

Segmenting Geometric Reliefs from Textured Background Surfaces

Shenglan Liu^{1,2}, Ralph R. Martin¹, Frank C. Langbein¹ and Paul L. Rosin¹

¹ School of Computer Science, Cardiff University, UK

² CMEE, Nanjing University of Aeronautics and Astronautics, China
{Shenglan.Liu, Ralph, F.C.Langbein, Paul.Rosin}@cs.cf.ac.uk



Fig. 1: A pen container showing a relief on a textured background and a corresponding segmentation.

ABSTRACT

Segmentation of geometric reliefs from a textured background has various applications in reverse engineering. We consider two approaches to solve this problem. The first classifies parts of a surface mesh as relief or background, and then uses a snake which moves inwards towards the desired relief boundary, which is coarsely located using an energy based on the classification. The second approach initially smoothes the surface to eliminate the background texture, and locates the snake at the relief boundary using an energy based on the step between the background and the relief.

Both snakes start at simple user-drawn contours, and are driven towards the relief boundaries by the snake energy functional. In both cases, the snake has different evolution phases with different energy terms, to initially rapidly drive the snake towards the relief boundary, and to later accurately match it.

To describe geometric textures, we analyze surface differential properties, and integral and statistical quantities based upon them, computed at multiple scales taken over local neighborhoods, following similar ideas from image texture processing. For classification, we use a support vector machine together with sequential forward floating search for feature selection. A straightforward Laplacian method is used for smoothing.

We use example scanned models to demonstrate that both approaches are useful, but are suitable for different types of model.

Keywords: Reliefs, segmentation, geometric texture, snakes, mesh processing, reverse engineering.

1. INTRODUCTION

Sculptured *reliefs* are widely used in various industries for purposes such as applying brands to products, decorating craftwork, and beautifying packaging. For example, in the packaging market, the requirement for more elaborate

designs continues to grow as manufacturers wish to distinguish their products, and as consumers come to expect superior packaging. Along with the growing significance of product packing, there is pressure to minimize costs and to keep the time-to-market to a minimum. Thus, it is often not possible for an artist to spend days producing detailed relief designs. Instead, there is a necessity for relief decoration to be produced quickly. One approach to doing so is for a reverse engineering process to extract previously designed and manufactured reliefs, and reapply them to new CAD models, providing large cost and time savings for the packaging industry.

The reverse engineering of reliefs requires several steps. Firstly, triangle mesh data must be captured from the surface of the object bearing the relief, using a 3D scanner. The relief must then be separated from the background in a segmentation step. Finally, the relief must be applied to the new surface, which typically will have a different geometric shape. This paper addresses the *segmentation* stage of reverse engineering of reliefs.

Generally, we may define a *relief* to be an area of a surface with sculpted features different from those of the underlying surface, and which is raised by a small height; this height is typically (but not always) larger than the characteristic size of features on the background. There are various kinds of reliefs, and they can be imposed on diverse backgrounds. The simplest case is that of an isolated relief delimited by a single outer contour, lying on a smooth and slowly varying background. We have already addressed the segmentation problem for such reliefs in [30].

Here we consider the segmentation of reliefs lying on a *textured background*, as in Fig. 1. While we could consider the container in Fig. 1 to bear an *all-over* relief, in many cases of this kind, we are more interested in segmenting e.g. the bird and branch in the foreground from the underlying textured background pattern. Note that this is a *geometric texture*, a small displacement field superimposed upon a base surface which is assumed to be smooth. The texture itself may be repetitive, or it may be stochastic in nature. It may also vary from place to place, according to the shape of the underlying surface, or otherwise. Our goal is to take as input a triangular mesh created from data points captured by a 3D scanner, and to produce as output a single connected component of the mesh representing the relief. In practice we find a boundary curve separating the relief from the rest of the mesh, as shown in Fig. 1.

Two alternative approaches to this problem are readily apparent. The first is to *classify* each part of the mesh as belonging to the textured background, or to the relief, and then to place an optimal boundary curve separating these regions which is both smooth, and which minimizes the amount of mesh lying on the wrong side of the boundary curve according to the classification. The second approach starts by *smoothing* the surface to eliminate the background texture (and detail inside the relief), and then uses our previous method for segmenting reliefs lying on smooth backgrounds [30]. The *classification approach* relies on the assumption that there is sufficient difference in surface properties between the textured background and the relief. The key issues are which *surface properties* to use to analyse the surface locally, and which *classification method* to use to combine them to give a classification for each vertex. The *smoothing approach* relies on being able to remove the texture but at the same time leaving a clear step to the relief, and not smoothing that away too. Thus, the height of the relief above the background should be sufficiently large compared to the variations in height of the background texture. This requirement is often satisfied—a relief is supposed to stand out from its background—but not always, so this approach is not universally applicable.

This paper considers methods for segmenting reliefs on textured backgrounds based on these two approaches. In both approaches we use a *snake* to move from an initial user-given contour to the desired boundary. Minimizing its energy is used to finally obtain the optimal relief boundary. To analyze geometric texture, we measure various local surface differential properties, and integral and statistical descriptors based upon them, computed at multiple scales taken over local neighborhoods. For classification, we apply a non-linear supervised learning method, a *support vector machine* (SVM). In this approach, the snake first coarsely approaches the classification boundary, and is then adjusted to the desired relief boundary. In the smoothing approach, a simple Laplacian surface smoothing method is adopted. Here, we first move the snake to approximately the correct location using the smoothed model; subsequently, further optimization is used on the original unsmoothed model to make the snake accurately lie at the desired final position.

In Section 2 we review related work: we briefly discuss *image texture* segmentation methods (little work has been done on *geometric texture* segmentation), classifier selection, mesh smoothing, snakes, and our previous relief segmentation method. In Section 3 we present an overview of our two approaches, while details are discussed in Sections 4 and 5 individually. Results are demonstrated in Section 6, and conclusions are drawn in Section 7.

2. RELATED WORK

2.1 Texture Segmentation in Images

Texture segmentation in digital images has been widely studied for several decades. Numerous approaches are used; good surveys are provided in [45] and [53]. Generically, many approaches follow independent phases of feature extraction, classification (after perhaps selecting certain features), and clustering.

The first important phase is feature extraction, where a set of feature descriptors is generated to represent the characteristics of each pixel or region. Feature extraction methods are categorized in [53] into statistical methods (e.g. *co-occurrence matrix* features and *autocorrelation* features [18]), model-based methods (e.g. *autoregressive* features [32] and *Markov random field* models [31]), signal processing methods (e.g. Laws masks, Gabor filters and wavelet transforms [44]) and geometrical methods (e.g. Voronoi tessellation features [52]). The first three methods are most commonly used in practice. Various comparative studies have been performed, usually coming to the conclusion that co-occurrence features are the most useful [10, 38, 49, 57], or occasionally that no single feature extraction method is consistently superior [6, 44]. Methods are still being improved. For example, Deng and Clausi [11] recently gave a superior scheme for implementing the Markov random field model, while Montiel et al. [36] usefully extended co-occurrence matrices to one-dimensional conditional histograms. Further papers [5, 37, 59] use a combination of features of several types to achieve improved texture segmentation performance.

Some texture feature extraction methods, such as autocorrelation features and Markov random field models, rely on the regular grid structure of a 2D image, so are difficult to directly extend to an irregular 3D triangle mesh. Others are more readily modified, such as co-occurrence matrix features, and we adopt these for our algorithm. The co-occurrence matrix approach allows many features to be extracted, such as energy, entropy, correlation and contrast, which are usually referred to as *second-order statistics* [18]. We also extend the *first-order* statistical features described in [33], such as mean, variance and energy, to characterize geometric texture.

Geometric texture on 3D surfaces has mainly been studied for purposes of texture synthesis, to produce geometric details for visual realism. Recent literature on this work includes papers such as [1] and [26]. There is little on geometric texture segmentation or classification, although Lai et al. [27] gives one approach.

2.2 Classifiers

As well as the features used, another crucial component in texture segmentation is the classifier, which decides to which region (in our case, foreground or background) each point belongs. Various commonly used classifiers include the *k-nearest neighbor approach* (KNN), *Fisher linear discriminants*, *support vector machines* (SVMs) and *neural networks* [12].

Support vector machines, introduced by Vapnik [54] solve the *two-class* pattern recognition problem, based on structural risk minimization. The idea is to map the input feature vectors into a high dimensional feature space through some non-linear mapping (chosen *a priori*), and then to construct an optimal separating hyperplane in this space. SVMs have proven successful for various tasks such as text categorization [21], face detection [39], gene selection [17], and importantly, image texture classification [24, 29].

In this paper, we apply an SVM to the problem of classification of mesh points as *textured background* or *relief*, as SVMs offer the following advantages which are relevant to our application:

- SVMs are usually based on a linear decision hyperplane and a simple kernel function, so they are relatively robust to problems of overfitting the training data.
- SVMs avoid the need for a large training set when used with many features, again because of their simple model.
- SVMs are relatively insensitive to inclusion of features which provide little discrimination between the classes.

2.3 Mesh Smoothing

Mesh smoothing, or fairing, is a technique generally based on minimization of some kind of fairness energy functional. It is usually implemented by modifying the locations of mesh points without changing the mesh topology. Some approaches for discrete meshes are non-linear: for example, Welch and Witkin [55] minimize a functional based on

total curvature using a local quadratic least-squares approximation, while Schneider and Kobbelt [46] propose an algorithm satisfying boundary conditions based on solving a partial differential equation. However, most mesh smoothing schemes are based on linear operators, leading to simple and fast algorithms: for example, the well-known *Laplacian smoothing* method [13, 51], and *mean filtering* and *median filtering* [60] which are extensions of ideas from image processing. Recently, more attention has been paid to feature-preserving smoothing, which tries to minimize the effects of noise while simultaneously preserving features such as edges [3, 15, 48, 60].

Our previous relief segmentation method relies on a smooth background surface and a small but definite relief step height [30]. Thus, to be able to use the same method in our *smoothing approach*, ideally, the details of the texture should be removed but the step feature at the relief boundary would be kept. We considered various feature-preserving smoothing methods, including those in [15, 48, 60]. Unfortunately, if such methods are tuned to retain the step feature, they also preserve the features in the background texture too much, causing our previous relief segmentation algorithm for smooth backgrounds to fail. Thus, we adopt in this paper the fastest and simplest smoothing method—the classical Laplacian approach [13]. However, it only works well when there is a sufficiently large relief step height.

2.4 Snakes

Snakes, or, more formally, *active contour models*, were proposed by Kass [23] to detect features in an image or image sequence. A snake is an energy-minimizing spline (here, a polyline) controlled both by *internal* forces, such as rigidity and elasticity of the curve, which tend to make it smooth, and shrink inwards, and *external* forces such as constraint and image forces, which help to drive it toward the desired image features. Various later improvements to image snakes allow them to avoid sensitivity to choice of the initial contour [9], and to explore image boundary concavities [7, 58].

Snakes have also been used on surface meshes for segmentation [35], and for detection of features such as sharp edges and corners [22, 28]. Other studies of snakes on meshes have considered the problem of topology control [2, 22].

2.5 Relief Segmentation on a Smooth Background

The relief segmentation problem on a triangle mesh in the simplest case of an isolated relief lying on a smooth and slowly varying background was addressed in our earlier paper [30]. A snake was used to solve the problem. The snake starts from a user-drawn curve on the mesh, and evolves until it matches the boundary of the relief. To deal with the particular problem of relief segmentation, various specific features were needed:

- A feature energy term specifically designed to detect the step at the edge of a relief.
- A deflation force with strength determined dynamically to balance the snake's internal energy, to make the snake insensitive to choice of initial contour.
- A refinement phase designed to make the snake explore relief contour concavities.

Encouraged by the successful use of a snake in our earlier work, we also use a snake in this paper to locate the relief boundary, in both the texture classification, and surface smoothing, approaches. The details of the snake and its use differ in each case, however.

3. ALGORITHM OVERVIEWS

We now outline our two approaches to relief segmentation on textured backgrounds: the *classification approach* and the *smoothing approach*.

3.1 Overview of the Classification Approach

The classification approach is based on extending ideas of texture classification from image processing. There are three main steps (see Fig. 2): calculation of surface properties at mesh points, classification of mesh points using an SVM, and boundary curve location using a snake. We assume that the user indicates a few small typical areas belonging to the relief, and a few belonging to the background, using a simple point-and-drag interface, to provide training information for the classifier (see Fig. 4(a)). We also assume the user specifies an initial starting contour lying on the background, which may be quite far from the final segmentation boundary. The algorithm moves it to the final boundary.

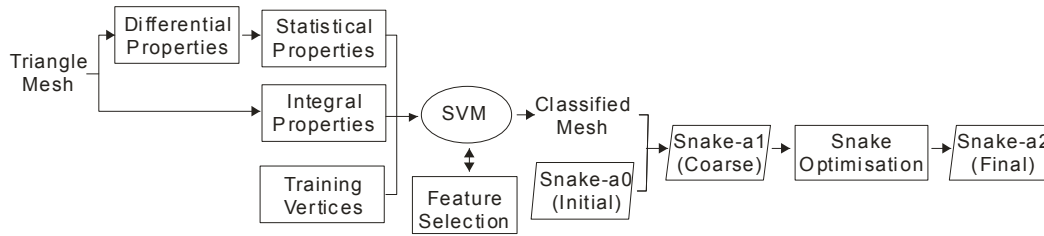


Fig. 2: Flowchart of classification approach.

- *Property calculation*

Firstly, we calculate various local surface *differential properties* at each mesh point (such as planarity, curvature and normal difference). These properties are estimated using a small neighborhood of nearby mesh points. We also calculate *integral property* estimates at multiple scales corresponding to neighborhood balls of different radii. In addition to the basic properties themselves, we also compute multiple-scale *statistical properties* from the differential properties including first- and second-order statistics, which measure the probabilities of individual vertex values and the probabilities of their co-occurrence. We discuss later the choice of particular properties used.

- *Classification by SVM*

The user-specified training data provides several surface regions known to belong to background or relief. To train the SVM model, we may use *all* property features or only certain *selected* features. To perform feature selection, we first use the SVM to determine how well each property can distinguish the mesh points inside the training regions. For a given input mesh, certain properties may give good discrimination, while others only poor discrimination, between background and relief. We then build a classifier based on the ones which give good discrimination, using the technique of *sequential floating forward search (SFFS)* [43] to choose the best set of properties to use for this mesh. After training the SVM with these properties, we use it to classify each mesh point as background or relief.

- *Snake segmentation*

Given the user-specified initial contour, the snake is evolved to find a coarse boundary between the texture and relief using a feature energy term based on the classification. However, individual vertices at the relief boundary can easily be misclassified, so this coarse snake needs to be optimized. The final accurate relief boundary is found by giving the snake a new attractive energy term and a revised feature energy term. Both terms are based on integral properties which are good at locating the boundary when calculated at an appropriate neighborhood size.

3.2 Overview of the Smoothing Approach

The mesh smoothing approach has three main steps: mesh smoothing, snake-based segmentation on the smoothed mesh, and snake location optimisation on the original mesh, as shown in Fig. 3. In this approach, the user simply draws an initial contour for the snake, which again may be far from the final boundary.

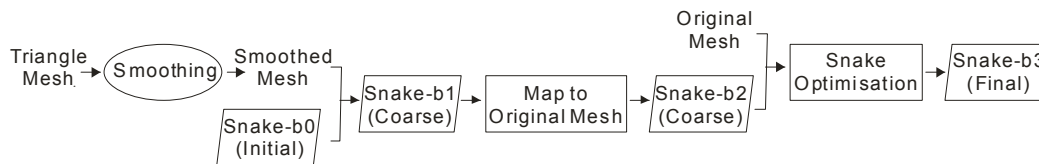


Fig. 3: Flowchart of smoothing approach.

- *Mesh smoothing*

Initially, Laplacian smoothing is applied to the mesh, with the aim of removing the texture. After several iterations of smoothing, small mesh features such as the background texture, and details on the relief, are removed.

- *Snake segmentation on the smoothed mesh*

On the smoothed mesh, we apply our previous approach, for relief segmentation on a surface with a smooth background, to move the initial contour to the relief boundary [30]. In the coarse evolution phase, we adopt the same energy terms as before such as internal energy, feature energy and deflation energy. The refinement evolution phase can be omitted when there are no deep concavities; we note that this smoothing approach is not well suited to reliefs which have deep concavities, and in such cases the classification approach may be better.

- *Snake optimisation on the original mesh*

The above step locates the snake on the smoothed mesh. However, because smoothing causes significant shape distortion of the original mesh, a final step is needed to accurately locate the final contour on the original mesh. The snake output from the previous stage is mapped back to the original mesh and its position is optimized. We use the same snake optimisation method as in the classification approach, adding a new attractive energy term and a revised feature energy term.

4. TEXTURE CLASSIFICATION APPROACH

4.1 Texture Properties

We now consider in detail differential geometric properties, and derived properties, of a surface. These are *local surface properties*, *integral surface properties*, and *statistical properties* derived from them. The integral and statistical properties are considered to be *geometric texture properties*, which we use for classification.

4.1.1 Surface Properties

Differential geometric quantities such as normals and curvatures may be used to describe the local shape of a surface. On discrete surfaces such as triangle meshes, they can be estimated at each vertex as spatial averages around this vertex. The normal at a vertex can be estimated by various methods. Two commonly used approaches are to use weighted averages of face normals in a neighborhood [50], and to fit a least-squared plane [40] or higher-order surface. Many methods have also been proposed for curvature estimation, such as least-squares paraboloid surface fitting [35], Taubin's discrete approximation [50], and a method using Voronoi cells [34]. Here we follow [50] and [35] to estimate normals and curvatures.

From normal vectors, the derived *local surface properties* of *planarity* and *normal difference* can be computed [47]. These give information about how the surface is varying in a small neighborhood. The planarity L_{pl} at a vertex is defined as the signed distance between the vertex and a plane fitted to its neighbors. One method to compute this quantity at a vertex \mathbf{v} is:

$$L_{pl} = \mathbf{n} \cdot \left(\frac{1}{|N|} \sum_{\mathbf{v}' \in N} (\mathbf{v}' - \mathbf{v}) \right), \quad (1)$$

where N is some neighborhood of \mathbf{v} . The normal difference L_{nd} at a vertex is defined as the average absolute difference between the vertex normal and the normal at each neighbor. It can be computed using:

$$L_{nd} = 1 - \frac{1}{|N|} \sum_{\mathbf{n}'} (\mathbf{n}' \cdot \mathbf{n}), \quad (2)$$

where \mathbf{n} is the normal at \mathbf{v} and \mathbf{n}' is the normal at the neighbor \mathbf{v}' .

Various differential attributes are associated with curvature: mean curvature, Gaussian curvature, principal curvatures and principal directions. We choose the mean curvature L_{Ht} , which is less sensitive to noise than Gaussian curvature, as a representative curvature property.

Thus, for every vertex on the triangle mesh, we so far have three surface properties—planarity, normal difference and mean curvature. We estimate these using the immediate 1-ring neighbors, and refer to them as *local surface properties*.

Other papers have suggested using extended neighborhoods to deal with noisy data, for example using neighbors inside a given geodesic distance [41], or calculating properties on multiple scales [42]. In particular, for a relief on a textured geometric background surface, use of *integral surface properties* covering neighborhoods chosen according to the size of texture seems particularly appropriate for classifying surface points, while neighborhoods of a similar size to the relief step height are useful for distinguishing the relief boundary in the snake segmentation phase. The texture

pattern size and the relief step height can be easily estimated by asking the user to select an appropriate pair of points on the mesh. In fact, we adopt here the idea of using *multiple* scales. Balls of different radii are used to collect neighbors to estimate the above three surface properties. The radii of the balls used can be specified by the user, or more simply, the largest radius can be set by the user and other sizes generated according to some simple rules. Most of our tests use the latter method. For example, if we wish to use three sizes of balls, where the largest has radius r , the other two radii might be set to be $0.5r$ and $0.25r$.

4.1.2 Statistical Properties

Image texture has been widely studied for a long time, but there is still no universally accepted definition of image texture. It can be regarded as a function of the *spatial variation* in pixel intensities (in a grayscale image). Two intuitive characteristics have been agreed. Firstly, there is significant variation in grey levels between *nearby* image elements within a given texture. Secondly, texture is a homogeneous property at some spatial scale *larger* than the resolution of the image.

Geometric texture can be defined analogously as spatially-varying surface displacements (in a normal direction) relative to an assumed underlying surface, which is smooth at some larger scale than the texture. Because the underlying surface of the scanned mesh model is *a priori* unknown, the displacement for every vertex is not known, either. However, variations in the local surface properties we have discussed previously can in practice be used instead of variations in displacements: these properties vary significantly between nearby vertices, and are homogeneous at some scale larger than the resolution of the mesh. This allows statistical methods such as first- and second-order statistics as used for image texture analysis to be extended to geometric textures. Using the same neighborhoods as for computing integral surface properties, we compute the following statistics on multiple-scale balls.

First-order statistics in an image measure the probability of a given grey value occurring at a randomly chosen location. We may apply the same idea to geometric properties, computing the first-order statistics for neighborhoods of various scales. Any local surface property L can have its value normalized, and discretized as ι , into B bins ($\iota = 0, \dots, B-1$). B is set to 32 in our algorithm. For each vertex v , a histogram of the values of ι at v and its neighborhood N is then computed using

$$H(\iota) = \frac{N_\iota}{|N|+1}; \iota = 0, \dots, B-1, \quad (3)$$

where N_ι is the number of vertices having value ι .

First-order statistical features such as mean, variance, energy and entropy at v can then be computed from the corresponding histogram for a geometric property L using the following equations:

Mean

$$S_1 = \frac{1}{|N|+1} \sum_{\{N,v\}} L, \quad (4)$$

Variance

$$S_2 = \frac{1}{|N|+1} \sum_{\{N,v\}} (L - S_1)^2, \quad (5)$$

Energy

$$S_3 = \sum_{\iota=0}^{B-1} H^2(\iota), \quad (6)$$

Entropy

$$S_4 = - \sum_{\iota=0}^{B-1} H(\iota) \log_2(H(\iota)). \quad (7)$$

Second-order statistics in an image measure the probability of a pair of pixel values occurring at some vector \mathbf{d} apart in the image; the probability function is represented by a co-occurrence matrix. As vertices on triangle mesh are

distributed irregularly, it is not possible to find vertex pairs at a fixed vector apart. Instead, we choose a fixed direction corresponding to some diagonal direction of the bounding box of the model, and a distance equal to the length of a typical triangle edge as a stand-in for the displacement vector \mathbf{d} . Then, for every vertex, we choose as its partner in a pair that particular 1-ring neighbor which is most nearly separated by this vector. (This approach relies on all triangles having a reasonably uniform size, which is the case for the meshes output by many scanning devices). The $B \times B$ co-occurrence matrix C_d for local surface property L is now defined as follows: $C_d(i, j)$ at vertex v is the number of occurrences of pairs of values i and j , at vertices with a displacement of approximately \mathbf{d} , with one vertex in the neighborhood of v :

$$C_d(i, j) = \left| \{ (\mathbf{v}', \mathbf{v}' + \mathbf{d}) : t(\mathbf{v}') = i, t(\mathbf{v}' + \mathbf{d}) = j \} \right|, \quad (8)$$

where $\mathbf{v}' \in \{N, \mathbf{v}\}$.

Second-order statistical geometric texture features such as energy, entropy, contrast and homogeneity can now be computed at v from the co-occurrence matrix using the following equations, where we now write just C for C_d for simplicity:

Energy

$$S_5 = \sum_{i=0}^{B-1} \sum_{j=0}^{B-1} C^2(i, j), \quad (9)$$

Entropy

$$S_6 = \sum_{i=0}^{B-1} \sum_{j=0}^{B-1} C(i, j) \log_2(C(i, j)), \quad (10)$$

Contrast

$$S_7 = \sum_{i=0}^{B-1} \sum_{j=0}^{B-1} (i - j)^2 C(i, j), \quad (11)$$

Homogeneity

$$S_8 = \sum_{i=0}^{B-1} \sum_{j=0}^{B-1} \frac{C(i, j)}{1 + (i - j)^2}. \quad (12)$$

4.2 SVM Classification

The SVM classifier is used to identify vertices of the mesh as belonging to relief or background. All integral and statistical properties can be combined together as its input, or a *feature selection* method can be used to choose a particular set of properties. The latter can save computation time, while still producing good classification results for a particular relief.

4.2.1 Basic SVM Procedure

Given objects in two classes, represented by a set of properties in a multidimensional space, an SVM classifier works by seeking the optimal hyperplane which separates objects from the two classes. The *optimal* hyperplane has maximum *margin*, i.e. the distance from the hyperplane to the closest data point for each class. Thus, the classifier is determined only by those data points which are at the margin: the *support vectors*. An SVM can solve nonlinear problems by using a set of nonlinear basis functions such as polynomial functions, radial basis functions, etc., to map the input features into a new space.

The problem to be solved can be stated in the following way: given m training samples (\mathbf{x}_i, y_i) where \mathbf{x}_i is a vector of property values at a vertex, and y_i is the class label for the vertex (*texture* or *background*), we wish to produce an SVM model such that y_i can be reliably predicted from \mathbf{x}_i for the remaining vertices.

A large number of software implementations of SVM have been developed. We use LIBSVM [20] which is an efficient open source SVM package written in C++, with a helpful guide to its use for practitioners [19].

The SVM procedure comprises the following five steps:

1. *Training data selection.* The user chooses a few representative regions of the mesh belonging to the background, and a few belong to the relief. These are specified by selecting a centre point and dragging to a radius, as shown in Fig. 4(a).
2. *Feature vector preparation.* The input features for the SVM can be any of the texture properties described in Section 4.1. We may use all of the properties, or a subset can be chosen using a feature selection method described later in Section 4.2.2. Each feature in the feature vector \mathbf{x}_i is linearly scaled into the range $[-1, +1]$ to prevent attributes with greater numerical ranges dominating those with smaller ranges, and to also numerical difficulties during the calculation [19].
3. *Choosing SVM kernel and parameters.* We use a radial-basis-function (RBF) kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, where \mathbf{x}_i and \mathbf{x}_j are input vectors. This means there are two main SVM parameters to be determined, γ and the penalty factor C (see [19]). The optimal values can be found by cross-validation, but our tests show that using the default values provided by LIBSVM gives good results.
4. *SVM model construction.* Using the labeled training data, the SVM uses the kernel function to map the input vectors into a high-dimensional feature space, and thence constructs a decision hyperplane to discriminate between the background and relief. This is the SVM model.
5. *SVM classification.* Given the model, the SVM is now used to classify all vertices. Note that these include the original training points, which can help to identify outliers that may have previously been assigned to the incorrect class in the training set. Doing so improves the accuracy of the classification.

4.2.2 Feature Selection

The sizes chosen for the vertex neighborhoods significantly affects the values computed for texture properties, and thereby affects success of classification. To avoid the issue of having to carefully choose a particular neighborhood size, we initially calculate properties using balls of multiple sizes. This generally results in more texture features than are really needed, and some are usually poor classifiers. However, different background textures may be better discriminated by different sets of texture properties. Although the SVM approach has the advantage of avoiding overfitting, it is still important to find the best feature combination, both to reduce computational costs, and for good generalization performance. Thus, we normally select the features to be used for classification.

Weston et al. [56] propose a feature selection method embedded in the SVM which works by minimizing bounds on leave-one-out error. Chen and Lin [4] combine several feature selection strategies with LIBSVM, although the strategies (e.g. F-score) are independent of the SVM. Here we use *sequential forward floating search* [43] as a classifier-independent feature selection strategy. It provides a good trade-off between the optimality of the feature set chosen, and efficiency of choosing the feature set. It starts from an empty feature set, and iteratively adds or removes features as appropriate to improve the *classification accuracy* achieved on the training set. Further feature selection methods are discussed in [16] and [25].

To measure classification accuracy during training and feature selection, a *confusion matrix* is used, which contains information about actual and predicted classifications. In our background-and-relief two-class model, it contains four elements as shown in Tab. 1.

	<i>Actually is Background</i>	<i>Actually is Relief</i>
<i>Classified as Background</i>	<i>a</i>	<i>b</i>
<i>Classified as Relief</i>	<i>c</i>	<i>d</i>

Tab. 1: Confusion matrix.

Here, a is the number of vertices correctly classified as background, b is the number of relief vertices classified as background, and so on. From the confusion matrix, a variety of accuracy or error measures can be calculated, such as true positive rate: $a/(a+c)$, true negative rate: $d/(b+d)$, accuracy: $(a+d)/(a+b+c+d)$, etc: see [14]. We choose to

optimize the Kappa statistic¹ (κ) [8, 14] which can be shown to lead to a high overall classification accuracy with few errors:

$$\kappa = \frac{(a+d) - (((a+c)(a+b) + (b+d)(c+d)) / M)}{M - (((a+c)(a+b) + (b+d)(c+d)) / M)}, \quad (13)$$

where $M = a + b + c + d$.

We now explain our use of sequential floating forward search for feature selection, using the confusion matrix based on the training data. Suppose there are D elements in the overall set of possible features which are available for use: $F = \{f_i; i=1, \dots, D\}$. Our task is to select a subset $G = \{g_i; i=1, \dots, k; g_i \in F\}$ with optimization criterion the Kappa statistic, κ . We do the following:

- 1: Initialization: start with an empty output set $G = F$.
- 2: Select the best feature: $g^+ = \operatorname{argmax}_{g \in \{F-G\}} \kappa(G_k + g)$
- 3: if adding it gives an improvement: $\kappa(G_k + g) > \kappa(G_k)$ then
- 4: Add this feature to the set: $G_{k+1} = G_k + g^+; k = k + 1$
- 5: else
- 6: return the solution: G
- 7: end if
- 8: Select the worst feature: $g^- = \operatorname{argmax}_{g \in \{G_k\}} \kappa(G_k - g)$
- 9: if deleting it gives an improvement: $\kappa(G_k - g^-) > \kappa(G_{k-1})$ then
- 10: Delete this feature from the set: $G_{k-1} = G_k - g^-; k = k - 1$
- 11: goto step 8
- 12: else
- 13: goto step 2
- 14: end if

4.3 Snake-Based Segmentation

We use a snake for segmentation on a textured background; the snake is represented as a set of connected straight line segments, joining an ordered list of discrete vertices (snaxels). We use a somewhat different approach to the snake used on a smooth background in our earlier work [30]. There, snake evolution has three basic phases, each with a particular goal: coarse evolution, contour refinement and final stabilization. Coarse evolution moves the snake quickly from its initial position far from the relief to the approximate relief contour. The refinement phase explores concavities in the relief contour. Final stabilization captures fine detail. For the textured background case, we divide the evolution process into two phases: *coarse evolution* and *contour optimization*.

Both evolution phases start with an original snake contour and produce a new contour according to some energy functional. The energy generally includes four terms: internal *tension energy* (E_t) and *bending energy* (E_b), and external *feature energy* (E_f) and *deflation energy* (E_d) terms. The basic difference between the snake used here, and on a smooth background, lies in different choices for the energy terms, especially the feature energy. We follow our earlier approach except as where noted.

4.3.1 Coarse Evolution

The initial contour is constructed by joining several user-chosen vertices on the background. The snake segment length is set to be slightly less than the approximate relief step height. This result provides a good starting snake for the following contour optimization phase, which works in a more local manner.

Our previous approach [30] to segmenting reliefs on a smooth background is based on detecting a step variation in height approximately equal to the relief height, and uses a feature energy based on the value of the planarity property. However, when the background is textured, large planarity variations occur on the background as well as at the edge of the relief, so planarity is no longer appropriate for defining feature energy. Instead, we define an energy based on

¹ Note that Kappa does not mean curvature in this paper. We use the symbol L_H for mean curvature.

the difference in classification as either background or relief, enabling the snake to move towards the relief and stop at the boundary we define the feature energy E_f^c based on classification at vertex v_i as:

$$E_f^c(i) = 0.5 \times \text{Label}(v_i), \quad (14)$$

where $\text{Label}(v_i)$ is the classification label for a vertex, with a value of -1 for a vertex classified as background and $+1$ for relief. When the snaxel inside a face or on an edge of the mesh, the label is computed by linear interpolation. E_f^c takes a minimum value of -0.5 for snaxels on the background, and a maximum 0.5 for snaxels on the relief. On areas (mainly) classified as background, the feature energy is almost the same everywhere, so the snake will move towards the relief under the constraint force. When snaxels approach the relief boundary, they will stop moving because of the larger feature energy on the relief side.

4.3.2 Contour Optimization

In this phase, the snake is sampled with a segment length equal to the average triangle edge length, to be able to better capture the fine details. The snake resulting from the coarse evolution stage is near the relief boundary, and is usually less than the relief step height away from the boundary. Thus, we can now replace the feature energy by another one designed to place the snake exactly where the planarity has a local maximum, i.e. at the relief boundary. The feature energy at vertex v_i used in this case is:

$$E_f^0(i) = -\gamma \cdot \text{sign}(P(v_i))(P(v_i))^2, \quad (15)$$

where γ is a normalization parameter, and P is the integral planarity computed over a ball with radius equal to the approximate relief step height h .

Unlike in our approach to reliefs on a smooth background, this energy term may cause the resulting snake position to lie outside, or just inside the relief, due to imperfect classification. To prevent such problems, we turn off the deflation energy, which drives the snake inwards, and may cause the snake to overrun. Instead, we introduce another external energy: an attraction energy term E_a which essentially enlarges the search area to a radius equal to the relief step height h , in order to help prevent the snake from coming to rest in local minima. E_a is given by

$$E_a(i) = -\gamma(P(v_i) - P(q_i))^2, \quad (16)$$

where γ and P are defined as in Eqn. 15, and q_i is the point on the mesh with maximum planarity value within a geodesic distance h along a mesh curve perpendicular to the snake at v_i .

E_a is only added to candidate snaxel points on the *same* side of the snake as q_i , to provide a force in the appropriate direction. Because E_a only has an effect inside a radius of h , it cannot be used to make the snake explore relief contour concavities which are deeper than h .

5. SMOOTHING APPROACH

5.1 Mesh Smoothing

We adopt the traditional Laplacian mesh smoothing method [13]. The smoothing process updates the mesh vertices according to

$$v_{new} = v + \frac{1}{|N|} \sum_{v' \in N} (v' - v). \quad (17)$$

Generally, several iterations are needed until the texture is sufficiently smoothed: e.g. 5 to 10 iterations suffice for the 'Bird' model shown later.

5.2 Snake Segmentation on the Smoothed Mesh

Iteratively smoothing the mesh removes the texture, but at the same time significant shape distortion and surface shrinkage may also result, so the relief boundary on the smoothed mesh is not at the same position as on the original mesh. Thus, we only apply the coarse evolution phase of our previous method [30] to find the relief boundary on the *smoothed* mesh.

5.3 Snake Optimization on the Original Mesh

Note that Laplacian smoothing keeps the mesh topology and simply modifies the locations of mesh points. After the snake has been coarsely located on the smoothed mesh, it is easy to map the snake back to the original model, using

barycentric coordinates, providing a contour close to the true relief boundary. The snake's location is now optimized on the *original* mesh in the same way as in Section 4.3.2.

6. EXPERIMENTS

We carried out experiments on a variety of relief models having different texture sizes and shapes, scanned by a Minolta Vivid VI-910. All computations were performed on a PC with 2.4GHz AMD Athlon CPU and 1GB of RAM.

6.1 Results using the Classification Approach

6.1.1 Choice of Properties

As computing many properties takes a long time, it is important not to compute properties which have little classification power. We thus performed various experiments to evaluate the usefulness of different properties. We used the user-supervised training data to evaluate how well each property can classify the regions in each example, using the Kappa statistic, κ , to assess performance. [14] suggests that poor accuracy corresponds to $\kappa < 0.4$; good classification accuracy to $0.4 < \kappa < 0.75$ and excellent classification to $\kappa > 0.75$. Correct prediction for every vertex would correspond to a value of $\kappa = 1.0$.

For each model, the features used in the SVM comprise the integral properties and statistical measures based on the three local surface properties over multi-scale ball neighborhoods. We used three ball sizes, giving a total of $9 \times 3 \times 3$ input features. The radii of the balls used were $0.25r$, $0.5r$ and r , where r is chosen by the user to be large enough to cover the characteristic size of the texture.

In Tab. 2, we give an example showing the Kappa statistic κ for each different feature for the 'Bird' model shown later in Fig. 5. Clearly, certain features give poor classification results, while other features give very good prediction, such as the second order energy of normal difference for balls of size 3.

We carried out further experiments on the SVM using three kinds of feature set: using all features, using features chosen by SFFS, and using features whose $\kappa > 0.75$. The corresponding classification results are illustrated in Figs. 4–6. For all three models and for all three kinds of feature sets, the classification accuracy on the training data is very close to the perfect value $\kappa = 1.0$. While using all the features is generally slightly better, SFFS always produces much smaller feature sets; however, these were different for the different models. Referring to the feature in the i^{th} row and j^{th} column of Tab. 2 as F_{ij} , SFFS produced the following feature sets for each example in turn: 'Men' model: $\{F_{36}, F_{12}\}$, 'Bird' model: $\{F_{69}, F_{39}, F_{12}\}$, 'Lady' model: $\{F_{73}, F_{89}, F_{31}\}$.

A comparison of computation times taken for performing classification using different feature sets is shown in Tab. 3; the most time-consuming part of our program is calculation of the texture properties at multiple scales. Doubtless our algorithm could be optimised significantly; for example, it took 25 minutes just to calculate the mean of planarity on 3 balls for all the vertices of the 'Bird' model shown in Fig. 5. When using SFFS, or only features with $\kappa > 0.75$, all texture properties only need to be calculated on the *training data*, and for the *whole model*, only a few *selected* properties need to be calculated. In particular, SFFS greatly reduces the calculation time required.

	Feature	Planarity			Curvature			Normal Difference		
		Ball-1	Ball-2	Ball-3	Ball-1	Ball-2	Ball-3	Ball-1	Ball-2	Ball-3
		1	2	3	4	5	6	7	8	9
1	Integral	-0.57	-0.57	-0.44	-0.57	-0.57	-0.14	-0.57	-0.57	-0.57
2	1 st Mean	-0.57	-0.57	-0.57	-0.57	-0.57	-0.43	-0.57	0.37	0.83
3	1 st Variance	0.63	0.56	0.34	0.75	0.81	0.75	-0.57	-0.57	-0.57
4	1 st Energy	0.71	0.80	0.84	-0.57	-0.57	-0.57	0.71	0.79	0.82
5	1 st Entropy	0.78	0.86	0.86	-0.57	-0.57	-0.57	0.71	0.80	0.79
6	2 nd Energy	-0.57	-0.57	-0.57	-0.57	-0.57	-0.57	0.77	0.86	0.91
7	2 nd Entropy	0.32	0.88	0.94	-0.57	-0.57	0.48	0.82	0.87	0.89
8	2 nd Contrast	0.79	0.83	0.87	-0.57	-0.57	-0.49	0.79	0.81	0.76
9	2 nd Homo	0.68	0.80	0.88	-0.57	-0.57	-0.57	0.80	0.85	0.89

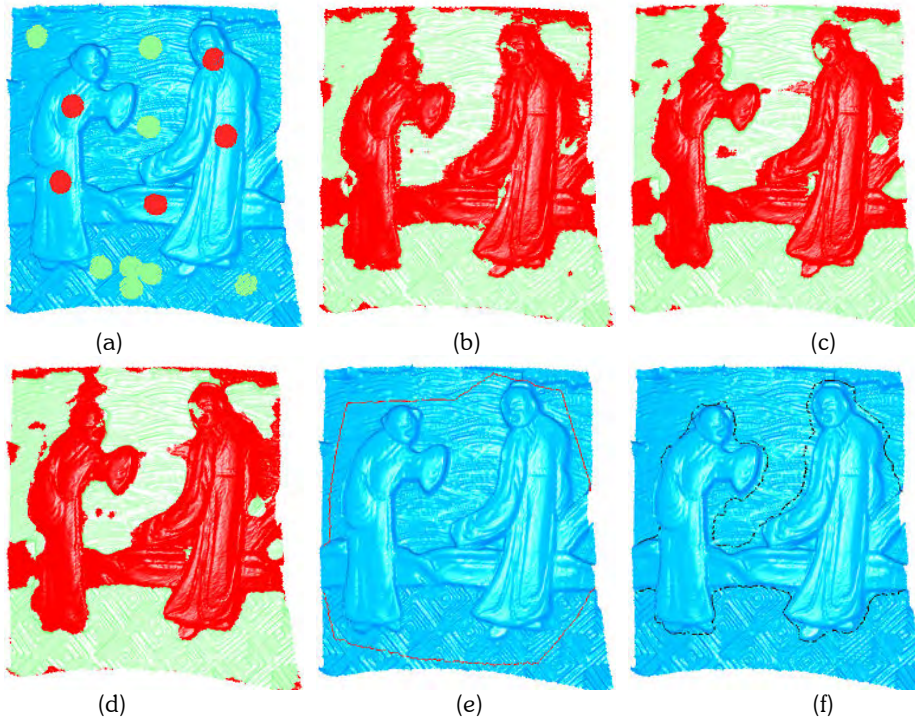
Tab. 2: Classification Accuracy on training data for 'Bird' Model.

Model	Vertices	Faces	Texture Training Vertices	Relief Training Vertices	All Features		SFFS		$\kappa > 0.75$	
					Properties (h)	SVM (min)	Properties (h)	SVM (min)	Properties (h)	SVM (min)
Men	70,284	139,380	2377	1535	6.8	1.3	1.3	6.0	3.4	2.1
Bird	55,955	110,945	3843	2209	5.2	1.3	1.1	8.7	2.4	2.6
Lady	62,404	123,748	2012	1596	6.1	1.2	1.5	8.0	2.6	2.3

Tab. 3: Time taken for SVM classification.

6.1.2 Segmentation Results

Fig. 4 illustrates segmentation of a 'Men' relief from a textured background using our classification approach. This model was scanned from a vase; note that there are actually two distinct textures in the background (corresponding to sky and floor). Fig. 4(a) shows the user-selected training data: the light (green) spots belong to the background and the dark (red) ones to the relief. These areas were chosen without particular care to select typical or interesting regions. The radius of the largest ball for property calculation, r , was set to 8 times the average triangle edge length for the whole model, which is about the same size as the sample spots in Fig. 4(a). Using all different sets of features in the SVM provided the classification results shown in Fig. 4(b)–(d). Using all features (b) resulted in slightly better classification than using SFFS (c) or features with $\kappa > 0.75$ (d). Fig. 4(e) shows the original snakes created from a few user-selected vertices; in this case we sought two separate open contours. The coarse relief contour obtained, based on the classification result of (b), is shown in Fig. 4(f). During the optimization phase, the relief height was estimated to be 2.4mm, giving the final contour shown in Fig. 4(g). Fig. 4(h) shows the final result from another viewpoint.



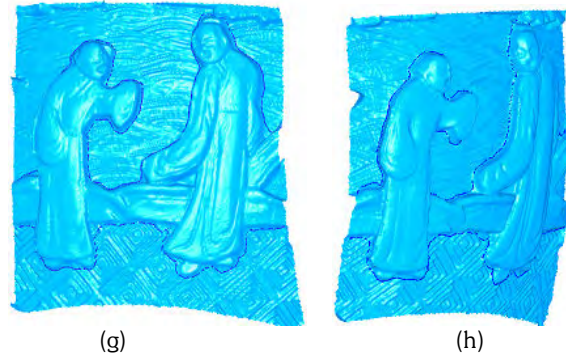
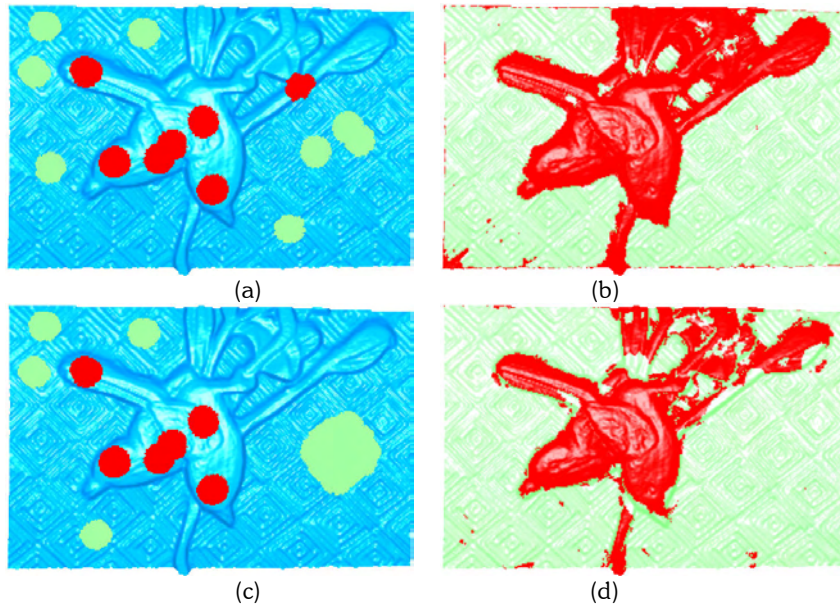


Fig. 4: Segmenting the ‘Men’ relief using the classification approach: (a) training data, (b) classification using 81 features, (c) classification using features selected by SFFS, (d) classification using features with $\kappa > 0.75$, (e) original contour, (f) coarse contour based on classification result of (b), (g) final contour, (h) final contour, another view.

Model	Boundary	Coarse Evolution		Optimization	
		Snaxels	Time (seconds)	Snaxels	Time (seconds)
Men	Upper	272	9	605	56
	Lower	145	7	246	26
Bird	Left	162	13	234	11
	Right	120	8	225	58
Lady	Left	220	11	385	22

Tab. 4: Time taken for snake computations.

Tab. 4 shows the computation times taken for the snake phases of the process, for this and subsequent examples; the classification times were given in Tab. 3.



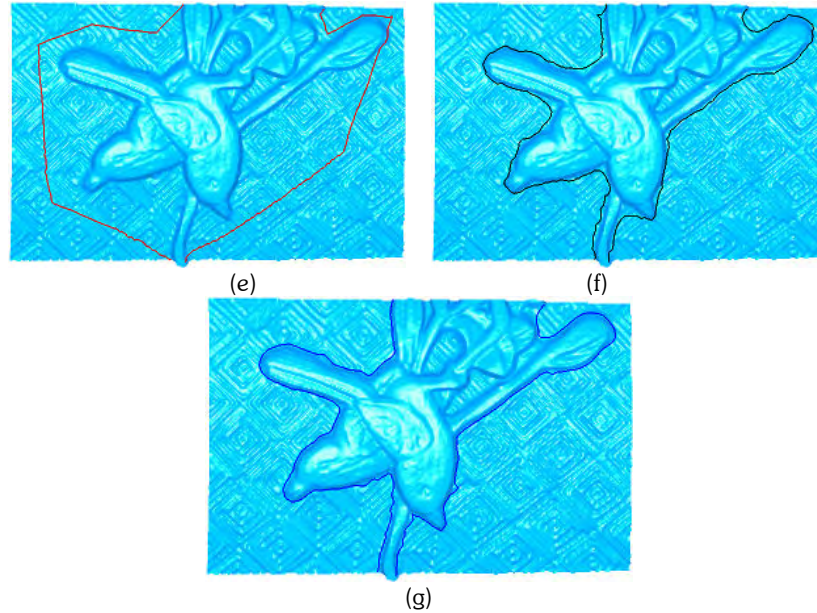
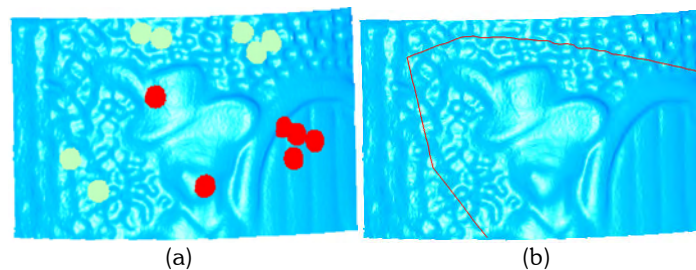


Fig. 5: Segmenting the 'Bird' relief using the classification approach: (a) training data, (b) classification using 81 features and training data (a), (c) another training data, (d) classification using 81 features and training data (c), (e) original contour, (f) coarse contour based on classification result of (b), (g) final contour.

Fig. 5 shows segmentation of a 'Bird' model using the classification approach. Again, r was set to 8 times the average edge length (2.2mm); the relief height h was estimated as 2.4mm. Fig. 5(a) shows the training data. Fig. 5(b) shows the classification result produced using all 81 features and the training data in (a) which were again randomly selected. A different choice of training data is shown in Fig. 5(c), where no training points were chosen on the narrow parts of the relief. In this case the classification result in Fig. 5(d) is poorer, and does not recognize such areas well. Fig. 5(e) shows the original user specified snakes—again we seek two open contours on left and right. Fig. 5(f) shows the snakes after coarse evolution using the classification result in (b), while (f) is the result after the optimization step.

Fig. 6 shows a final example. In this 'Lady' relief model, r was again 8 times the average edge length (1.7mm), and the relief height h was estimated as 1.2mm. Unlike the first two examples, the texture here is stochastic, and furthermore there is no sharp step at the relief boundary; the relief step height in some places is only as high as the depth of the pits in the texture. However, we still achieve reasonable segmentation results using the classification approach. Fig. 6(a) shows the training data and (b) shows the initial snake contour. Fig. 6(c) shows the classification results produced using all 81 features. Fig. 6(d) and (e) shows the snake its coarse and final states based on the classification result in (c). Fig. 6(f) shows the classification results produced using features selected by SFFS, and (g) and (h) are the snake results from (f). Although there are slight differences between (c) and (f), the final snake is nearly the same in each case.



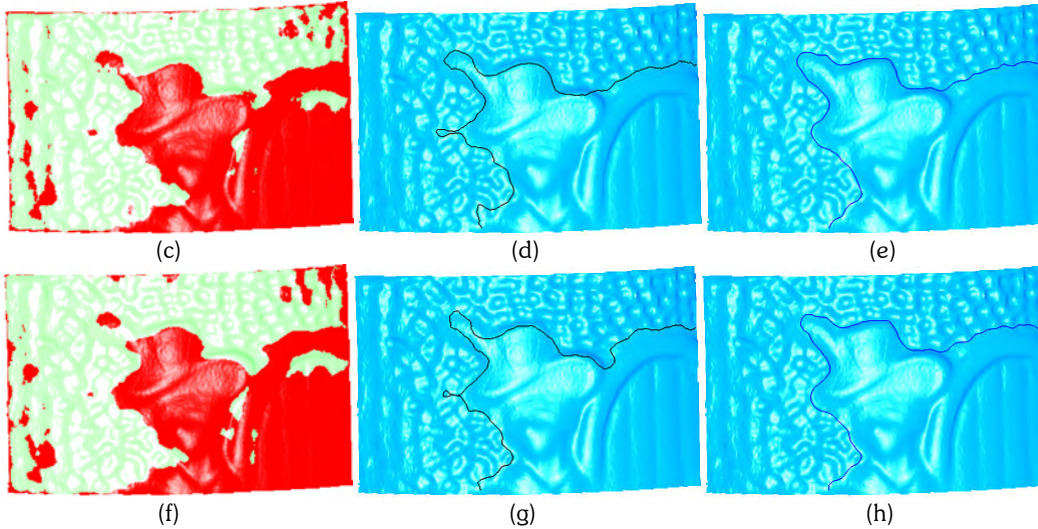


Fig. 6: Segmenting the 'Lady' relief using the classification approach: (a) training data, (b) original contour, (c) classification using 81 features, (d) and (e) corresponding coarse and final contours, (f) classification using features selected by SFFS, (g) and (h) corresponding coarse and final contours.

6.2 Results using the Smoothing Approach

The smoothing approach was also tested on the same three models. However, the 'Lady' model only has a weak step between the textured background and the relief, which is no higher than the detail within the texture. As a result, smoothing destroys this step before it smooths the background sufficiently, and the smoothing approach fails to produce a result as the smoothing approach cannot locate the relief boundary for this model.

Fig. 7 shows segmentation of the 'Bird' relief using the smoothing method. Fig. 7(a) shows the smoothed mesh after 5 iterations of Laplacian smoothing and the resulting snakes on either side. The outer red lines are the original contours specified by the user and the inner black lines are the extracted relief boundary on the smoothed model. These black lines mapped back to the original unsmoothed mesh are shown in Fig. 7(b); they are already very near the exact boundary. Fig. 7(c) shows the final contour after the snake optimization phase. Fig. 8 shows similar results for the 'Men' relief.

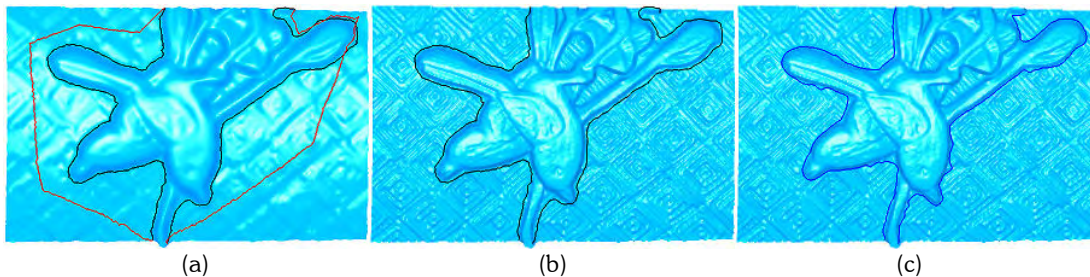


Fig. 7: Segmenting the 'Bird' relief using smoothing: (a) smoothed mesh and resulting snake on coarse mesh, (b) snake transferred to original mesh, (c) final optimized snake.

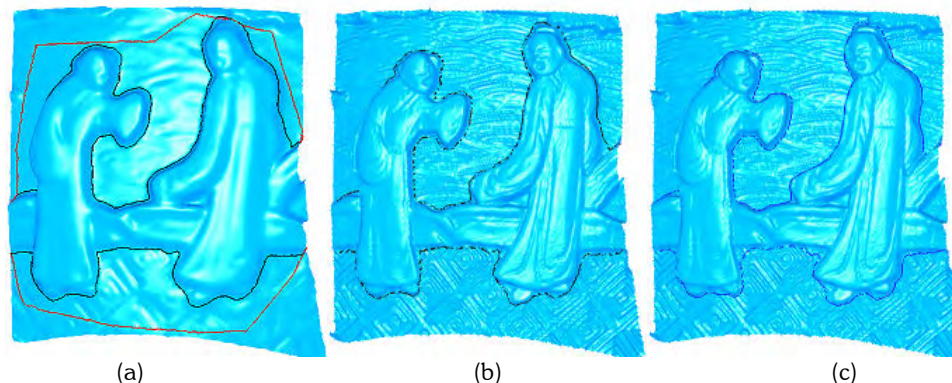


Fig. 8: Segmenting the 'Men' relief using smoothing: (a) smoothed mesh and resulting snake on coarse mesh, (b) snake transferred to original mesh, (c) final optimized snake.

The smoothing approach is much more efficient than the classification one. For all the models, the Laplacian smoothing step took only a few seconds and the snake evolution time was about 2 or 3 minutes.

7. CONCLUSIONS

Segmenting a relief from a textured background is a challenging task in relief reverse engineering. In this paper, two approaches have been proposed to deal with this task. Both are based on snakes, but to initially locate the contour, one uses a feature energy based on classification, while the other is based on mesh smoothing and detecting the step at the edge of the relief. From our experiments, the following conclusions can be drawn.

- Clearly, certain features give poor classification results for all examples, such as integral planarity for balls of size 1. Other features reliably give very good prediction such as homogeneity of normal difference for balls of size 3.
- Three feature selection strategies, using all features, features selected by SFFS, and features with $\kappa > 0.75$ were considered. Using all features can give slightly better results, but takes much longer. Using SFFS for feature selection on user-chosen training data significantly reduces the number of properties and computing time. Using features with $\kappa > 0.75$ is not recommend as it both takes a long time, and does not give significantly better results than SFFS.
- Clearly, the smoothing approach is much faster than the classification approach, and so for a given model, it is better to try this approach first if possible. However, it does not work well, if at all, if the relief contour has deep concavities, or if the relief step is no longer distinct after the textures have been smoothed.
- Overall, both approaches are useful when segmenting reliefs on textured backgrounds, but the results are not completely reliable and a modest amount of hand-editing may still be necessary.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the support of Delcam plc, including many helpful discussions with Richard Barratt and Steve Hobbs, and the support of EPSRC grant GR/T24425, for this work. We also wish to thank Xianfang Sun for helpful discussions and assistance with smoothing.

REFERENCES

- [1] Bhat, P.; Ingram, S.; Turk, G.: Geometric texture synthesis by example, In Eurographics Symposium on Geometry Processing, 2004, 43–46.
- [2] Bischoff, S.; Weyand, T.; Kobbelt, L.: Snakes on triangle meshes, In Bildverarbeitung für die Medizin, 2005, 208–212.
- [3] Chen, C.-Y.; Cheng, K.-Y.: A sharpness dependent filter for mesh smoothing, Computer Aided Geometric Design, 22(5), 2005, 376–391.
- [4] Chen, Y.-W.; Lin, C.-J.: Combining SVMs with various feature selection strategies, 2005, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [5] Clausi, D. A.; Deng, H.: Design-based texture feature fusion using Gabor filters and co-occurrence probabilities,

- IEEE Transactions on Image Processing 14(7), 2005, 925–936.
- [6] Clausi, D. A.; Yue, B.: Texture segmentation comparison using grey level co-occurrence probabilities and Markov random fields, In Proc. of the 17th International Conference on Pattern Recognition, 2004, 584–587.
 - [7] Cohen, L. D.; Cohen, I.: Finite element methods for active contour models and balloons for 2d and 3d images, IEEE Trans. Pattern Analysis and Machine Intelligence, 15(11), 1993, 1131–1147.
 - [8] Cohen, J.: A coefficient of agreement for nominal scales, Educational and Psychological Measurement, 20(1), 1960, 27–46.
 - [9] Cohen, L. D.: On active contour models and balloons, CVGIP: Image Understanding, 53(2), 1991, 211–218.
 - [10] Connors, R.; Harlow, C.: A theoretical comparison of texture algorithms, IEEE Transaction Pattern Analysis and Machine Intelligence, 2(3) 1980, 204–222.
 - [11] Deng, H.; Clausi, D. A.: Unsupervised image segmentation using a simple MRF model with a new implementation scheme, Pattern Recognition, 37(12) 2004, 2323–2335.
 - [12] Duda, R. O.; Hart, P. E.; Stork, D. G.: Pattern Classification, second ed. New York : John Wiley Sons, 2001.
 - [13] Field, D. A.: Laplacian smoothing and Delaunay triangulations, Communications in Numerical Methods in Engineering, 4, 1988, 709–712.
 - [14] Fielding, A.; and Bell, J; A review of methods for the assessment of prediction errors in conservation presence/absence models, Environmental Conservation, 24(1), 1997, 38–49.
 - [15] Fleishman, S.; Drori, I.; Cohen-Or, D.: Bilateral mesh denoising. ACM Trans. Graphics, 22(3), 2003, 950–953.
 - [16] Guyon, I.; Elisseeff, A.: An introduction to variable and feature selection, Journal of Machine Learning Research, 3, 2003, 1157–1182.
 - [17] Guyon, I.; Weston, J.; Barnhill, S.; Vapnik, V.: Gene selection for cancer classification using support vector machines, Machine Learning, 46(1-3) 2002, 389–422.
 - [18] Haralick, R.; Shanmugam, K.; Dinstein, I.: Textural features for image classification, IEEE Transactions on Systems, Man, and Cybernetics, SMC-3 (3), 1973, 610–621.
 - [19] Hsu, C.; Chang, C.-C.; Lin, C.-J.: A practical guide to support vector classification, 2003, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
 - [20] Hsu, C.; Chang, C.-C.; Lin, C.-J.: Libsvm: a library for support vector machines, 2004, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
 - [21] Joachims, T.: Text categorization with support vector machines: Learning with many relevant features, In Proceedings of the European Conference on Machine Learning, 1998, 137–142.
 - [22] Jung, M.; Kim, H.: Snaking across 3d meshes, Proceedings of the Computer Graphics and Applications, 12, 2004, 87–93.
 - [23] Kass, M.; Witkin, A.; Terzopoulos, D.: Snakes: Active contour models, International Journal of Computer Vision, 1(4), 1988, 321–331.
 - [24] Kim, K. I.; Jung, K.; Park, S. H.; Kim, H. J.: Support vector machines for texture classification, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(11), 2002, 1542–1550.
 - [25] Kohavi, R.; John, G.: Wrappers for feature selection, Artificial Intelligence, 1-2, 1997, 273–324.
 - [26] Lai, Y.; Hu, S.; Gu, D. X.; Martin, R. R.: Geometric texture synthesis and transfer via geometry, In ACM Solid and Physical Modeling, 2005, 15–26.
 - [27] Lai, Y.; Hu, S.; Martin, R. R.: Feature sensitive mesh segmentation, In ACM Solid and Physical Modeling, 2006, 17–26.
 - [28] Lee, Y.; Lee, S.: Geometric snakes for triangular meshes, Computer Graphics Forum, 21, 3, 2002, 229–238.
 - [29] Li, S.; Kwoka, J. T.; Zhua, H.; Wang, Y.: Texture classification using the support vector machines, Pattern Recognition, 36(12), 2003, 2883–2893.
 - [30] Liu, S.; Martin, R. R.; Langbein, F. C.; Rosin, P. L.: Segmenting reliefs on triangle meshes, In Proc. ACM Symp. Solid and Physical Modeling, ACM, 2006, 7–16.
 - [31] Manjunath, B. S.; Chellapa, R.: Unsupervised texture segmentation using Markov random field models, IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(5), 1991, 478–482.
 - [32] Mao, J.; Jain, A. K.: Textural classification and segmentation using multiresolution simultaneous autoregressive models, Pattern Recognition, 25(2), 1992, 173–188.
 - [33] Materka, A.; Strzelecki, M.: Texture analysis methods: a review, Technical Report University of Lodz, 1998, http://www.eletel.p.lodz.pl/cost/pdf_1.pdf.
 - [34] Meyer, M.; Desbrun, M.; Schroder, P.; Barr, A. H.: Discrete differential-geometry operators for triangulated 2-manifolds, In Visualization and Mathematics III, Springer-Verlag, Heidelberg, H. Hege and K. Polthier, Eds., 2003, 35–57.

- [35] Milroy, M. J.; Bradley, C.; Vickers, G. W.: Segmentation of a wrap-around model using an active contour, *Computer-Aided Design*, 29(4) 1997, 299–320.
- [36] Montiel, E.; Aguado, A. S.; Nixon, M. S.: Texture classification via conditional histograms, *Pattern Recognition Letters*, 26(11) 2005, 1740–1751.
- [37] Muneeswaran, K.; Ganesan, L.; Arumugam, S.; Soundar, K. R.: Texture image segmentation using combined features from spatial and spectral distribution, *Pattern Recognition Letters*, 27(7), 2005, 755–764.
- [38] Ohanian, P.; Dubes, R.: Performance evaluation for four classes of textural features, *Pattern Recognition*, 25(8), 1992, 819–833.
- [39] Osuna, E.; Freund, R.; Girosi, F.: Training support vector machines: An application to face detection, In *Proceedings of Computer Vision Pattern Recognition*, 1997, 130–136.
- [40] OuYang, D.; Feng, H.-Y.: On the normal vector estimation for point cloud data from smooth surfaces, *Computer-Aided Design*, 37(10), 2005, 1071–1079.
- [41] Page, D. L.; Sun, Y.; Koschan, A. F.; Paik, J.; Abidi, M. A.: Normal vector voting: Crease detection and curvature estimation on large, noisy meshes, *Graphical Models*, 64(3–4), 2002, 199–229.
- [42] Pottmann, H.; Huang, Q.-X.; Yang, Y.-L.; Kolpl, S.: Integral invariants for robust geometry processing, *Tech. Rep. 146*, Vienna Univ. of Techn., 2005, <http://www.geometrie.tuwien.ac.at/ig/papers/tr146.pdf>
- [43] Pudil, P.; Ferri, F.; Novovicova, J.; Kittler, J.: Floating search methods for feature selection with nonmonotonic criterion functions, In *Proceedings of the Twelfth International Conference on Pattern Recognition*, 1994, 279–283.
- [44] Randen, T.; Husoy, J. H.: Filtering for texture classification: A comparative study, *Pattern Recognition*, 21(4) 1999, 291–310.
- [45] Reed, T. R.; du Buf, J.: A review of recent texture segmentation and feature extraction techniques, *CVGIP: Image Understanding*, 57(5), 1993, 359–372.
- [46] Schneider, R.; Kobbelt, L.: Geometric fairing of irregular meshes for free-form surface design, *Computer Aided Geometric Design*, 18(4), 2001, 359–379.
- [47] Shamir, A.: A formulation of boundary mesh segmentation, In *Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium on (3DPVT'04)*, 2004, 82–89.
- [48] Shen, Y.; Barner, K. E.: Fuzzy vector median-based surface smoothing, *IEEE Trans. Visualization and Computer Graphics*, 10(3), 2004, 252–265.
- [49] Strand, J.; Taxt, T.: Local frequency features for texture classification, *Pattern Recognition*, 27(10), 1994, 1379–1406.
- [50] Taubin, G.: Estimating the tensor of curvature of a surface from a polyhedral approximation, In *Proc. 5th Intl. Conf. on Computer Vision*, 1995, 902–907.
- [51] Taubin, G.: A signal processing approach to fair surface design, In *SIGGRAPH'95 Conference Proceedings*, 1995, 351–358.
- [52] Tuceryan, M.; Jain, A. K.: Texture segmentation using Voronoi polygons, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2), 1990, 211–216.
- [53] Tuceryan, M.; Jain, A. K.: Texture analysis, In *The Handbook of Pattern Recognition and Computer Vision*, Chen C. H.; Pau L. F.; and Wang P. S. P., Eds., second ed. World Scientific Publishing Co., ch. 2.1, 207–248, 1998.
- [54] Vapnik, V.: *The Nature of Statistical Learning Theory*, Springer, New York, 1995.
- [55] Welch, W.; Witkin, A.: Free-form shape design using triangulated surfaces, In *Computer Graphics Proceedings, Annual Conference Series*, 1994, 247–256.
- [56] Weston, J.; Mukherjee, S.; Chapelle, O.; Pontil, M.; Poggio, T.; Vapnik, V.: Feature selection for SVMs, In *Proc. Advances Neural Information Processing Systems*, 2000, 668–674.
- [57] Weszka, J.; Dyer, C. R.; Rosenfeld, A.: A comparative study of texture measures for terrain classification, *IEEE Transactions on Systems, Man and Cybernetics* 6(4), 1976, 269–285.
- [58] Xu, C.; Prince, J. L.: Snakes, shapes, and gradient vector flow, *IEEE Trans. Image Processing*, 7(3), 1998, 359–369.
- [59] Xu, Q.; Yang, J.; Ding, S.: Color texture analysis using the wavelet-based hidden Markov model, *Pattern Recognition Letters*, 26(11), 2005, 1710–1719.
- [60] Yagou, H.; Ohtake, Y.; Belyaev, A. G.: Mesh smoothing via mean and median filtering applied to face normals. In *GMP 2002*, 124–131.