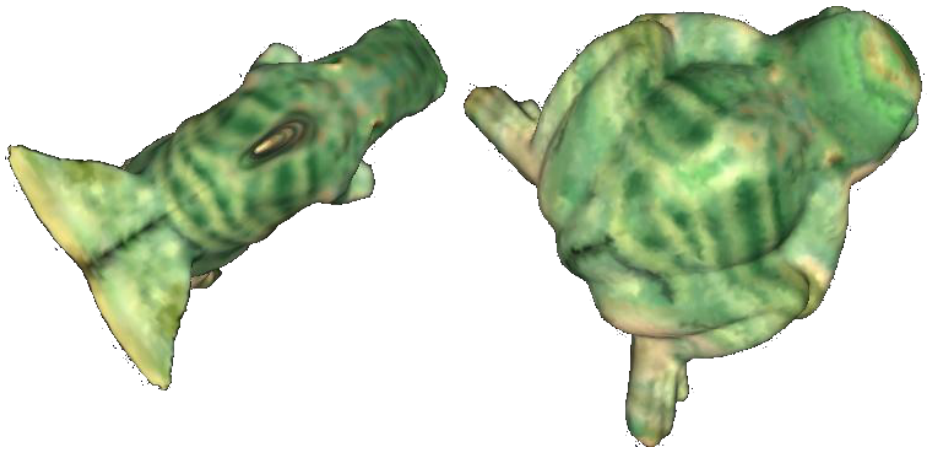


Figure 4.8: *Transfer between widely differing models. Left: input mesh and texture. Right: output texture with correspondences clicked at head, tail, tip of leg and center of back and belly.*

4.6.2 Texture painting power assists

Texture painting in film and game production can require a great deal of manual efforts. What's worse, that effort can be for naught if the topology or geometry of a model is modified after painting, requiring reparameterization and repainting due to distortion. We

propose two uses of MeshMatch to reduce effort in the texture painting work-flow: Texture by example and texture update.

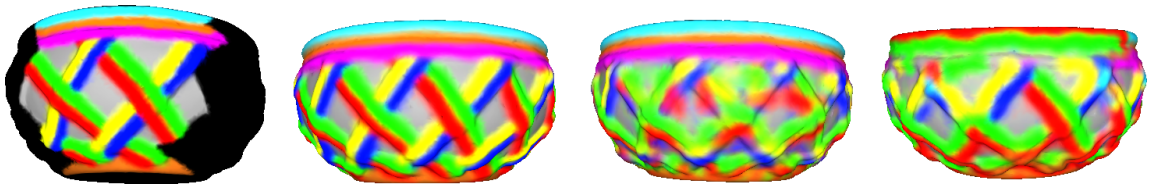


Figure 4.9: *Texture painting by example. Left to right: (a)input mesh with partial painted texture; (b)output mesh with texture propagated to unpainted regions using our method; (c) output mesh by running our algorithm with only shape descriptors (d) output mesh by using smaller fan size and small scale HKS to mimic Mertens et al. [139]. Notice that (b-d) all show the opposite side of the bowl from the side shown on (a).*

Texture by example: In addition to transfer between models, MeshMatch can also be used to transfer texture within a model. This can be useful, for example, to propagate painted textures automatically from regions already painted to regions of a model that have not yet been touched. Figure 4.9 illustrates some examples. Although textures were manually created for just one surface region, they are successfully propagated across the rest of the surface. In this case, the object is globally self-similar, so we use automatically-computed HKS features rather than requiring manual correspondence. However, note that matching HKS features alone (Figure 4.9c) cannot properly resolve the differently colored stripes – only by using local texture patches can we synthesize the proper complex color pattern seen in the source.

Mertens et al. [139] also use shape features to guide texture synthesis. However, their method uses smaller texture patches and mostly local shape features. In Figure 4.9d we demonstrate the impact of patch size and shape feature dimensionality by reducing the fan radii to $R = 2$ and using only one dimension of fine-scale HKS. Because of the efficiency of the MeshMatch matching algorithm, we are able to integrate shape and color features into a single optimization over a high dimensional space. Although Mertens et al. use a broader combination of several different features, their method handles only stochastic

textures with a simple statistical correlation to the underlying shape features, whereas our method can handle essentially arbitrary relationships by using nearest-neighbor search.

Texture update: Figure 4.10 depicts a creature that has been significantly lengthened after textures were painted. In this case we re-synthesized the texture map for the entire torso and copied textures verbatim for other body parts. However, if the geometric modeling package retains knowledge about the portions of geometry that were edited and those that were not, we could make finer grained updates by synthesizing new texture only in modified regions. No explicit reparameterization is required.

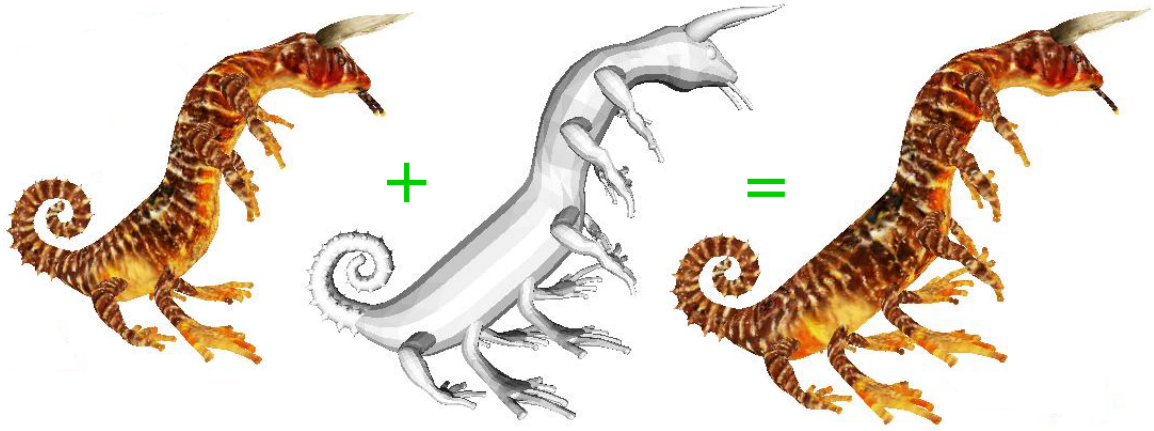


Figure 4.10: *Reapplying texture after geometry changes. Left to right: input geometry and texture, modified geometry, texture updated to fit new geometry.*

4.6.3 Facial detail transfer

Another possible application is transferring fine-scale surface details from one surface to another. For example, Figure 4.11 shows how pores, wrinkles, and creases captured in a highly detailed scan of a face can be used to synthesize plausible details on smooth face scan. To do this, we separate the detailed face scan into a smooth component and a displacement map, and then use the shape features to guide transfer of the displacement map onto the target smooth face. Notice that many details have been synthesized plausibly, including folds in the eyelids, both crows' feet around the eyes, and the nasal labial folds

extending downwards from the nostrils along the cheeks. These results compare favorably with Golovinskiy et al.[140], who provided the input data for this example – their parametric synthesis algorithm does not replicate such large scale structures.

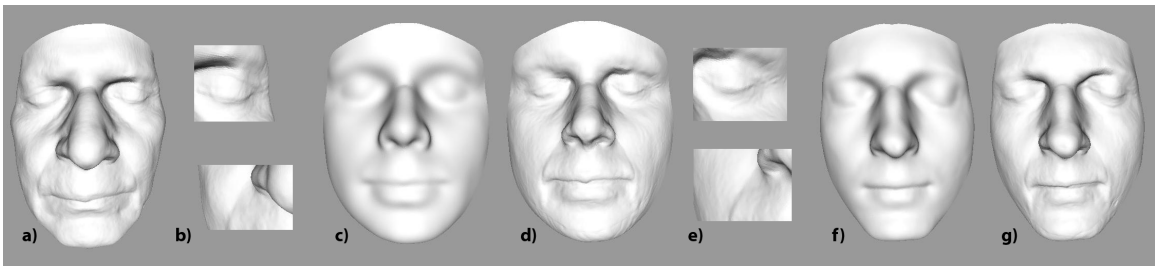


Figure 4.11: *Face detail transfer. (a) SRC detailed face. (b) Zoom-in on eye(top) and nose(bottom) of SRC. (c) DST smooth face 1. (d) Face 1 with details transferred. (e) Zoom-in on eyes(top) and nose(bottom) of face 1. (f) DST smooth face 2. (g) Face 2 with details transferred.*

4.6.4 Schelling distribution transfer

We also use our system to transfer Schelling distribution (Chapter 3) from a collection of known meshes to a new mesh in the same object category. Input is the mesh geometry and the Schelling distribution on known meshes and we use no manual correspondences in the experiments.

We performance a set of leave-one-out experiments on all 19 categories and 380 meshes (20 meshes per category) studied in Chapter 3. Given a mesh M_j^c for category c, MeshMatch transfers Schelling distribution from each mesh $M_i^c \in \{M_k^c, k! = j\}$ of the same category c to M_j^c (denoted as $S_{i->j}$). We then take the average of $\{S_{i->j}, \forall i\}$ as the output for M_j^c. To construct the shape component, we reuse the leave-one-out regression model in Section 3.6 to combine multiple shape features of $\{M_k^c, k! = j\}$, and apply on both the source mesh and the target mesh M_i^c. As "shape" and "color" have similar magnitude, they are equally weighted for all experiments. Note that all parameters are kept the same for the thousands

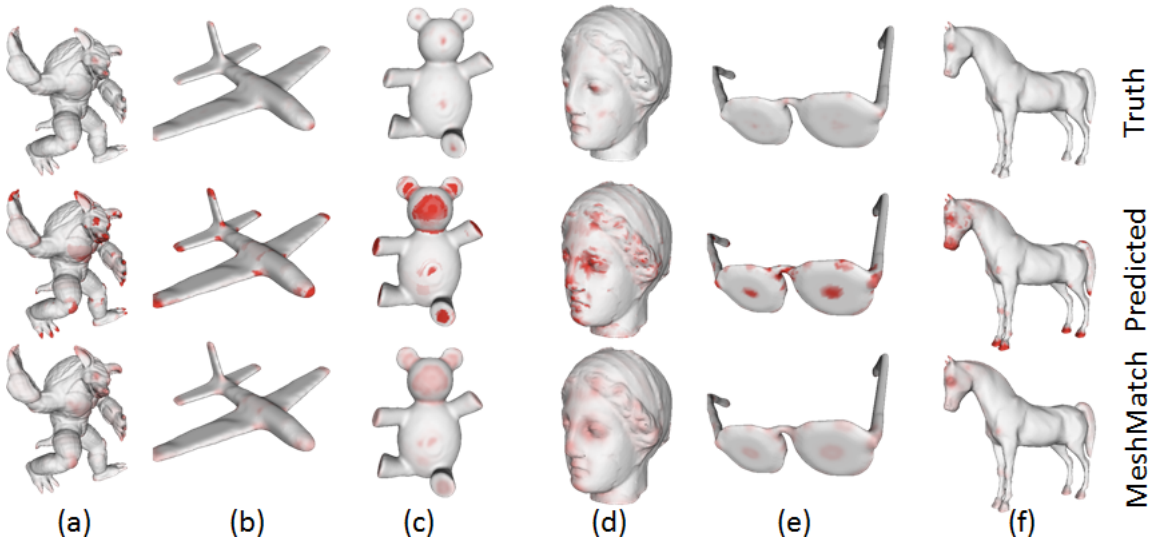


Figure 4.12: *In-class transfer of Schelling distributions. Top row is the groundtruth schelling distribution; middle row is predicted values from regression models in Section 3.6; and bottom row is the average of our in-class transfer results using meshmatch*

run of transfers and it is not impossible to get better results if parameters are tuned category by category.

Figure 4.12 compares Meshmatch results (bottom row) with the ground truth Schelling distribution (top row) and the predicted values from section 3.6 (middle row). Our results using MeshMatch have similar redness as the ground truth while the predicted values tend to amplify the redness. Our results also capture the Schelling distribution peaks at tips, eyes and some segment centers. However, there are failure cases: e.g, the teddy bear face in (c) seems to be strongly influenced by the shape component of the features (which is almost the same as the predicted value).

Table 4.1 shows the L2 distance with respect to the groundtruth Schelling distribution. Both per category and overall results are presented (the lower the better). For all object categories, our results ("Transfer.Avg") have lower L2 distance over the predicted values from regression 3.6("Pred.Inclass"), probably because the regression model tends to amplify the Schelling distribution peaks. We also tried picking the best transfer with the lowest L2 dis-

tance ("Transfer.Best") instead of averaging all transfer results per mesh. Unsurprisingly, the results improve from "Transfer.Avg" but not by much .

Category	Pred.Inclass	Transfer.Avg	Transfer.Best
Human	0.25	0.09	0.08
Cup	0.23	0.15	0.08
Glasses	0.20	0.06	0.05
Airplane	0.20	0.06	0.05
Ant	0.21	0.06	0.06
Chair	0.18	0.06	0.06
Octopus	0.17	0.06	0.05
Table	0.16	0.06	0.05
Teddy	0.22	0.07	0.06
Hand	0.22	0.07	0.05
Plier	0.17	0.06	0.04
Fish	0.15	0.05	0.04
Bird	0.18	0.06	0.05
Armadillo	0.18	0.06	0.05
Bust	0.20	0.09	0.08
Mech	0.11	0.05	0.04
Bearning	0.16	0.10	0.07
Vase	0.20	0.08	0.06
Fourleg	0.20	0.06	0.06
All	0.19	0.07	0.06

Table 4.1: *L2 distance of transferred Schelling distribution versus ground truth (the lower the better)*

While the redness distribution of "Transfer.Avg" is visually more similar to the groundtruth, "Pred.Inclass" still outperforms in the correlation statistics (Table 4.2). We think it is related to multiple sources that blur signals. First is the averaging of 19 transfer results per mesh, which looks blurrier than individual transfers. Second is the influence of the shape component which acts as the guidance. If the shape component is fuzzy for a large region in the coarser level (e.g., on human faces), then color synthesis will have difficulty recovering the true pattern in the finer levels with a smaller fan size. It is possible to alleviate this effect by manual tuning of the shape-to-color weights (as in other examples), but here we stick to a uniform setting for all runs. The third source of blurring is the texture reconstruction and color voting process using a Gaussian filter - that process essentially trades blurring for anti-aliasing.

Category	Pred.Inclass	Transfer.Avg	Transfer.Best
Human	0.45	0.40	0.42
Cup	0.43	0.46	0.49
Glasses	0.53	0.43	0.45
Airplane	0.66	0.53	0.56
Ant	0.51	0.42	0.45
Chair	0.57	0.43	0.48
Octopus	0.60	0.46	0.50
Table	0.52	0.44	0.48
Teddy	0.47	0.40	0.41
Hand	0.51	0.40	0.43
Plier	0.59	0.49	0.53
Fish	0.66	0.58	0.61
Bird	0.60	0.51	0.54
Armadillo	0.60	0.44	0.48
Bust	0.35	0.35	0.38
Mech	0.60	0.60	0.65
Bearning	0.61	0.60	0.65
Vase	0.45	0.42	0.45
Fourleg	0.53	0.46	0.48
All	0.54	0.46	0.50

Table 4.2: *Correlation of transferred Schelling distribution versus ground truth (the higher the better)*

4.7 Speed

Our system runs at interactive rates. The MeshMatch portion of the algorithm takes only 5 iterations and 6.5 seconds to converge for a mesh with 64K vertices, each represented by a 182-dimensional feature vector. This requires no extra storage for spatial indexing structures, beyond the nearest neighbor field itself.

The texture transfer system based on MeshMatch employs several mesh processing steps that are not fast, but can be precomputed offline once and use again and again. Specifically, for a mesh with 64K vertices, building the multi-resolution data structure takes 120 seconds and building geodesic fans takes 40 seconds. However, the remainder of the pipeline runs at near-interactive rates: at the finest scale (64K vertex mesh), finding nearest neighbors with MeshMatch takes 6.5 seconds per iteration, and reconstructing the texture from the correspondence field takes 5.5 seconds per iteration. These runtimes scale linearly with

the number of vertices, so a rough preview at 16K vertices takes only about 3 seconds. Using five EM iterations at five multi-resolution levels, the total texture transfer time is approximately 100 seconds.

Several further speed improvements are possible: For example, MeshMatch presently uses a fast randomized search for vertices but exhaustive search over orientations. Barnes et al. [169] have demonstrated that image patch search over orientations can also be performed efficiently using randomized search with propagation, so we believe augmenting our method using a similar technique could result in a possible additional 10x speedup.

4.8 Conclusion and Future Work

This chapter has described a surface correspondence algorithm based on PatchMatch [2] and demonstrated its use in a variety of applications to transfer texture, details, and Schelling distribution. The most important advantage of this algorithm is that it enables finding nearest neighbors between surface points in a high-dimensional feature space encoding large-scale geometric and color features. As a result, non-parametric texture synthesis algorithms can be used to replicate large-scale patterns like stripes from a zebra and creases from a face, in addition to small scale patterns like pores on a face.

Our current implementation is just a first prototype and has several limitations that suggest directions for future work.

First, our system has some parameters that might need to be adjusted for each example. The main parameters we have adjusted are the starting scale and the different shape-vs.-color weights for each level of the multi-resolution hierarchy – these are adjusted based on the size of patterns in the source texture. While these parameters can be useful for artistic control, it would be interesting to investigate future methods to set them automatically based on scale-space analysis of the source data.

Second, our interactive shape features were designed for efficient computation, and not quality. In particular, the mapping becomes difficult to control when more than about 15 correspondences are added, as the influence of each point affects a large portion of the model. In addition, one-to-many correspondences create extreme distortion in the shape mapping in regions where the one-to-one mapping is split to one-to-many. Although our method can still synthesize plausible texture in these regions, better methods for both global and local control are desirable. In addition, better results will achieve if shape features suitable for the specific application are selected, but this is a separate topic.

Finally, it will be interesting to study which applications can best leverage the types of surface correspondences found by MeshMatch. Unlike much of the previous work in inter-surface mapping, our algorithm does not aim to find a smooth, bijective map between surfaces with low distortion everywhere. Rather, it aims to find correspondences between points with similar features. We have demonstrated that correspondences of this type are useful for texture transfer in computer graphics, but it will be interesting to see what other types of applications can benefit from this approach (e.g., computer vision, archaeology, art, etc.).

Chapter 5

Conclusion and Future Work

This thesis studies two classical problems in shape analysis. Motivated by the fact that humans are superior than computers in addressing these problems, we revisit these problems from the angle of using crowdsourced data to learn from humans.

First, we investigated the mesh segmentation problem. We designed a benchmark of 4300 manually generated segmentations for 380 surfaces meshes. This data reveals a probabilistic distribution of how people decompose shapes into parts. It also enables us to propose metrics and software to evaluate performances of mesh segmentation algorithms, and prompts learning-based mesh segmentation algorithms that achieve the state-of-art quality.

Second, we visited the problem of salient point detection on meshes from a social/psychological perspective, with Schelling experiments asking people to select points that are most likely to be selected by other people. The resulting Schelling distribution becomes a base for a new model of saliency, which predicts saliency on unseen meshes using our regression models.

Common to these two problems are the desire to transfer properties from existing meshes to unseen meshes. Hence we propose the MeshMatch algorithm and a non-parametric multi-

resolution approach based on the algorithm for fast property transfer. We show a variety of applications including the transfer of textures, details, and Schelling distribution.

Our experience shows that for problems that humans naturally and easily solve but computers do not, it is valuable to understand and mimic what humans do. The data-driven approach combined with crowdsourced data from well-designed experiments is powerful, and could possibly advance problems that are even well-researched in the past.

5.1 Future Work

We envision several directions for future work.

First, it is worth identifying more problems in shape analysis and broader domains where approach in this thesis could be similarly applied. Whereas we focus on problems analyzing single objects, it might be interesting to analyze scenes and the interaction between objects. We also see a trend with recent papers [12, 14, 15] applying data-driven approach to quantitatively address problems related to human semantics and being vague to describe.

Second, it is desirable that generalized principles and tool-kits could be created to ease the design and implementation of crowdsourcing systems. Crowdsourcing is powerful and could be widely used, but the design is often ad-hoc to the specific problems. We are glad to see multi-disciplinary research efforts from HCI, security, systems and economics emerging [41, 170, 33, 42].

Third, data collection is currently separated from analysis and modeling, but interleaving all three together might offer benefits. On the one hand, the analysis and data training might direct the system to collect more data for cases harder to analyze and/or model; on the other hand, some applications could possibly benefit from "online learning". Related to this are human computation systems [36, 34] that integrate algorithms with human decisions.

In shape analysis, computer graphics and many other fields, algorithms still have a lot to learn from humans. With ever bigger data crowdsourced, and with analytic and learning techniques advanced, I believe the gap between algorithms and humans will shrink more.

Bibliography

[1] Chen, X., Saparov, A., Pang, B., Funkhouser, T.: Schelling points on 3D surface meshes. ACM Transactions on Graphics (Proc. SIGGRAPH) **31** (2012)

[2] Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: PatchMatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics (Proc. SIGGRAPH) **28** (2009)

[3] Amazon: Mechanical turk (2009) http://www.mturk.com.

[4] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM (2004) 175–184

[5] Johnson, A., Hebert, M.: Using spin-images for efficient multiple model recognition in cluttered 3-D scenes. IEEE PAMI **21** (1999) 433–449

[6] Mitra, N.J., Pauly, M., Wand, M., Ceylan, D.: Symmetry in 3d geometry: Extraction and applications. In: EUROGRAPHICS State-of-the-art Report. Volume 1. (2012) 7

[7] Lipman, Y., Chen, X., Daubechies, I., Funkhouser, T.: Symmetry factored embedding and distance. ACM Transactions on Graphics (SIGGRAPH 2010) **29** (2010)

[8] Zhang, H., Van Kaick, O., Dyer, R.: Spectral methods for mesh processing and analysis. In: Proceedings of Eurographics State-of-the-art Report. (2007) 1–22

[9] Kazhdan, M., Funkhouser, T., Rusinkiewicz, S.: Rotation invariant spherical harmonic representation of 3d shape descriptors. In: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Eurographics Association (2003) 156–164

[10] Ankerst, M., Kastenmüller, G., Kriegel, H.P., Seidl, T.: 3d shape histograms for similarity search and classification in spatial databases. In: Advances in Spatial Databases, Springer (1999) 207–226

[11] Chen, X., Golovinskiy, A., Funkhouser, T.: A benchmark for 3D mesh segmentation. ACM Transactions on Graphics (Proc. SIGGRAPH) **28** (2009)

[12] Cole, F., Golovinskiy, A., Limpaecher, A., Barros, H.S., Finkelstein, A., Funkhouser, T., Rusinkiewicz, S.: Where do people draw lines? ACM Transactions on Graphics (Proc. SIGGRAPH) **27** (2008)

[13] Cole, F., Golovinskiy, A., Limpaecher, A., Barros, H., Finkelstein, A., Funkhouser, T., Rusinkiewicz, S.: Where do people draw lines? Communications of the ACM **55** (2012) 107–115

[14] O'Donovan, P., Agarwala, A., Hertzmann, A.: Color compatibility from large datasets. In: ACM Transactions on Graphics (Proc. SIGGRAPH). Volume 30., ACM (2011) 63

[15] Secord, A., Lu, J., Finkelstein, A., Singh, M., Nealen, A.: Perceptual models of viewpoint preference. ACM Transactions on Graphics (TOG) **30** (2011) 109

[16] Eitz, M., Berlin, T., Hays, J., Alexa, M.: How do humans sketch objects? ACM Trans. Graph. (Proc. SIGGRAPH) **31** (2012)

[17] Biederman, I.: Recognition-by-components: A theory of human image understanding. Psychological Review **94** (1987) 115–147

[18] Hoffman, D.D., Richards, W., Pentl, A., Rubin, J., Scheuhammer, J.: Parts of recognition. Cognition **18** (1984) 65–96

[19] Hoffman, D.D., Singh, M.: Salience of visual parts. Volume 63. (1997)

[20] Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Transactions on Graphics (Proc. SIGGRAPH) **22** (2003) 954–961

[21] Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., Seidel, H.: Mesh scissoring with minima rule and part salience. Computer Aided Geometric Design **22** (2005) 444–465

[22] Koch, C., Ullman, S.: Shifts in selective visual attention: towards the underlying neural circuitry. Human Neurobiology **4** (1985) 219–227

[23] Attneave, F.: Some informational aspects of visual perception. Psychological Review **61** (1954)

[24] Privitera, C., Stark, L.: Algorithms for defining visual regions-of-interest: Comparison with eye fixations. PAMI **22** (2000) 970–982

[25] Kim, Y., Varshney, A., adn François Guimbretière, D.J.: Mesh saliency and human eye fixations. ACM Transactions on Applied Perception **7** (2010)

[26] Lee, C.H., Varshney, A., Jacobs, D.W.: Mesh saliency. ACM Transactions on Graphics (SIGGRAPH 2005) (2005)

[27] Yuen, M., King, I., Leung, K.: A survey of crowdsourcing systems. In: Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Confernece on Social Computing (SocialCom), IEEE (2011) 766–773

[28] Doan, A., Ramakrishnan, R., Halevy, A.: Crowdsourcing systems on the world-wide web. Communications of the ACM **54** (2011) 86–96

[29] Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, Ieee (2009) 248–255

[30] Fellbaum, C.: Wordnet. Theory and Applications of Ontology: Computer Applications (2010) 231–243

[31] Von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M.: recaptcha: Human-based character recognition via web security measures. Science **321** (2008) 1465–1468

[32] Ahn, L., Blum, M., Hopper, N., Langford, J.: Captcha: Using hard ai problems for security. In: Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques, Springer-Verlag (2003) 294–311

[33] Bernstein, M., Little, G., Miller, R., Hartmann, B., Ackerman, M., Karger, D., Crowell, D., Panovich, K.: Soylent: a word processor with a crowd inside. In: Proceedings of the 23nd annual ACM symposium on User interface software and technology, ACM (2010) 313–322

[34] Von Ahn, L.: Human computation. In: Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE, IEEE (2009) 418–419

[35] Quinn, A., Bederson, B.: Human computation: a survey and taxonomy of a growing field. In: Proceedings of the 2011 annual conference on Human factors in computing systems, ACM (2011) 1403–1412

[36] Cooper, S., Khatib, F., Treuille, A., Barbero, J., Lee, J., Beenen, M., Leaver-Fay, A., Baker, D., Popovic, Z., et al.: Predicting protein structures with a multiplayer online game. Nature **466** (2010) 756–760

[37] Von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM (2004)

319–326

[38] Gingold, Y., Shamir, A., Cohen-Or, D.: Micro perceptual human computation for visual tasks. ACM Transactions on Graphics (TOG) (2012)

[39] Mason, W., Watts, D.: Financial incentives and the performance of crowds. ACM SIGKDD Explorations Newsletter **11** (2010) 100–108

[40] Horton, J., Chilton, L.: The labor economics of paid crowdsourcing. In: Proceedings of the 11th ACM conference on Electronic commerce, ACM (2010) 209–218

[41] Huang, E., Zhang, H., Parkes, D., Gajos, K., Chen, Y.: Toward automatic task design: A progress report. In: Proceedings of the ACM SIGKDD workshop on human computation, ACM (2010) 77–85

[42] Bernstein, M., Brandt, J., Miller, R., Karger, D.: Crowds in two seconds: Enabling realtime crowd-powered interfaces. In: Proceedings of the 24th annual ACM symposium on User interface software and technology, ACM (2011) 33–42

[43] Bernstein, M., Tan, D., Smith, G., Czerwinski, M., Horvitz, E.: Personalization via friendsourcing. ACM Transactions on Computer-Human Interaction (TOCHI) **17** (2010) 6

[44] Kin-Chung Au, O., Zheng, Y., Chen, M., Xu, P., Tai, C.: Mesh segmentation with concavity-aware fields. Visualization and Computer Graphics, IEEE Transactions on (2011) 1–1

[45] Huang, Q., Koltun, V., Guibas, L.: Joint shape segmentation with linear programming. In: ACM Transactions on Graphics (TOG). Volume 30., ACM (2011) 125

[46] ZHANG, J., ZHENG, J., WU, C., CAI, J.: Variational mesh decomposition. Transaction on Graphics **31** (2012)

[47] Kalogerakis, E., Hertzmann, A., Singh, K.: Learning 3d mesh segmentation and labeling. ACM Transactions on Graphics (proc. SIGGRAPH) **29** (2010)

[48] Chen, X., Funkhouser, T., Goldman, D.B., , Shechtman, E.: Non-parametric texture transfer using meshmatch. Adobe Technical Report 2012-2 (2012)

[49] Biasotti, S., Marini, S., Mortara, M., Patanè, G.: An overview on properties and efficacy of topological skeletons in shape modelling. In: Shape Modeling International. (2003) 245–256, 297

[50] Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. ACM Trans. Graph. **23** (2004) 652–663

[51] Zöckler, M., Stalling, D., Hege, H.C.: Fast and intuitive generation of geometric shape transitions. The Visual Computer **16(5)** (2000) 241–253

[52] Gregory, A.D., State, A., Lin, M.C., Manocha, D., Livingston, M.A.: Interactive surface decomposition for polyhedral morphing. The Visual Computer **15** (1999) 453–470

[53] Zuckerberger, E.: Polyhedral surface decomposition with applications. Computers and Graphics **26** (2002) 733–743

[54] Lévy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. ACM Trans. Graph. **21** (2002) 362–371

[55] Attene, M., Katz, S., Mortara, M., Patane, G., Spagnuolo, M., Tal, A.: Mesh segmentation - a comparative study. In: SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006, Washington, DC, USA, IEEE Computer Society (2006) 7

[56] Over, P., Leung, C., Ip, H., Grubinger, M.: Multimedia retrieval benchmarks. IEEE MultiMedia **11** (2004) 80–84

[57] Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: in Proc. 8th Int'l Conf. Computer Vision. (2001) 416–423

[58] Smeaton, A.F., Kraaij, W., Over, P.: TREC video retrieval evaluation: a case study and status report. In: Coupling Approaches, Coupling Media and Coupling Languages for Information Retrieval (RIAO). (2004)

[59] Federico, M., Giordani, D., Coletti, P.: Development and evaluation of an Italian broadcast news corpus. In: Second International Conference on Language Resources and Evaluation (LREC). (2000) 921–924

[60] Shlafman, S., Tal, A., Katz, S.: Metamorphosis of polyhedral surfaces using decomposition (2002)

[61] Golovinskiy, A., Funkhouser, T.: Randomized cuts for 3D mesh analysis. ACM Transactions on Graphics (Proc. SIGGRAPH ASIA) **27** (2008)

[62] Garland, M., Willmott, A., Heckbert, P.S.: Hierarchical face clustering on polygonal surfaces. In: I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics, New York, NY, USA, ACM (2001) 49–58

[63] Gelfand, N., Guibas, L.J.: Shape segmentation using local slippage analysis. In: SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, New York, NY, USA, ACM (2004) 214–223

[64] Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical mesh segmentation based on fitting primitives. Vis. Comput. **22** (2006) 181–193

[65] Lai, Y.K., Hu, S.M., Martin, R.R., Rosin, P.L.: Fast mesh segmentation using random walks. In: Symposium on Solid and Physical Modeling. (2008) 183–191

[66] Katz, S., Leifman, G., Tal, A.: Mesh segmentation using feature point and core extraction. Visual Computer (2005)

[67] Mortara, M., Patane, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Blowing bubbles for the multi-scale analysis and decomposition of triangle meshes. Algorithmica (Special Issues on Shape Algorithms) **38** (2004) 227–248

[68] Liu, R., Zhang, H.: Segmentation of 3d meshes through spectral clustering. In: PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference, Washington, DC, USA, IEEE Computer Society (2004) 298–305

[69] Lin, H.Y.S., Liao, H.Y.M., Lin, J.C.: Visual salience-guided mesh decomposition. IEEE Transactions on Multimedia **9** (2007) 46–57

[70] Agathos, A., Pratikakis, I., Perantonis, S., Sapidis, N., Azariadis, P.: 3d mesh segmentation methodologies for cad applications. **4** (2007) 827–841

[71] Shamir, A.: A survey on mesh segmentation techniques. Computer Graphics Forum **28** (2008) 1539–1556

[72] Mortara, M., Patanè, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. In: ACM symposium on Solid modeling and applications, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association (2004) 339–344

[73] Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: Labelme: A database and web-based tool for image annotation. Int. J. Comput. Vision **77** (2008) 157–173

[74] Correia, P., Pereira, F.: Objective evaluation of relative segmentation quality. In: ICIP00. (2000) Vol I: 308–311

[75] Everingham, M., Muller, H., Thomas, B.T.: Evaluating image segmentation algorithms using the pareto front. In: ECCV '02: Proceedings of the 7th European

Conference on Computer Vision-Part IV, London, UK, Springer-Verlag (2002) 34–48

[76] Huang, Q., Dom, B.: Quantitative methods of evaluating image segmentation. In: ICIP '95: Proceedings of the 1995 International Conference on Image Processing (Vol. 3)-Volume 3, Washington, DC, USA, IEEE Computer Society (1995) 3053

[77] Unnikrishnan, R., Pantofaru, C., Hebert, M.: A measure for objective evaluation of image segmentation algorithms. In: Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05), Workshop on Empirical Evaluation Methods in Computer Vision. Volume 3. (2005) 34 – 41

[78] Zhang, Y.: A survey on evaluation methods for image segmentation. PR **29** (1996) 1335–1346

[79] Zhang, H., Fritts, J.E., Goldman, S.A.: Image segmentation evaluation: A survey of unsupervised methods. Comput. Vis. Image Underst. **110** (2008) 260–280

[80] Benhabiles, H., Vandeborre, J., Lavoue, G., Daoudi, M.: A framework for the objective evaluation of segmentation algorithms using a ground-truth of human segmented 3d-models. In: Shape Modeling International. (2009)

[81] Giorgi, D., Biasotti, S., Paraboschi, L.: SHREC:SHape REtrieval Contest: Watertight models track, http://watertight.ge.imati.cnr.it/ (2007)

[82] Shilane, P., Min, P., Kazhdan, M., Funkhouser, T.: The princeton shape benchmark. In: Shape Modeling International, Washington, DC, USA, IEEE Computer Society (2004) 167–178

[83] Robbiano, F., Attene, M., Spagnuolo, M., Falcidieno, B.: Part-based annotation of virtual 3d shapes. In: CW '07: Proceedings of the 2007 International Conference on Cyberworlds, Washington, DC, USA, IEEE Computer Society (2007) 427–436

[84] Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., Seidel, H.P.: Intelligent mesh scissoring using 3D snakes. In: 12th Pacific Conference on Computer Graphics and Applications (PG). (2004)

[85] Wu, H.Y., Pan, C., Pan, J., Yang, Q., Ma, S.: A sketch-based interactive framework for real-time mesh segmentation. In: Computer Graphics International. (2007)

[86] Rand, W.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association **66** (1971) 846–850

[87] Shapira, L., Shamir, A., Cohen-Or, D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis. Comput. **24** (2008) 249–259

[88] Chazelle, B., Dobkin, D., Shourhura, N., Tal, A.: Strategies for polyhedral surface decomposition: An experimental study. Computational Geometry: Theory and Applications **7** (1997) 327–342

[89] Kraevoy, V., Julius, D., Sheffer, A.: Shuffler: Modeling with interchangeable parts. In: Pacific Graphics. (2007)

[90] Johnson, A.: Surface landmark selection and matching in natural terrain. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Volume 2. (2000) 413–420 Using saliency in choosing spin images.

[91] Zhang, H., Sheffer, A., Cohen-Or, D., Zhou, Q., van Kaick, O., Tagliasacchi, A.: Deformation-driven shape correspondence. Comput. Graph. Forum **27** (2008) 1431–1439

[92] Funkhouser, T., Shilane, P.: Partial matching of 3d shapes with priority-driven search. In: Symposium on Geometry Processing. (2006)

[93] Alexa, M.: Merging polyhedral shapes with scattered features. Visual Computer **16** (2000) 26–37

[94] Kraevoy, V., Sheffer, A.: Cross-parameterization and compatible remeshing of 3d models. ACM Transactions on Graphics (Proc SIGGRAPH) **23** (2004) 861–869

[95] Zhang, E., Mischaikow, K., Turk, G.: Feature-based surface parameterization and texture mapping. ACM Transactions on Graphics **24** (2005)

[96] Sumner, R., Popovic, J.: Deformation transfer for triangle meshes. ACM Transactions on Graphics (Proc SIGGRAPH) **23** (2004) 399–405

[97] Xu, K., Zhang, H., Tagliasacchi, A., Liu, L., Li, G., Meng, M., Xiong, Y.: Partial intrinsic reflectional symmetry of 3d shapes. ACM Transactions on Graphics, (Proceedings SIGGRAPH Asia 2009) **28** (2009) to appear

[98] Zhou, Y., Huang, Z.: Decomposing polygon meshes by means of critical points. In: MMM. (2004) 187–195

[99] Simpson, J.: Oxford English Dictionary, Second Edition. Oxford University Press (1989) http://dictionary.oed.com.

[100] Schelling, T.: The Strategy of Conflict. Harvard University Press (1960)

[101] Parker, P.: Webster's On-line Dictionary: The Rosetta Edition. http://www.websters-online-dictionary.org (2011)

[102] van Kaick, O., Zhang, H., Hamarneh, G., Cohen-Or, D.: A survey on shape correspondence. In: Proc. of Eurographics State-of-the-art Report. (2010)

[103] Castellani, U., Cristani, M., Fantoni, S., Murino, V.: Sparse points matching by combining 3d mesh saliency with statistical descriptors. Computer Graphics Forum **27** (2008) 643–652

[104] Lipman, Y., Funkhouser, T.: Mobius voting for surface correspondence. ACM Transactions on Graphics (SIGGRAPH 2009) **28** (2009)

[105] Sun, J., Ovsjanikov, M., Guibas, L.: A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. In: Computer Graphics Forum. Volume 28., Wiley Online Library (2009) 1383–1392

[106] Zaharescu, A., Boyer, E., Varanasi, K., Horaud, R.: Surface feature detection and description with applications to mesh matching. In: CVPR. (2009)

[107] Li, X., Guskov, I.: Multi-scale features for approximate alignment of point-based surfaces. In: Symposium on Geometry Processing. (2005)

[108] Li, X., Guskov, I.: 3d object recognition from range images using pyramid matching. In: Workshop on 3D Representation for Recognition (3dRR). (2007)

[109] Novotni, M., Degener, P., Klein, R.: Correspondence generation and matching of 3d shape subparts. Technical Report CG-2005-2, Universität Bonn (2005)

[110] Schlattmann, M., Degener, P., Klein, R.: Scale space based feature point detection on surfaces. Journal of WSCG **16** (2008)

[111] Sonthi, R., Kunjur, G., Gadh, R.: Shape feature determination using the curvature region representation. In: Proc. Solid Modeling, ACM (1997)

[112] Hisada, M., Belyaev, A., Kunii, T.: A skeleton-based approach for detection of perceptually salient features on polygonal surfaces. Computer Graphics Forum **21** (2002) 689–700

[113] Chua, C., Jarvis, R.: Point signatures: A new representation for 3D object recognition. International Journal of Computer Vision **25** (1996) 63–85

[114] Shilane, P., Funkhouser, T.: Distinctive regions of 3d surfaces. ACM Transactions on Graphics **26** (2007)

[115] Bronstein, A., Bronstein, M., Bustos, B., Castellani, U., Crisani, M., Falcidieno, B., Guibas, L., Kokkinos, I., Murino, V., Ovsjanikov, M., Patane, G., Sipiran, I.,

Spagnuolo, M., Sun, J.: SHREC 2011: robust feature detection and description benchmark. In: Eurographics Workshop on 3D Object Retrieval. (2010)

[116] Gal, R., Cohen-Or, D.: Salient geometric features for partial shape matching and similarity. ACM Transaction on Graphics (2006)

[117] Ko, B., Nam, J.: Object-of-interest image segmentation based on human attention and semantic region clustering. J Opt Soc Am A Opt Image Sci Vis **23** (2006) 2462–2470

[118] Moreels, P., Perona, P.: Evaluation of features detectors and descriptors based on 3d objects. IJCV **73** (2007) 263–284

[119] Schmid, C., Mohr, R., Bauckhage, C.: Evaluation of interest point detectors. IJCV **37** (2000) 151–172

[120] Stark, M., Schiele, B.: How good are local features for classes of geometric objects. (2007) 1–8

[121] Zuliani, M., Kenney, C., Manjunath, B.: A mathematical comparison of point detectors. In: Computer Vision and Pattern Recognition Workshop. (2004) 172

[122] Sebe, N., Lew, M.: Comparing salient point detectors. Pattern Recognition Letters **24** (2003) 89–96

[123] Huang, T., Cheng, K., Chuang, Y.: A collaborative benchmark for region of interest detection algorithms. (2009) 296–303

[124] von Ahn, L., Dabbish, L.: Designing games with a purpose. Commun. ACM **51** (2008) 58–67

[125] Milanes, R., Wechsler, H., Gil, S., Bost, J., Pun, T.: Integration of bottom-up and top-down cues for visual attention using non-linear relaxation. IEEE Computer Vision and Pattern Recognition (1994) 781–785

[126] Tsotsos, J., Culhane, S., Wai, W., Lai, Y., Davis, N., Nuflo, F.: Modeling visual-attention via selective tuning. Artificial Intelligence **78** (1995) 507–545

[127] Itti, L., Koch, C., Neibur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 1254–1259

[128] Rosenholtz, R.: A simple saliency model predicts a num ber of motion popout phenomena. Vision Research **39** (1999) 3157–3163

[129] Santella, A., DeCarlo, D.: Robust clustering of eye movement recordings for quantification of visual interest. In: Eye Tracking Research and Applications (ETRA). (2004) 27–34

[130] Lewis, D.: Convention: A Philosophical Study. Harvard University Press (1969)

[131] Cole, F., Sanik, K., DeCarlo, D., Finkelstein, A., Funkhouser, T., an d Manish Singh, S.R.: How well do line drawings depict shape? ACM Transactions on Graphics (Proc. SIGGRAPH) **28** (2009)

[132] Bronstein, A., Bronstein, M., Kimmel, R.: Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching. Proceedings of the National Academy of Science (2006) 1168–1172

[133] Rusinkiewicz, S.: Estimating curvatures and their derivatives on triangle meshes. In: Symposium on 3D Data Processing, Visualization, and Transmission. (2004)

[134] Breiman, L.: Random forests. Machine Learning **45** (2001) 5–32

[135] Witten, I.H., Frank, E.: Data mining: Practical machine learning tools and techniques, 2nd edition. (2005)

[136] Kim, V., Lipman, Y., Funkhouser, T.: Blended intrinsic maps. ACM Transactions on Graphics (SIGGRAPH 2011) **30** (2011)

[137] Heer, J., Bostock, M.: Crowdsourcing graphical perception: Using Mechanical Turk to assess visualization design. In: ACM Human Factors in Computing Systems (CHI). (2010) 203–212

[138] Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: Proceedings of SIGGRAPH 1997. Computer Graphics Proceedings, Annual Conference Series (1997) 209–216

[139] Mertens, T., Kautz, J., Chen, J., Durand, F.: Texture transfer using geometry correlation. In: Eurographics Symposium on Rendering. (2006)

[140] Golovinskiy, A., Matusik, W., Pfister, H., Rusinkiewicz, S., Funkhouser, T.: A statistical model for synthesis of detailed facial geometry. ACM Trans. Graph. (Proc. SIGGRAPH) **25** (2006)

[141] Allen, B., Curless, B., Popović, Z.: The space of human body shapes: reconstruction and parameterization from range scans. ACM Trans. Graph. (SIGGRAPH) **22** (2003) 587–594

[142] Nguyen, M.X., Yuan, X., Chen, B.: Geometry completion and detail generation by texture synthesis. The Visual Computer (Special Issue for Pacific Graphics 2005) **21** (2005) 669–678

[143] Bhat, P., Ingram, S., Turk, G.: Geometric texture synthesis by example. In: Proc. Eurographics/ACM SIGGRAPH Symposium on Geometry Processing. (2004) 41–44

[144] Alexa, M.: Recent advances in mesh morphing. Computer Graphics Forum **21** (2002) 173–198

[145] Wei, L.Y., Levoy, M.: Texture synthesis over arbitrary manifold surfaces. In: SIGGRAPH '01: Proc. 28th annual conf. on Comp. graph. and interactive techniques. (2001) 355–360

[146] Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: SIGGRAPH '01: Proc. 28th annual conf. Comp. graph. and interactive techniques. (2001) 327–340

[147] Praun, E., Hoppe, H.: Spherical parametrization and remeshing. ACM Trans. Graph. **22** (2003) 340–349

[148] Sheffer, A., Gotsman, C., Dyn, N.: Robust spherical parameterization of triangular meshes. Computing **72** (2004) 185–193

[149] Hormann, K., Lévy, B., Sheffer, A.: Mesh parameterization: Theory and practice. In: SIGGRAPH 2007 Course Notes. Number 2, San Diego, CA, ACM Press (2007) vi+115

[150] Schreiner, J., Asirvatham, A., Praun, E., Hoppe, H.: Inter-surface mapping. In: ACM SIGGRAPH 2004 Papers. (2004) 870–877

[151] Bronstein, A.M., Bronstein, M.M., Kimmel, R.: Efficient computation of isometry-invariant distances between surfaces. SIAM J. Scientific Computing (2006)

[152] Ovsjanikov, M., Mérigot, Q., Mémoli, F., Guibas, L.: One point isometric matching with the heat kernel. In: Eurographics Symposium on Geometry Processing (SGP). (2010)

[153] Wang, S., Gu, X., Qin, H.: Automatic non-rigid registration of 3d dynamic data for facial expression synthesis and transfer. In: CVPR 2008. (2008) 1–8

[154] Dinh, H., Yezzi, A., Turk, G.: Texture transfer during shape transformation. ACM Trans. Graphics **24** (2005)

[155] Heeger, D.J., Bergen, J.R.: Pyramid-based texture analysis/synthesis. In: SIGGRAPH '95: Proc. 22nd annual conf. Comp. graph. and interactive techniques. (1995) 229–238

[156] Breckon, T.P., Fisher, R.B.: 3D surface relief completion via non-parametic techniques. IEEE Trans. Pattern Analysis and Machine Intelligence **30** (2008) 2249 – 2255

[157] Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: IEEE Int'l. Conf. on Comp. Vis., Corfu, Greece (1999)

[158] Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture optimization for example-based synthesis. ACM Transactions on Graphics, SIGGRAPH 2005 (2005)

[159] Wei, L., Han, J., Zhou, K., Bao, H., Guo, B., Shum, H.: Inverse texture synthesis. In: ACM Transactions on Graphics (TOG). Volume 27., ACM (2008) 52

[160] Han, J., Zhou, K., Wei, L., Gong, M., Bao, H., Zhang, X., Guo, B.: Fast example-based surface texture synthesis via discrete optimization. The Visual Computer **22** (2006) 918–925

[161] Lu, J., Georghiades, A.S., Glaser, A., Wu, H., Wei, L.Y., Guo, B., Dorsey, J., Rushmeier, H.: Context-aware textures. ACM Transactions on Graphics **26** (2007)

[162] Simakov, D., Caspi, Y., Shechtman, E., Irani, M.: Summarizing visual data using bidirectional similarity. In: CVPR. (2008)

[163] Turk, G.: Texture synthesis on surfaces. In: SIGGRAPH '01: Proc. 28th annual conf. Comp. graph. and interactive techniques. (2001) 347–354

[164] Kwatra, V., Adalsteinsson, D., Kim, T., Kwatra, N., Carlson, M., Lin, M.: Texturing fluids. IEEE Trans. Visualization and Computer Graphics **13** (2007) 939–952

[165] Xu, K., Cohen-Or, D., Ju, T., Liu, L., Zhang, H., Zhou, S., Xiong, Y.: Feature-aligned shape texturing. ACM Transactions on Graphics, (Proc. SIGGRAPH Asia 2009) **28** (2009)

[166] Dey, T.K., Li, K., Luo, C., Ranjan, P., Safa, I., Wang, Y.: Persistent heat signature for pose-oblivious matching of incomplete models. Computer Graphics Forum (Symposium on Geometry Processing) (2010)

[167] Fu, H., Cohen-Or, D., Dror, G., Sheffer, A.: Upright orientation of man-made objects. ACM Trans. Graph. **27** (2008)

[168] Zelinka, S., Garland, M.: Similarity-based surface modelling using geodesic fans. In: SGP '04: Proc. 2004 Eurographics/ACM SIGGRAPH Symp. on Geom. Proc. (2004) 204–213

[169] Barnes, C., Shechtman, E., Goldman, D.B., Finkelstein, A.: The generalized PatchMatch correspondence algorithm. In: European Conference on Computer Vision. (2010)

[170] Little, G., Chilton, L., Goldman, M., Miller, R.: Turkit: human computation algorithms on mechanical turk. In: Proceedings of the 23nd annual ACM symposium on User interface software and technology, ACM (2010) 57–66