Visual Comput (2008)
DOI 10.1007/s00371-008-0226-z

ORIGINAL ARTICLE

Chunxia Xiao*
Hongbo Fu
Chiew-Lan Tai

Hierarchical aggregation for efficient shape extraction

C. Xiao (✉)
Computer School, Wuhan University,
430072 Wuhan, P.R. China
cxxiao@whu.edu.cn

H. Fu · C.-L. Tai
Department of Computer Science &
Engineering, Hong Kong University of
Science & Technology,
Hong Kong
{fuhb, taicl}@cse.ust.hk

Abstract This paper presents an efficient framework which supports both automatic and interactive shape extraction from surfaces. Unlike most of the existing hierarchical shape extraction methods, which are based on computationally expensive top-down algorithms, our framework employs a fast bottom-up hierarchical method with multiscale aggregation. We introduce a geometric similarity measure, which operates at multiple scales and guarantees that a hierarchy of high-level features are automatically found through local adaptive aggregation. We also show that the aggregation process allows easy incorporation of user-specified constraints, enabling users to interactively extract features of interest. Both our automatic and the interactive shape extraction methods do not require explicit connectivity information, and thus are applicable to unorganized point sets. Additionally, with the hierarchical feature representation, we design a simple and effective method to perform partial shape matching, allowing efficient search of self-similar features across the entire surface. Experiments show that our methods robustly extract visually meaningful features and are significantly faster than related methods.

Keywords Mesh segmentation · Geometric similarity measure · Shape matching · Shape extraction

1 Introduction

Automatic extraction of visually meaningful pieces from surfaces is an important step towards object recognition. Since features are usually of different sizes, hierarchical/multiscale shape extraction methods are desirable. Most existing hierarchical methods (see [2, 30] and references therein) analyze objects in a *top-down* manner and focus on finding an ultimate partition of a surface, i.e. a division of the surface into non-overlapping features. These methods generally need to compute all pairwise global similarity measures between vertices or faces of the entire surface. Hence, they are computationally expensive and their time complexities are $\Omega(N^2)$, where N denotes the number of vertices or faces in the object.

In contrast, *bottom-up* approaches using aggregation operations are efficient, as they only involve computation of local similarity measures. However, extracting global features through local aggregation is generally challenging [7]. We solve this problem by presenting a bottom-up framework using adaptive aggregation with a multiscale similarity measure. Although our framework cannot guarantee a partition of the surface, we show that it enables easy integration of both automatic and interactive shape extraction into the same framework. We also demonstrate that our framework has advantages for some certain ap-

*This work was done when Chunxia Xiao was a postdoctoral research fellow at Hong Kong University of Science & Technology.

plications, in particular partial shape matching. These algorithms are efficient, with linear time complexity in the number of vertices.

Our automatic feature-extraction method is inspired by Sharon et al.'s work on image segmentation using adaptive aggregation [34]. However, adapting this algorithm from the two-dimensional domain to three-dimensional (3D) surfaces is not straightforward (see the discussion in Sect. 3). We bridge the gap by introducing a new geometric similarity measure which operates at multiple scales. Our method automatically builds an input surface into a hierarchy of aggregates through adaptive aggregation, weighted by the multiscale similarity measure. This similarity measure employs the accumulated statistics within each aggregate, thus making the aggregation process insensitive to small fluctuations or (locally) repeated patterns in surfaces. Its adaptivity nature guarantees that the statistics of different aggregates are not wrongly accumulated across the aggregate boundaries. After a hierarchy of aggregates is built, we use an energy formulation derived from a normalized cut [35] to automatically identify at all levels the visually meaningful features of different sizes (see an example in Fig. 1).

Since whether or not a shape-extraction result is desirable is somewhat subjective and application oriented, the results of fully automatic extraction algorithms, including ours, may not always be desirable. User interaction is needed to aid the segmentation process towards yielding desirable results. We present an interactive tool to extract features of interest. Our tool allows the user to casually draw a small set of strokes to give hints on which vertices should be included in or excluded from a desired feature. We design a constrained weighted aggregation algorithm which produces visually meaningful aggregates respecting the user's intention. Our interactive tool is fast, providing instant visual feedback even for large-scale models.

Partial shape matching is a fundamental tool for many geometric applications, such as self-shape matching and partial shape retrieval. Based on our multiscale similarity measure and energy function for feature identification, we introduce a simple and effective partial shape matching approach. The hierarchical structure allows us to search matching subparts across the surface efficiently. The initialization time of our approach is significantly shorter than that of the state-of-the-art method proposed by Gal and Cohen-Or [6].

Both our automatic and interactive shape extraction methods do not require explicit connectivity information. Therefore, besides irregular meshes, we adapt these methods to unorganized point sets.

2 Related work

Mesh-segmentation research has been active for a long time and many techniques have been proposed (see the latest surveys in [2] and [30]). Existing techniques can be categorized into two main groups. One group segments a mesh into patches that best fit a given set of primitives, e.g. planes, spheres and cylinders [1]. The other group aims to obtain visually meaningful pieces. Our algorithm falls into the latter group, which we briefly review here.

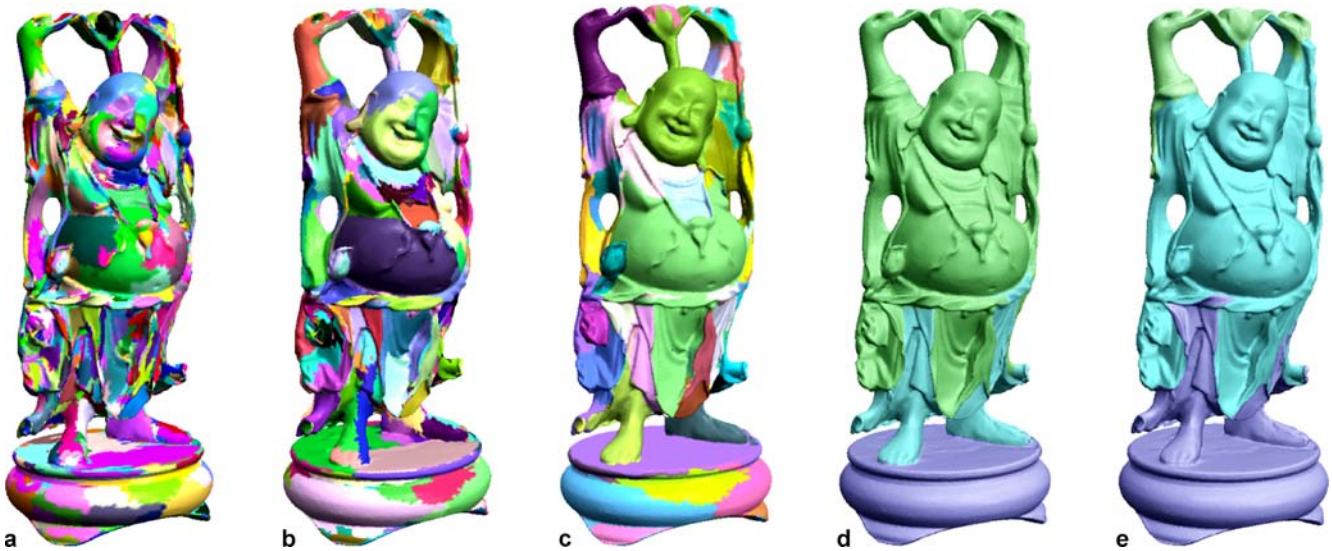

Fig. 1a–e. Our method efficiently builds a bottom-up hierarchy of aggregates through multiscale aggregation. **a**, **b** and **c** Aggregates at levels 6, 7 and 9, respectively. Aggregates with low energy at every level of the hierarchy are then automatically detected as visually meaningful features. **d** and **e** Features (in *blue*) at levels 11 and 13, respectively

Automatic surface segmentation. Most state-of-the-art mesh-segmentation techniques are based on clustering algorithms. By defining a similarity measure based on geodesic distances and dihedral angles between all pairs of mesh faces, Katz and Tal [14] used k-means clustering and fuzzy clustering to segment meaningful components. This method was later extended to perform pose-invariant segmentation in [13] by choosing k feature points instead of k random points [14] as clustering seeds. The time complexity and the memory cost of computing all pairwise geodesic distances are high, making these methods prohibitive for large-scale models. For large models, k-means clustering can be performed on a simplified coarse mesh [14] or a hierarchy of feature-sensitive meshes [15]. Mortara et al. [24] presented a plumber algorithm that segments a surface into connected components that are either body parts or elongated features, and this segmentation method can be done at single or multiscale. Lai et al. [15] proposed to use statistical quantities of local surface characteristics for clustering, allowing more meaningful segmentation of surfaces with small noisy shapes or geometric textures.

Liu and Zhang [21] adapted spectral clustering to mesh segmentation. They used a similarity measure based on pairwise geodesic distances and minimized a normalized cost associated with a cutting of a mesh. Their method suffers from a similar performance problem as the work of Katz and Tal [14]. Later, Liu et al. [20] accelerated that method using the Nyström approximation method. Liu and Zhang [22] reduced the mesh-segmentation problem to efficient contour analysis by spectrally projecting a mesh to a plane. Yamazaki et al. [38] presented a method based on spectral clustering to segment unorganized point sets.

Performing clustering algorithms only once generally does not give desirable segmentations. Therefore, existing segmentation methods usually produce a *top-down* hierarchical segmentation by recursively applying a clustering algorithm to each segmented part. This leads to a partition of the surface, with each part corresponding to a feature. Another main category of mesh segmentation is to build the hierarchy of features from the bottom up. The earlier bottom-up techniques segment a mesh into regions through growing regions from certain sources [23, 26, 36, 41]. However, due to its local nature, region growing easily leads to over-segmentation and is sensitive to noise. Garland et al. [7] built a bottom-up hierarchy of aggregates, which are, however, best-fit planar patches rather than meaningful features. By combining curvature maps and graph cuts in a multiscale framework, Gatzke and Grimm [8] presented a multiscale approach to extract meaningful features.

Interactive surface segmentation. Unlike automatic segmentation, interactive segmentation techniques generally aim to extract a feature of interest from a whole surface through simple user interaction, that is, to partition an input surface into two parts: foreground and background.

There are two main categories of user interfaces for interactive feature extraction: boundary-based and region-based. In boundary-based methods, the user draws strokes near the cutting boundary of a desired feature. Finding the final cutting boundary is equivalent to solving a closed shortest path problem constrained by the user-drawn strokes [5, 17, 18, 31]. With region-based methods, the user draws strokes indicating the inside or outside of a region of interest. Ji et al. [12] used a watershed algorithm [26] with a new feature-sensitive metric to grow the regions from the user-specified strokes. Zelinka and Garland [40] presented an interactive region-based algorithm by computing minimum-cost graph cuts and accelerated it in a multiresolution framework.

Our interactive feature-extraction tool is region-based. Unlike the most closely related work by Ji et al. [12], whose feature metric does not operate at multiple scales, our multiscale similarity measure leads to more intuitive segmentation results, especially for regions with small fluctuations or geometric textures (see a comparison example in Fig. 7).

Local shape matching. How to measure similarity between local shapes is crucial in local shape matching. Many local shape comparison methods have been proposed (e.g. based on curvature matching at multiple scales [9, 39] and signature vectors derived from salient feature points [19]). More recently, Gal and Cohen-Or [6] defined a set of sparse local surface descriptors based on surface curvature analysis, to capture the essence of the mesh geometry, and used the descriptors for partial shape matching of surfaces.

3 Automatic geometric shape extraction

In this section, we present our efficient method for automatic shape extraction from meshes. It is inspired by Sharon et al.'s idea of image segmentation based on weighted aggregation [34]. Their method first adaptively assembles pixels into small aggregates according to luminance resemblance, and then assembles the small aggregates. To extract perceptually more meaningful features, besides resemblance at pixel level, they introduced multiscale similarity measures between aggregates. At every level of a pyramid, aggregates whose associated cutting costs have low values are detected as the salient segments.

However, to extend the above image-segmentation method to 3D meshes, we need to address two main issues. First, the definitions of similarity measures for images and surfaces are very different. Vertex positions of surfaces,

which are the counterpart of pixel colors or luminance values of images, are not sufficient to define an effective geometric similarity measure. We need additional shape descriptors defined on 3D surfaces, such as curvatures and normals. Second, unlike images, meshes are irregular both in connectivity and sampling.

3.1 Geometric similarity measurement

We introduce a geometric similarity measure between adjacent vertices. We regard a mesh to be segmented as a graph $G = (V, E)$, with $V = \{v_i | 1 \le i \le n\}$ denoting its set of n nodes, each corresponding to a mesh vertex, and E being the set of undirected edges $\{e_{ij} = (v_i, v_j)\}$, each corresponding to a mesh edge.

Our geometric similarity measure is defined in terms of two shape deviation metrics which indicate the differences in local shapes. Since curvature is an important geometric measure of local shapes, we define the first deviation metric as the curvature contrast between neighboring vertices v_i and v_j:

$$\kappa(v_i, v_j) = |\kappa_H(v_i) - \kappa_H(v_j)|,$$

where $\kappa_H(v)$ denotes the mean curvature at vertex v. We choose to use the curvature flow operator [4] to evaluate the normalized mean curvature: $\kappa_H(v_i) = |\delta(v_i)|$, with

$$\delta(v_i) = \frac{1}{\sum_{j\in N(i)}(\cot\alpha_j + \cot\beta_j)} \times \sum_{j\in N(i)} (\cot\alpha_j + \cot\beta_j)(v_i - v_j),$$

where $N(i)$ denotes the 1-ring neighboring node indices of node v_i, and α_j and β_j are the two angles opposite the edge e_{ij}. To achieve a more robust curvature estimation for the vertices on noisy or poorly triangulated meshes, we may also exploit the normal cycle method [3] or the curvature map method [9].

Directional curvatures are important to distinguish orientations of shapes. To determine the boundary of the surface segmentation components, we define the second shape deviation metric based on the surface creases, that is, we define this metric based on the convexities and normal contrast of adjacent vertices:

$$\xi(v_i, v_j) = 1 + \lambda \cdot \text{concave}(v_i, v_j) |\arccos(n_i \cdot n_j)|,$$

where $\lambda \in [0, 1]$ ($\lambda = 0.5$ in all our experiments), n_i and n_j are the unit normals at v_i and v_j, respectively, and $\text{concave}(v_i, v_j) = -1$ if both v_i and v_j are convex vertices or both are concave vertices, and 1 otherwise. A vertex v_i is regarded as a convex vertex if $\delta(v_i) \cdot n_i \ge 0$, and concave otherwise. We compute the normal n_i as the average normals of faces adjacent to v_i, weighted by the corresponding face areas.

Based on the above two shape deviation metrics, we define a *similarity measure* between the end points of an edge e_{ij} as $\omega_{ij} = \exp(-\alpha \cdot \text{weight}(v_i, v_j))$, with

$$\text{weight}(v_i, v_j) = \eta \frac{\kappa(v_i, v_j)}{\mu_\kappa} + (1 - \eta)\frac{\xi(v_i, v_j)}{\mu_\xi},$$

where α and η are two positive parameters (we use $\alpha = 0.5$ and $\eta = 0.6$ in all our experiments) and μ_κ and μ_ξ are the means of $\{\kappa(v_i, v_j)\}$ and $\{\xi(v_i, v_j)\}$ between all the adjacent vertices in the surface, respectively. Through dividing $\kappa(v_i, v_j)$ and $\xi(v_i, v_j)$ by their means, we alleviate the potential problem due to irregular sampling. Figure 2 visualizes our similarity measure between adjacent vertices.

3.2 Shape extraction with adaptive aggregation

We now describe an adaptive vertex aggregation process to segment a graph G weighted by the similarity measures $\{\omega_{ij}\}$. Similar to [34], we use a normalized energy associated with a cutting of the graph into two subgraphs to identify the meaningful features:

$$\Gamma(u) = \frac{\sum_{i>j} \omega_{ij}(u_i - u_j)^2}{\sum_{i>j} \omega_{ij} u_i u_j} = \frac{u^{\mathrm{T}} L u}{\frac{1}{2} u^{\mathrm{T}} W u}, \quad (1)$$

where $u = (u_1, \ldots, u_n)^{\mathrm{T}}$ is a vector of state variables, W is the affinity matrix, with $W_{ij} = \omega_{ij}$ if v_i and v_j are adjacent and $W_{ij} = 0$ otherwise, and $L = D - W$ is the Laplacian matrix of the graph, with $D = \text{diag}(d_1, d_2, \ldots, d_n)$, $d_i = \sum_j \omega_{ij}$. Any Boolean assignment of u that yields a low-energy value $\Gamma(u)$ corresponds to a salient segment S in the mesh: the vertices $i \in \{1, 2, \ldots, n\}$ for which $u_i = 1$ are the vertices in S; $u_i = 0$ means not in S.

Liu and Zhang [21] minimized an energy similar to $\Gamma(u)$ through global optimization, which is recursively

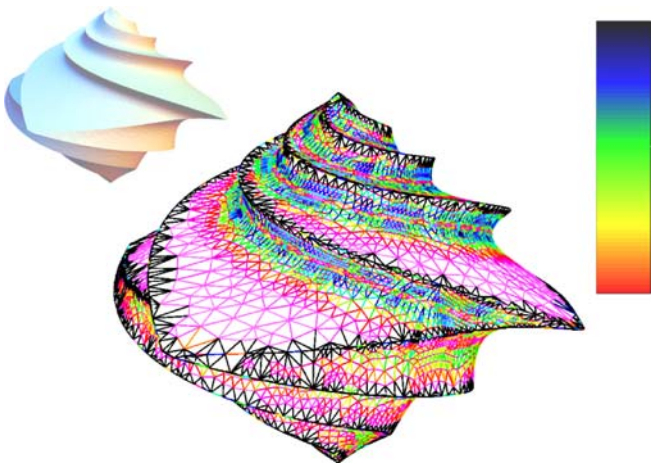

Fig. 2. Visualization of geometric similarity computed between adjacent vertices (*red*: high similarity, *black*: low similarity)

performed to obtain a top-down hierarchical segmentation. This process, however, is computationally expensive. Instead of directly minimizing the energy $\Gamma(u)$, we present a fast approximate method, based on algebraic multigrid solvers.

We use an iterative process of adaptive vertex aggregation. We start by choosing a subset of vertices from the original graph, denoted by $C = \{c_k | 1 \le k \le N\}$, to constitute a coarse graph (usually $N \in [\frac{n}{4}, \frac{n}{2}]$). These N vertices, called seeds, are chosen such that every vertex v in the original graph is strongly coupled (that is, similar in terms of ω_{ij}) to at least one seed adjacent to v. We adopt a method similar to [32] to generate the seeds, as follows. We first order the nodes (vertices or aggregates) by the area they represent. We select the first node with the largest area to be a seed. Then, we scan nodes according to this order and check their degree of coupling to the previously selected nodes. Whenever we encounter a node that is weakly coupled to the selected nodes, we add that node to the list of seeds.

Then, the state variables u can be approximately represented through $u = PU$, where $U = (U_1, U_2, \ldots, U_N)^{\mathrm{T}}$ denotes a vector of state variables corresponding to C and P is a sparse matrix $\{p_{ij}\}_{n \times N}$. For simplicity of notation, we assume that the vertices in C correspond to the first N vertices of V; then $U_k = u_k$, $k = 1, 2, \ldots, N$, and

$$p_{ik} = \begin{cases} \omega_{ik} / \sum_j \omega_{ij} & \text{if } i > N, \\ 0 & \text{if } 1 \le i \le N \text{ and } i \ne k, \\ 1 & \text{if } 1 \le i \le N \text{ and } i = k. \end{cases}$$

Substituting u in Eq. 1 by PU, we obtain the energy to be minimized at the coarse level as

$$\Gamma(U) \simeq \frac{U^{\mathrm{T}}(P^{\mathrm{T}}LP)U}{\frac{1}{2}U^{\mathrm{T}}(P^{\mathrm{T}}WP)U} = \frac{\sum_{k>l} \widetilde{\omega}_{kl}(U_k - U_l)^2}{\sum_{k>l} \widehat{\omega}_{kl} U_k U_l}, \qquad (2)$$

where the weights $\widetilde{\omega}_{kl}$ and $\widehat{\omega}_{kl}$ are derived from $P^{\mathrm{T}}LP$ and $P^{\mathrm{T}}WP$, respectively. $\Gamma(U)$ can be approximated by replacing $\widetilde{\omega}_{kl}$ with $\widehat{\omega}_{kl}$ [34]. In the approximate representation of $\Gamma(U)$, $\widehat{\omega}_{kl}$ acts as a similarity measure between neighboring aggregates k and l, like the similarity measure ω_{ij} between neighboring vertices i and j at the finest level. Note that $\widehat{\omega}_{kl}$ here only operates at the current coarse level; in Sect. 3.3 we will extend it to operate at multiple scales. This coarsening procedure is repeated level after level. Figures 1 and 3 show the aggregates at different levels for the Buddha and Moai models, respectively.

To identify the salient features at every level of the aggregate hierarchy, we evaluate the value of $\Gamma(U)$. Specifically, for the kth aggregate at a particular level, we compute $\Gamma(U)$ by setting the kth entry of U to 1 and the rest of the entries to zero. An aggregate S is then regarded as salient if and only if both the following conditions are satisfied:

- its associated energy $\Gamma(U)$ is below a given threshold,
- Area$(S) \in [s_1, s_2]$, where Area(S) is the area of S relative to the total area of the mesh, and $0 < s_1 < s_2 < 1$ ($s_1 = 0.01$ and $s_2 = 0.5$ by default).

The second condition has two purposes. First, it avoids the trivial solution $u = (1, 1, \ldots, 1)$ (i.e. Area$(S) = 1$) of having only one segment composing the entire model. Second, it avoids tiny aggregates at low levels of aggregation, which are too small (i.e. Area$(S) < s_1$) to be salient features.

3.3 Multiscale similarity measure

The vertex aggregation process we have described above is based on the weights computed solely from geometric properties of the finest graph. In this subsection, we show that statistical measurement computed on the newly formed aggregates can be used to improve feature-extraction results.

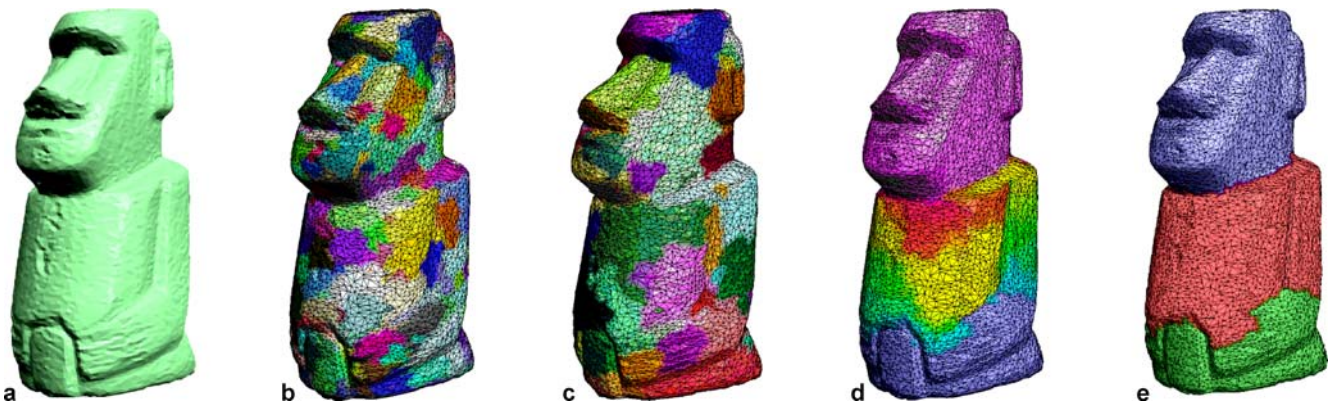

Fig. 3a–e. Examples of shape extraction with multiscale aggregation. **b**, **c** and **d** Aggregates at levels 5, 6 and 7, respectively. **e** A meaningful feature (in *blue*) detected at level 7 and the corresponding fuzzy region (in *red*)

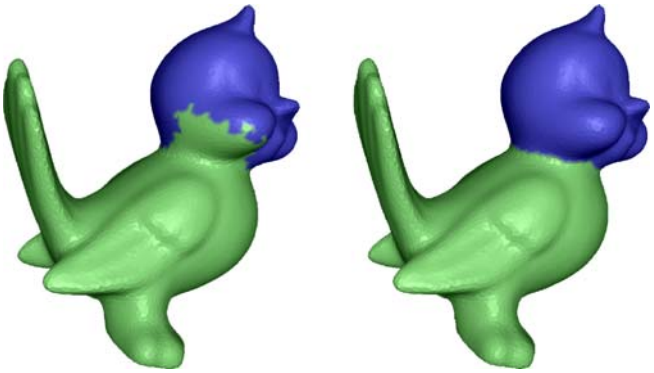

Fig. 4. Feature extraction with the similarity measure defined only at vertex levels (*left*) and with multiscale similarity measures (*right*)

Statistics of curvature are important for recognizing patches of consistent geometry. We associate two types of curvature statistical information, denoted by $(G_k, Q_k)^{\mathrm{T}}$, to each aggregate k. Here, G_k is the average curvature of the subaggregates in aggregate k, which is useful for robust detection of regions whose curvature falls off gradually near the patch boundaries (see the head region of the bird in Fig. 4) or for handling noisy boundaries. The entry Q_k is the average variance of curvature of subaggregates in aggregate k, which is useful for characterizing geometric textures. Both G_k and Q_k are computed from the subaggregates by averaging weighted according to the interpolation matrix P, and include its average curvature and the variances in the average curvatures of its subaggregates at all finer scales [33].

Now we define a multiscale similarity measure $\overline{\omega}_{kl}$ between neighboring aggregates k and l as

$$\overline{\omega}_{kl} = (1+\exp(-\rho|G_k - G_l|)) \times (1+\exp(-\varsigma|Q_k - Q_l|))\widehat{\omega}_{kl},$$

where ρ and ς are two parameters weighting the importance of each entry in G_k and Q_k, respectively. These two parameters are used to control the preference of specific scales over others according to prior knowledge of a mesh. In our experiments, we use $\rho = 1.0$ and $\varsigma = 1.0$. The multiscale similarity measure $\overline{\omega}_{kl}$ reflects not only the contrast at the current level, but also the contrast in the properties at all finer levels. Hence, incorporating $\overline{\omega}_{kl}$ into the aggregation process enables us to detect significant geometry transitions seen at any level of scale.

Since the multiscale measurement of each aggregate effectively summarizes the statistical information of its local properties, multiscale feature extraction leads to visually more meaningful results. Figure 3 shows that our feature-extraction method works well for models with small fluctuating features. Figure 4 demonstrates the effectiveness of our method for identifying desirable boundaries within regions that are locally smooth.

3.4 Feature-boundary optimization

The boundaries of features detected at coarse levels are not precise. We use a top-down process, similar to that of [32], to find a precise boundary for each feature at the finest level. Specifically, for a feature at level m with Boolean state vector U^m, we iteratively perform the following steps until we obtain $u = U^0$:

1. Apply the interpolation equation $U^{m-1} \simeq P^m U^m$.
2. The interpolated state vector U^{m-1} is no longer a Boolean vector. To bring the entries of U^{m-1} closer to Boolean values, we modify U^{m-1} as follows:

$$u_i^{m-1} = \begin{cases} 0 & \text{if } u_i^{m-1} < \delta_1, \\ 1 & \text{if } u_i^{m-1} > \delta_2, \\ u_i^{m-1} & \text{if } \delta_1 \le u_i^{m-1} \le \delta_2, \end{cases}$$

where $\delta_1 = 1 - \delta_2 \in [0.1, 0.2]$. The region with vertices whose state variables have values between δ_1 and δ_2 is regarded as fuzzy (e.g. the red region in the left-hand image of Fig. 6).

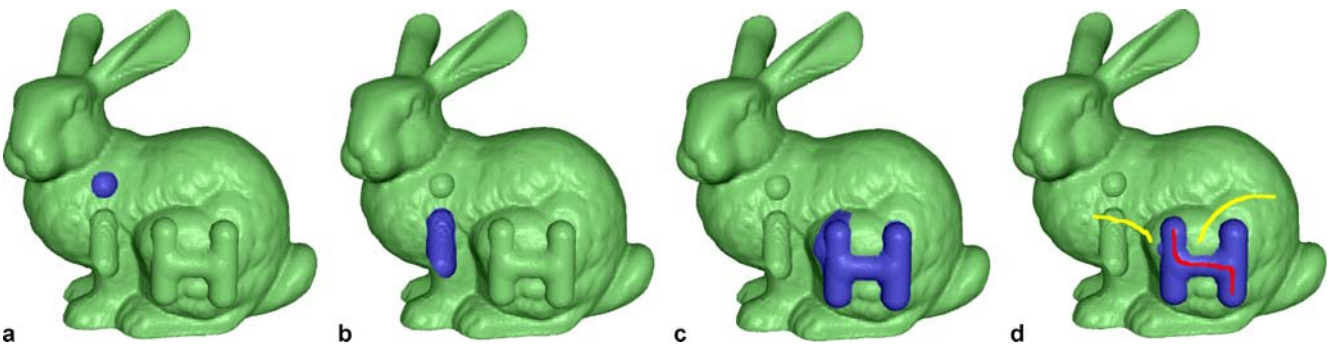

Fig. 5a–d. a, **b** and **c** Automatically detected features (in *blue*) at levels 3, 4 and 5, respectively. **d** The feature in (**c**) is refined with user interaction (*foreground strokes* in *red*, *background strokes* in *yellow*)

Fig. 6. *Left*: the fuzzy region (in *red*) during boundary optimization of a salient feature (in *blue*). *Right*: the final optimized feature boundary

3. To obtain a smoother boundary, for each vertex v_i in the fuzzy region, we apply a Gauss–Seidel relaxation sweep step [32] twice:

$$u_i^{m-1} = \begin{cases} 0 & \text{if } \text{avg}_i^{m-1} < \delta_1, \\ 1 & \text{if } \text{avg}_i^{m-1} > \delta_2, \\ u_i^{m-1} & \text{if } \delta_1 \le \text{avg}_i^{m-1} \le \delta_2, \end{cases}$$

where $\text{avg}_i^{m-1} = \sum_{j\in N(i)}(\omega_{ij}u_j^{m-1}) / \sum_{j\in N(i)}\omega_{ij}$.

At the finest level, we identify those vertices with corresponding state variables larger than δ_2 as being within the final salient region. Figure 6 (right) shows the intermediate fuzzy region during optimization and the final optimized feature boundary at the finest level.

4 Interactive geometric shape extraction

Now we present our interactive geometric feature extraction algorithm. Our user interface allows the user to draw foreground strokes and background strokes to indicate the inside and outside of a desired feature, respectively. We show that the user-specified constraints can be effectively incorporated in our bottom-up aggregation process.

Let $F = \{f_i\}$ and $B = \{b_j\}$ be the vertex indices marked by the foreground and background strokes, respectively. The interactive feature extraction problem can be formulated as a constrained energy minimization problem: $\arg\min_u (u^{\mathrm{T}}Lu / \frac{1}{2}u^{\mathrm{T}}Wu)$ subject to $u_{f_i} = 1$ for $f_i \in F$ and $u_{b_j} = 0$ for $b_j \in B$.

We use a constrained weighted aggregation algorithm to approximately solve the above minimization problem. It is an iterative procedure with each iteration consisting of the following steps:

1. *Coupling weights reassignment.* To ensure that the foreground vertices and background vertices are strongly coupled among themselves, we increase the weight ω_{ij} by setting

   $$\omega_{ij} = \max\{\max_{s\in N(i)} \omega_{is}, \max_{t\in N(j)} \omega_{jt}\}$$

   if both aggregates i and j belong to the foreground or both belong to the background; we decrease the weight ω_{ij} by setting

   $$\omega_{ij} = \min\{\min_{s\in N(i)} \omega_{is}, \min_{t\in N(j)} \omega_{jt}\}$$

   if one of the aggregates i and j is from the foreground but the other is from the background.
2. *Seed selection.* To have at least one foreground seed and one background seed left at the coarsest level, we let the foreground and background aggregates take precedence over all other aggregates for seed selection at each aggregation level. A subset of foreground aggregates are chosen as seeds such that each of the remaining foreground aggregates is strongly coupled to at least one of the foreground aggregate seeds. The same rule is applied to the selection of seeds for the background aggregates.
3. *Modification of interpolation matrix.* To ensure that the graph is aggregated into two parts, the interpolation matrix P is modified during each aggregation step. The modification is only applied to aggregate i that is both a non-seed and a non-stroke aggregate (neither foreground nor background aggregate) and has all three kinds of adjacent aggregate seeds (foreground, background and non-stroke aggregate seeds). We increase its coupling to the foreground aggregates and the background aggregates, and decrease its coupling to the other

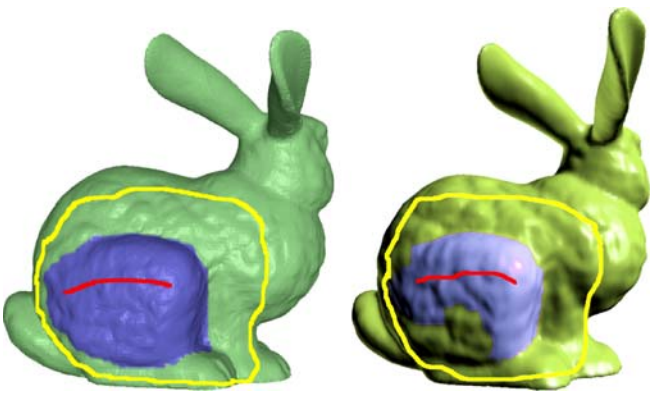

Fig. 7. Comparison with previous method [12]: with similar user-specified strokes, our interactive tool gives more intuitive results (*left*) than those of Ji et al. [12] (*right*)

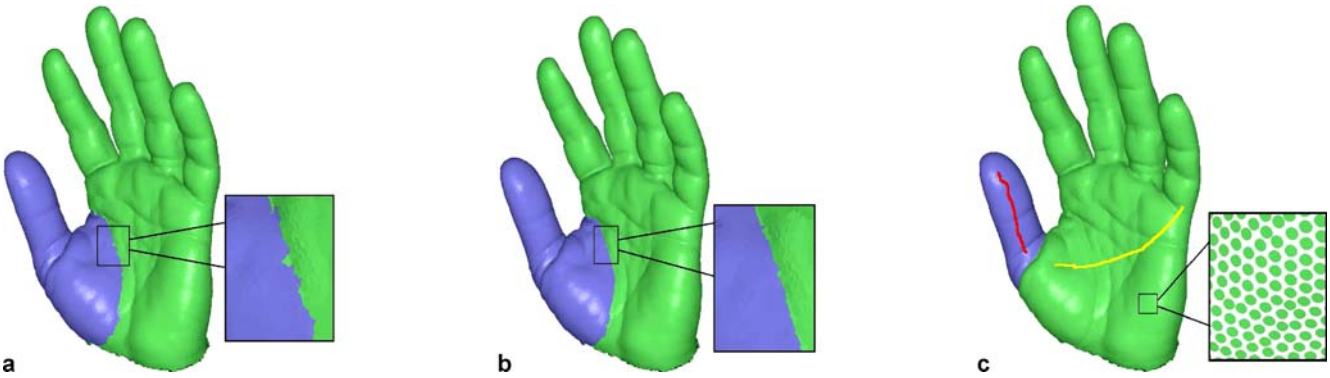

Fig. 8a–c. Extracting salient features from the hand model represented by point-based geometry with 45 K points. **a** Automatic extraction without boundary optimization. **b** Automatic extraction with boundary optimization. **c** Interactive extraction. The neighborhood size is set to 20

non-stroke aggregates by the following modification:

$$p_{if_s} = \frac{\beta\omega_{if_s}}{\varpi_i}, \quad p_{ib_t} = \frac{\beta\omega_{ib_t}}{\varpi_i}, \quad p_{ij} = \frac{(1-\beta)\omega_{ij}}{\varpi_i},$$

with $\varpi_i = \beta(\sum \omega_{if_s} + \sum \omega_{ib_t}) + (1-\beta)(\sum \omega_{ij})$, where $\beta \in (0.5, 1)$. Note that the modified weights satisfy $\sum_{k=1}^{N} p_{ik} = 1$.

At the coarsest level, there are only a foreground seed and a background seed left. Using the top-down boundary optimization process (Sect. 3.4), we extract a salient feature with an accurate boundary.

Figure 5 demonstrates that our interactive tool is useful for extracting more precise features with only a small set of strokes. In Fig. 7, we show a comparison example between our method and the method proposed by Ji et al. [12] (using code presented by the authors of [12]). Unlike [12], which uses a feature measure (to guide the region growing from the user-specified strokes) only at the vertex levels, our method with multiscale similarity measures is more robust to extract regions with noisy shapes or repeated patterns.

5 Extension to point-sampled geometry

A point-sampled geometry is an unstructured set of point samples, with each sample point specified by its location in 3D space, normal vector, color and size [29]. Unlike polygonal meshes, these discrete points do not have explicit connectivity between them. As our automatic and interactive methods both depend on only the coupling weights between adjacent points, rather than the explicit connectivity information, our methods can be easily extended to point-sampled geometry.

We first find the neighboring points for each point. Specifically, for each point, we employ a *kd*-tree to locate its *k* nearest neighbors efficiently. Then we build the coupling weight between each point and its neighboring points based on the mean curvature of each point. We exploit covariance analysis, a robust and efficient mathematical tool in point-sampled geometry [11, 27, 28, 37], to estimate various local surface properties, such as the normal vector and curvature. The discretization of the Laplace–Beltrami operator described in [16] can also be used to address the discretization of the normalized mean curvature for point-sampled surfaces. In our paper, we find that the normal vector and mean curvature computed using the method presented in [37] can obtain good curvature contrast between neighboring points, and the point-sampled surfaces presented in the paper are processed using this technique.

Here, the size of the neighborhood is important for estimating the local surface properties of the point-sampled geometry [28]. In our experiments, we set all neighborhood sizes between 15 and 25. With the normal and curvature at each point, we can define a geometric similarity between adjacent points. Multiscale similarity measures can also be defined similarly.

Since the point-sampled surfaces are usually densely point sampled, during the hierarchical aggregation processing, the area of each segmented patch is computed using the number of point sets in the segmented patch.

Figure 8 shows the effectiveness of our automatic and interactive feature extraction methods on point-sampled geometry. Similar to the boundary optimization techniques used in the case of mesh models, the boundary of the point set patch (Fig. 8b) is optimized using two Gauss–Seidel relaxation sweep steps.

6 Partial shape matching

In this section, we introduce a simple and effective partial shape matching approach, based on the multiscale similarity measure and the energy function for detecting salient features.

We define a geometric saliency descriptor for partial matching based on the theory of salience of visual parts proposed by Hoffman and Singh [10], which says that the saliency of a visual part depends on three factors: its size relative to the whole object, the degree to which it protrudes and the length of its boundaries. Our saliency descriptor includes multiscale curvature changes and the boundary energy function for salient features. In particular, the saliency of an aggregate S at a level m depends on the following five factors:

- $\Gamma(S)$: the energy function which determines whether S is a salient feature,
- Area(S): the area of S relative to the total area of the mesh,
- $N(S)$: the number of local minimum or maximum curvatures in S,
- MAC(S): the multi-average curvature vector of S, and
- MVC(S): the multi-variance curvature vector of S,

where MAC(S) is described by the vector $G_s^{[m]}$:

$$G_s^{[m]} = \left(\overline{G}_s^{[1,m]}, \ldots, \overline{G}_s^{[m-1,m]}, \overline{G}_s^{[m]}\right)^{\mathrm{T}}$$

and MVC(S) is described by the vector $Q_s^{[m]}$:

$$Q_s^{[m]} = \left(\overline{Q}_s^{[1,m]}, \ldots, \overline{Q}_s^{[m-1,m]}, \overline{Q}_s^{[m]}\right)^{\mathrm{T}}.$$

The factors MAC(S) and MVC(S) are the average curvature statistics over the aggregate S computed from its subaggregates at all finer scales, weighted according to the interpolation matrix [33].

We define the saliency descriptor of aggregate S as a vector of the multilevel properties:

$$F_S = (\Gamma(S), \mathrm{Area}(S), N(S), \mathrm{MAC}(S), \mathrm{MVC}(S))^{\mathrm{T}},$$

called the saliency grade. We define the similarity measure between aggregates S and L for partial shape matching as $\mathrm{Similarity}(S, L) = \exp(-(F_S)^{\mathrm{T}} \Lambda F_L)$, where Λ is a diagonal matrix which weights the importance of every factor. We set all the elements of the matrix Λ to 1 in all the examples in our paper, since we have no special interest in any factor. Let aggregate S be at level m_1 and aggregate L be at level m_2 with $m_1 < m_2$. Then, we only compare the m_1 entries in MAC(S) (MVC(S)) with the last m_1 entries in MAC(L) (MVC(L)). The distance between these saliency grades is computed using the Mahalanobis distance.

We can use the saliency grade F_S of aggregate S for self-shape matching with ease. At each aggregation level, the aggregates are chosen as seeds in descending order of average curvature. Using the method described in Sect. 3, all the aggregates at each level of the pyramid with low value $\Gamma(u)$ are automatically detected as the salient geometry features. Then, we compute the salient grade vector F_S for each feature and use it as the similarity-invariant vector index. The geometric features are extracted off-line and stored in a rotation- and scale-invariant database [6]. The salient geometric features S within the database with aggregated properties most similar to the query feature T are chosen as the features most partially matching the query feature. Specifically, while searching through the

Fig. 9. Self-shape matching. *Left*: a flower is cut out using our interactive tool. *Middle left*: the other five similar salient geometric features on the Buddha model are retrieved using the partial matching operator. *Middle right*: aggregates at level 7. *Right*: given one foot extracted using our interactive tool, three other feet are automatically retrieved

database, we compute the Similarity(S, T) between S and T using their summarized properties F_S and F_T, and then detect the features S with low Similarity(S, L) value as the most partially matching features to the query feature. The query feature T can be chosen either using the interactive techniques described in Sect. 4, or a user-specified region on the shape model.

As shown in the examples in Fig. 9, we first extract the query features (i.e. a flower on the Buddha model and a foot on the statue model) using the interactive technique described in Sect. 4, and then we search the feature vector databases; the other five similar salient features (including three other flowers) are found on the Buddha model and three other feet are found on the statue model. It is noteworthy that these retrieved features are visually noticeable in certain aggregation levels, e.g. level 9 for the Buddha model and level 7 for the statue model. For the Buddha model and the statue model, there are 150 and 138 geometric features stored in their databases, respectively. These examples demonstrate that using the presented saliency descriptors incorporated with the surface multiscale curvature analysis is an efficient partial matching method.

Our method is different from the method of Gal and Cohen-Or [6] in the following aspects. First, unlike [6], our descriptor for local shape comparison includes the multiscale curvature change and the boundary energy function of the features. Second, our saliency detection is based on hierarchical vertex aggregation. It avoids the construction of the implicit surface applied in [6] to define the curvature and the local surface descriptors, making our construction time and storage size much smaller.

7 Results and discussion

We have presented a variety of examples to demonstrate the effectiveness of our automatic and interactive methods to extract meaningful features. In this section, we discuss the runtime complexities of our methods, and also give more comparison results with existing methods.

Similar to the image-segmentation method of Sharon et al. [32], the computational complexities of our unconstrained and constrained weighted aggregation processes are both linear in the data size. At the finest level, very simple aggregation is done among individual vertices. Although the complexity per aggregate increases linearly with the number of levels, the number of aggregates drops geometrically. Despite the incorporation of the top-down boundary optimization process, the complexity remains linear since the finest scale of aggregates needed to optimize the segment boundaries is proportional to the number of aggregates involved.

We give the typical timing here for some examples used in this paper, measured on a 3.2 GHz Pentium 4 PC with 1 GB of RAM. Using our current unoptimized implementation, it takes about 1 s to build the whole hierarchy of aggregates for the Moai model with 10 002 vertices. For the hand model with 45 000 points, the statue model with 255 845 vertices and the Buddha model with 543 652 vertices, the corresponding running times of performing the whole bottom-up aggregation procedure are 2.5 s, 13.5 s and 25 s, respectively.

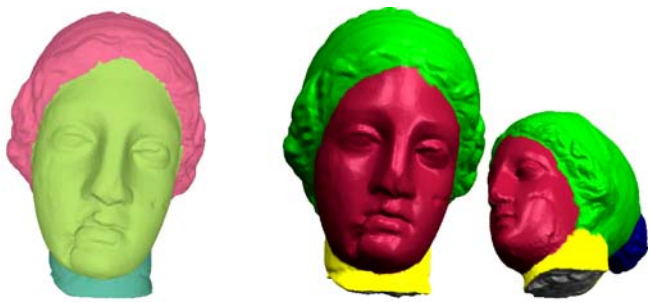

Fig. 10. Comparison with previous method [14]. *Left*: segmentation result with Katz et al. [14] (three patches). *Right* (two views): result with our method (five patches)

In terms of asymptotic complexity, our method is much faster than most of the state-of-the-art mesh-segmentation algorithms. For example, most of them [14, 15, 21] define similarity measures based on all pairwise geodesic distances, which are typically computed by Dijkstra's shortest-path algorithm in $O(N^2 \log N)$ time with N denoting the number of vertices to be segmented.

For the initialization of the feature database for partial shape matching, our method is significantly faster than that of Gal and Cohen-Or [6], as the latter involves the construction of an implicit surface to define the local surface descriptors. For example, the initialization time for the Buddha model is 19 min with their method (conducted on a 3.0 GHz Pentium 4 PC with 2 GB of RAM), compared with 25 s with our method.

In Fig. 10, we compare our automatic segmentation results with those by Katz and Tal [14]. We choose to use the Igea model, since it is hard to segment due to the bumpy hair region. The model is decomposed into three patches in [14] using the k-means clustering. At level 9, our bottom-up aggregation method produces five aggregates. Note that the face (in red) is detected as a salient feature with the minimal energy; the other four patches are also detected as meaningful.

Limitations. Shape-segmentation and feature-extraction methods are not omnipotent for all kinds of applications [2]. The salient features detected by our automatic shape extraction method generally do not form a partition of the original surface. This means that our automatic method is not suitable for applications which need segmentation methods to analyze objects in a global sense, e.g. skeleton extraction [14]. In addition, since our method

strictly relies on local measures, local aggregations (even in a multiscale manner) may not always suffice for capturing some global characteristics, for example global symmetry.

8 Conclusion

We presented a bottom-up framework to efficiently extract meaningful features from either irregular meshes or unorganized point sets. We introduced a multiscale geometric similarity measure, ensuring that high-level features are obtained through local aggregation. We integrated both automatic and interactive shape extraction methods into the same framework. A simple and effective solution to partial shape matching was also presented. All these methods are very efficient, thanks to the bottom-up construction process.

It is very interesting to explore other measures, apart from curvature and normal direction, with a similar bottom-up framework. For the multiscale measures, we only consider the curvature statistics. We plan to incorporate high-level shape descriptors, for example, ridge–valley lines [25], to facilitate feature extraction. In this paper, we demonstrated the use of the partial shape matching operator in only one application, which is self-shape matching. Other promising applications include shape alignment and surface reconstruction.

Acknowledgement We would like to thank Ligang Liu for the code of the easy mesh cutting algorithm [12]. We would also like to thank the anonymous reviewers for their insightful comments. This work was supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project No. HKUST620107) and a grant from the NSF of China (No. 40701154).

References

1. Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical mesh segmentation based on fitting primitives. Visual Comput. **22**(3), 181–193 (2006)
2. Attene, M., Katz, S., Mortara, M., Patane, G., Spagnuolo, M., Tal, A.: Mesh segmentation – a comparative study. In: International Conference on Shape Modeling and Applications (SMI'06), p. 7. IEEE Computer Society Press (2006)
3. Cohen-Steiner, D., Morvan, J.M.: Restricted Delaunay triangulations and normal cycle. In: ACM Symposium on Computational Geometry (San Diego, CA), pp. 237–246. ACM, New York, NY (2003)
4. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proceedings of ACM SIGGRAPH 99, pp. 317–324. ACM Press/Addison-Wesley Publishing Co. (1999)
5. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. ACM Trans. Graph. **23**(3), 652–663 (2004)
6. Gal, R., Cohen-Or, D.: Salient geometric features for partial shape matching and similarity. ACM Trans. Graph. **25**(1), 130–150 (2006)
7. Garland, M., Willmott, A., Heckbert, P.S.: Hierarchical face clustering on polygonal surfaces. In: Proceedings of the 2001 Symposium on Interactive 3D Graphics (SI3D '01), pp. 49–58. ACM (2001)
8. Gatzke, T., Grimm, C.: Feature detection using curvature maps and the min-cut/max-flow algorithm. In: Geometric Modeling and Processing, pp. 578–584. IOS Press (2006)
9. Gatzke, T., Grimm, C., Garland, M., Zelinka, S.: Curvature maps for local shape comparison. In: Shape Modeling International, pp. 244–256. IEEE Computer Society (2005)
10. Hoffman, D.D., Singh, M.: Salience of visual parts. Cognition **63**(1), 29–78 (1997)
11. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. In: SIGGRAPH, pp. 71–78. Academic Press Professional, Inc. (1992)
12. Ji, Z., Liu, L., Chen, Z., Wang, G.: Easy mesh cutting. Comput. Graph. Forum **25**(3), 283–291 (2006)
13. Katz, S., Leifman, G., Tal, A.: Mesh segmentation using feature point and core extraction. Visual Comput. **21**(8–10), 649–658 (2005)
14. Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans. Graph. **22**(3), 954–961 (2003)
15. Lai, Y.K., Zhou, Q.Y., Hu, S.M., Martin, R.R.: Feature sensitive mesh segmentation. In: Solid and Physical Modeling Symposium 2006 (SPM'06), pp. 17–25. ACM (2006)
16. Lange, C., Polthier, K.: Anisotropic smoothing of point sets. Spec. Issue Comput. Aided Geom. Des. **22**(7) (2005)
17. Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., Seidel, H.P.: Intelligent mesh scissoring using 3D snakes. In: Pacific Graphics, pp. 279–287. IEEE Computer Society (2004)
18. Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., Seidel, H.P.: Mesh scissoring with minima rule and part salience. Comput. Aided Geom. Des. **22**(5), 444–465 (2005)
19. Li, X., Guskov, I.: Multi-scale features for approximate alignment of point-based surfaces. In: Symposium on Geometry Processing, pp. 217–226. Eurographics Association (2005)
20. Liu, R., Jain, V., Zhang, H.: Subsampling for efficient spectral mesh processing. Lect. Notes Comput. Sci. **4035**, 172–184 (2006)
21. Liu, R., Zhang, H.: Segmentation of 3D meshes through spectral clustering. In: Pacific Graphics, pp. 298–305. IEEE Computer Society (2004)
22. Liu, R., Zhang, H.: Mesh segmentation via spectral embedding and contour analysis. Comput. Graph. Forum **26**(3), 385–394 (2007)
23. Mangan, A.P., Whitaker, R.T.: Partitioning 3D surface meshes using watershed segmentation. IEEE Trans. Vis. Comput. Graph. **5**(4), 308–321 (1999)
24. Mortara, M., Patane, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. In: Ninth ACM Symposium on Solid Modeling and Applications (SMI'04), pp. 339–344. IEEE Computer Society (2004)
25. Ohtake, Y., Belyaev, A., Seidel, H.P.: 3D scattered data approximation with adaptive compactly supported radial basis functions. In: International Conference on Shape Modeling and Applications (SMI'04), pp. 31–39. IEEE Computer Society (2004)
26. Page, D.L., Koschan, A., Abidi, M.: Perception-based 3D triangle mesh segmentation using fast marching watersheds. In: Computer Vision and Pattern Recognition, vol. 2, pp. 27–32. IEEE Computer Society (2003)
27. Pauly, M., Gross, M., Kobbelt, L.P.: Efficient simplification of point-sampled surfaces. In: Visualization, pp. 163–170. IEEE Computer Society (2002)

28. Pauly, M., Keiser, R., Gross, M.: Multi-scale feature extraction on point-sampled surfaces. In: Eurographics, pp. 281–290. Eurographics Association (2003)
29. Pfister, H., Zwicker, M., van Baar, J., Gross, M.: Surfels: surface elements as rendering primitives. In: SIGGRAPH, pp. 335–342. ACM Press/Addison-Wesley Publishing Co. (2000)
30. Shamir, A.: Segmentation and shape extraction of 3D boundary meshes. In: State-of-the-Art Report, Proceedings of Eurographics 2006, pp. 137–149. Eurographics Association (2006)
31. Sharf, A., Blumenkrants, M., Shamir, A., Cohen-Or, D.: SnapPaste: an interactive technique for easy mesh composition. Visual Comput. **22**(9), 835–844 (2006)
32. Sharon, E., Brandt, A., Basri, R.: Fast multiscale image segmentation. In: Computer Vision and Pattern Recognition, vol. 1, pp. 70–77. IEEE Computer Society (2000)
33. Sharon, E., Brandt, A., Basri, R.: Segmentation and boundary detection using multiscale intensity measurements. In: Computer Vision and Pattern Recognition, vol. 1, pp. 70–77. IEEE Computer Society (2001)
34. Sharon, E., Galun, M., Sharon, D., Basri, R., Brandt, A.: Hierarchy and adaptivity in segmenting visual scenes. Nature **442**, 810–813 (2006)
35. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**, 888–905 (2000)
36. Vieira, M., Shimada, K.: Surface mesh segmentation and smooth surface extraction through region growing. Comput. Aided Geom. Des. **22**, 771–792 (2005)
37. Xiao, C., Miao, Y., Liu, S., Peng, Q.: A dynamic balanced flow for filtering point-sampled geometry. Visual Comput. **22**(3), 210–219 (2006)
38. Yamazaki, I., Natarajan, V., Bai, Z., Hamann, B.: Segmenting point sets. In: Shape Modeling and Applications, pp. 46–51. IEEE Computer Society (2006)
39. Zelinka, S., Garland, M.: Similarity-based surface modelling using geodesic fans. In: Symposium on Geometry Processing, pp. 209–218. Eurographics Association (2004)
40. Zelinka, S., Garland, M.: Surfacing by numbers. In: Graphics Interface, pp. 107–113. Canadian Information Processing Society (2006)
41. Zhang, Y., Paik, J., Koschan, A., Abidi, M.A., Gorsich, D.: A simple and efficient algorithm for part decomposition of 3-D triangulated models based on curvature analysis. In: Proceedings of the International Conference on Image Processing, pp. 273–276. IEEE Computer Society (2002)

CHUNXIA XIAO received the Ph.D. degree from the State Key Laboratory of CAD&CG, Zhejiang University, China in 2006 and is currently an assistant professor at Computer School, Wuhan University, China. During October 2006–April 2007, he worked as a visiting scholar at the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include digital geometry processing, point-based computer graphics and image and video editing.

HONGBO FU received the B.S. degree in information science from Peking University, China in 2002 and the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology in 2007. His primary research interests fall in the field of computer graphics with an emphasis on digital geometry processing, character animation and hairstyle synthesis and analysis.

CHIEW-LAN TAI received the B.Sc. degree in mathematics from the University of Malaya, the M.Sc. degree in computer and information sciences from the National University of Singapore and the D.Sc. degree in information science from the University of Tokyo. She is an associate professor of computer science at Hong Kong University of Science and Technology. Her research interests include geometric modeling, computer graphics and reconstruction from architecture drawings.