

# Segmenting Reliefs on Triangle Meshes

Shenglan Liu\*

Ralph R. Martin

Frank C. Langbein

Paul L. Rosin

School of Computer Science, Cardiff University  
{Shenglan.Liu, ralph, F.C.Langbein, Paul.Rosin}@cs.cf.ac.uk

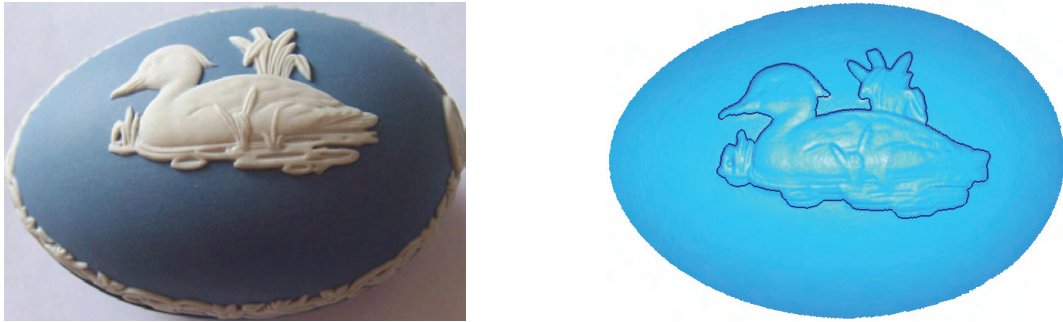


Figure 1: A porcelain relief and a corresponding segmented triangle mesh

## Abstract

Sculptural *reliefs* are widely used in various industries for purposes such as applying brands to packaging and decorating porcelain. In order to easily apply reliefs to CAD models, it is often desirable to reverse-engineer previously designed and manufactured reliefs. 3D scanners can generate triangle meshes from objects with reliefs; however, previous mesh segmentation work has not considered the particular problem of separation of reliefs from background. We consider here the specific case of segmenting a simple relief delimited by a single outer contour, which lies on a smooth, slowly varying background. Generally, such reliefs meet the surrounding surface in a small step, enabling us to devise a specific method for such relief segmentation.

We find the boundary between the background and the relief using an adaptive *snake*. It starts at a simple user-drawn contour, and is driven inwards by a collapsing force until it matches the relief's boundary. Our method is insensitive to the choice of the initial contour. The snake's limiting position is controlled by a feature energy term designed to find a step. A refinement strategy is then used to drive the snake into concavities of the relief contour.

We demonstrate operation of our algorithm using real scanned models with different relief contour shapes and triangle meshes with different resolutions.

**CR Categories:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

**Keywords:** Relief segmentation, snakes, mesh processing.

---

\*Corresponding author

## 1 Introduction

Complex decorative *reliefs* are often added to CAD models in such application areas as sign-making, packaging, and ceramics to make product designs more interesting, more characteristic of the company, or of higher intrinsic value. A typical example is the Wedgwood item shown on the left in Figure 1. New reliefs can be created from 2D artwork using software such as Delcam's ArtCAM, or can be hand-crafted, typically on a planar surface, and reverse-engineered using a 3D scanner. Extracting a relief from a planar background surface is a trivial task. However, in other cases it is desired to scan an existing relief lying on a curved object, rather than generating a new design. As in general such reliefs lie on freeform surfaces, reverse engineering methods are needed to separate the relief from the non-planar background, and to assist in the application of the relief to a different base surface.

A real world example of the need for reverse engineering reliefs occurs in the porcelain industry. When extending an existing range of porcelain, it is essential for any new item to exactly match existing designs on old items. At present it is necessary for a sculptor to hand-copy existing relief designs, a process that is time consuming and tedious for the artist, and expensive for the manufacturer—if indeed he can find suitably skilled workers. Clearly, the ability to automatically extract the relief, flatten it and then apply it to a new object would greatly reduce production time, and has the potential to produce better results.

To be more precise, in this paper, we define a *relief* to be extra material added locally to some underlying surface. This added material forms a surface with sculpted features clearly different from the underlying surface. In particular, we assume that the relief has a small height relative to the characteristic size of features on the underlying surface, which allows us to distinguish between the two surface regions. (There is nothing in principle, however, to prevent our methods being applied to 'negative reliefs', i.e., *embossing*, too, after making suitable minor changes).

There are various kinds of reliefs, and they can be imposed on diverse backgrounds. The simplest kind of relief is an *isolated* relief, which is bounded by a single outer contour (and which may also contain inner contours). See, for example, the duck in Figure 1. Other reliefs may be *cyclic* and wrap right around the object, such



Figure 2: Relief on a textured background

as the frieze around the rim in Figure 1. A more complex kind of relief is a *backgroundless* relief where the relief covers all or most of the surface of the model—we may envisage the model as having some smooth underlying surface over which a relief is applied everywhere. Relief processing also needs to take into account the nature of the *background* surface: the underlying surface maybe relatively simple and smooth, as in Figure 1, or more intricate—in more complex cases, the background can itself be textured, as in Figure 2. We could consider this object to be covered with a backgroundless relief, but alternatively, we might be interested in segmenting the bird and branch from the underlying pattern, which would then be considered to be a textured background.

In this paper we address a particular problem in reverse engineering of reliefs; other cases are to be considered in future work. We assume we are given as input a triangular mesh created from data points captured by a 3D scanner. Here we *only* consider the simple problem of segmenting the part of the mesh delimited by the outer contour of an *isolated* relief which lies on a *smooth* and slowly varying background. There may be multiple isolated reliefs present on the original object; we assume that the user draws a rough *closed contour* on the mesh around one relief to indicate which relief is to be segmented. This is done by specifying a few points on the mesh which are then automatically joined across the mesh; these may be relatively far from the relief. We can also handle the case where the relief is bounded by an *open contour*, as seen for example in Figure 7(c); in this case the user must indicate on which side of the initial contour the relief lies. We hope to extend our results to more complex reliefs and backgrounds in future, and so the approach taken has future generalisability in mind.

Our method uses a *snake* which starts from the user-drawn contour, and evolves until it matches the boundary of the relief. We adapt the snake to suit the mesh resolution, and allow most of the parameters to be determined automatically, minimising the need for non-technical users to have to understand and specify control parameters, whilst also providing high performance.

In Sections 2 and 3 we review closely related work: we briefly discuss the original idea of snakes as used for segmenting images, and their generalisation to 3D surfaces. In Section 4 we explain the novel contributions of this paper. The following Sections give details of our method: Section 5 explains the energy terms we use to control the snake and Section 6 describes the evolution process for the snake, Section 7 gives further implementation details, especially of the parameters used to control the method. Experiments using such snakes show that they can quickly and effectively determine isolated reliefs, as discussed in Section 8. Section 9 concludes the paper and gives various ideas for future extensions of our approach to other relief types.

## 2 Snakes on 2D Images

*Snakes* (or, more formally, *active contour models*) were originally proposed by Kass et al. [1988], and have been widely used for image segmentation and motion tracking in computer vision and image analysis. The basic idea is to deform a starting curve to an energy-minimising position under the influence of internal and external forces, in order to detect or follow features in an image or image sequence. Internal forces coming from the curve itself shrink and smooth the curve. External forces derived from the image help to drive the curve toward the desired features of interest using an iterative process. The snake evolves until an equilibrium of all forces is reached, which is equivalent to a minimum of the energy function.

There are two main limitations of the original snake approach. Firstly, the initial contour needs to be quite close to the object, otherwise the snake may converge to the wrong result, getting stuck in a local minimum. Secondly, when the object being sought is non-convex, snakes have difficulties in following the object's boundary in concavities. Both of these difficulties can also be encountered when using snakes for relief contour extraction.

If the snake is too sensitive to choice of initial contour, this forces the user to trace a starting contour which reasonably follows the complicated concavities and convexities of the relief's boundary, which is tedious and error-prone. In practice, concavities are very common on relief contours, and often they can be very deep and narrow. Various methods have been proposed to address the problem of initial contour sensitivity by modifying the external forces. The 'balloon' model [Cohen 1991] adds an additional pressure force (which can be inflationary or deflationary) to make the contour behave like a balloon. This helps avoid the contour getting stuck at weak features caused mainly by noise, and aiding convergence onto a more distant strong boundary. However, this introduces the new problem of choosing an appropriate pressure force.

Cohen and Cohen [1993] suggests employing an attractive force derived from the edge map of the image to provide a larger capture range for the snake in order to address the limitation on placement of the initial contour. Xu and Prince [1998] employs a gradient vector flow, an external force computed as a diffusion of the gradient vectors of a gray-level or binary edge map derived from the image, to address both limitations.

## 3 Snakes on 3D Surfaces

When extending the ideas of snakes from 2D images to 3D mesh surfaces, various problems arise. The first is the greater complexity caused by the connectivity of the mesh not being regular, unlike a 2D grid of pixels. Secondly, the internal energy terms caused by the internal forces must be defined in local surface tangent planes, rather than a global plane, and care must be taken when forces are added. Thirdly, the extra forces need to be calculated based on mesh features, instead of intensity image features. Generally, we require more complex energy terms to describe mesh features, and certain extended extra forces proposed for image snakes are difficult to directly adapt to meshes. Finally, the snake must be restricted to lie on the mesh when the energy function is minimised, thus leading to a constrained rather than unconstrained minimisation problem.

Milroy et al. [1997] extended snakes to 3D surfaces to perform surface segmentation. Their approach links curvature extrema points on an *orthogonal cross-section model*, a surface mesh that traces an object's contours with closed curves running in the  $x$ ,  $y$ , and  $z$  directions. They extended the definition of the internal energy to support

a discretised snake with irregular segment lengths in 3D. Two extra forces, an inflation force and an attraction force, were introduced to inflate contours, and attract small, closed, user-defined contours to the curvature extrema points. A greedy algorithm was used to minimise the energy.

Snakes on triangular meshes have been used to detect features where a certain property such as curvature changes drastically, allowing the detection of sharp edges and peak vertices [Lee and Lee 2002], for example. The authors define the external feature energy on the mesh in terms of the variation of normal directions between neighbouring faces. They parameterise the surface region surrounding the snake, then minimise the energy function and compute the motion of the snake in 2D, and finally remap the snake back onto the 3D mesh.

Other more recent research has also considered snakes for triangular meshes. Jung and Kim [2004] use snakes to find features related to Gaussian curvature. They move snakes step-by-step, from one mesh vertex to another, and give them the ability to change topology, allowing a snake to split into multiple separate snakes as appropriate. Based on their framework of parameterisation-free active contour models [2004], Bischoff et al. [2005] present a new representation and method for evolving snakes on triangular meshes. Their method enables collision detection and supports topological controls such as snake merging and splitting, in this case constraining the vertices of the snakes to move on mesh edges.

## 4 Novelty

Many methods have been proposed for segmentation of meshes; some of these try to cut a mesh into natural pieces (such as the fingers on a hand), whereas others try to find surface features such as sharp edges. As just one example, we cite the work by Funkhouser et al. [2004], which uses a least-cost path for mesh segmentation to find the natural seams of the mesh. Although snakes have been previously used for problems such as feature detection and segmentation of meshes, previously reported work is not particularly tailored to detecting reliefs on an underlying surface. Careful choice of appropriate forces for this specific problem will lead to better results than applying a general method, or a method which looks for, e.g., sharp edges. Note that a relief may meet the underlying surface in a sharp step, or the relief may blend with the underlying surface over a short distance. This is a different kind of feature from a sharp edge and warrants a specific type of feature energy term particular to relief segmentation. We describe our specific energy term for detecting a relief boundary in Section 5.3.

As mentioned above, other general issues are that snakes can be sensitive to the choice of initial contour, and can have particular difficulties with finding concavities in contours such as those found near the duck’s neck in Figure 1. Although sensitivity to initial contours has been tackled in previous work through the use of an external pressure force [Cohen 1991; Milroy et al. 1997], the force strength has to be set carefully if the equilibrium contour is to end up in the desired place, rather than overrunning, or ceasing to move inward too soon. The method proposed in Xu and Prince [1998] is able to move snakes into concavities in images, but is not directly applicable to 3D surface meshes. Our approach, on the other hand, works with a crudely specified initial contour which does not have to be close to the relief contour—it simply has to separate the desired relief from other possible reliefs.

Furthermore, previous methods based on snakes have required several parameters to be carefully controlled so that the snake’s movement leads to the desired results. In many cases appropriate param-

eter choices are data-dependent and can require considerable tuning by users to get good results. However, we expect many users of relief segmentation will not have a technical background, making it hard for them to understand how to tune the method for best results. Much time can be wasted in making a series of trials to find the best parameter settings. We thus have devised an approach which keeps such tuning to a minimum, and, where possible, we derive parameter settings from the model directly, without user interaction. Other remaining parameters have intuitively obvious meanings and effects.

Taking these points into account, we introduce a novel adaptive snake-based approach suitable for relief contour detection on a triangular mesh. Our main contributions are:

1. A feature energy term specifically designed for relief contour segmentation.
2. A deflation force with strength determined in such a way as to make the snake insensitive to choice of initial contour.
3. A snake refinement phase designed to make the snake explore relief contour concavities.
4. A denoising term in the deflation energy and use of bilateral filtering of feature properties to enable the snake to robustly traverse a noisy background.

The next several Sections present the details of our approach to using snakes for relief contour detection on triangular meshes.

## 5 Energy Terms

We now introduce the snake used; in this Section we define all the energy terms used and explain the resulting energy minimisation problem. Section 5.1 describes the energy functional used and approach to solving the energy minimisation problem. Details of the internal energy terms and external energy terms are explained separately in Sections 5.2 and 5.3.

### 5.1 Energy Functional

A parametric snake in 3D may be represented by a curve  $v(s) = (x(s), y(s), z(s))$ , where  $s$  represents arc-length. Following previous researchers’ ideas [Williams and Shah 1992; Milroy et al. 1997], the snake on the triangular mesh proposed in this paper is approximated using connected straight line segments, joined by an ordered list of discrete vertices (*snaxels*):

$$v(s) = [v_i], \quad i = 0, \dots, n-1 \quad (1)$$

where  $v_i$  is a point on the mesh lying either on a mesh vertex, an edge or a face. The list is a circular list for a closed relief contour:  $v_0$  and  $v_{n-1}$  denote the same snaxel, or it may be an ordinary list for an open relief contour. The distances between snaxels need not be equal, even approximately. The energy functional representing the energy of the snake can be written as

$$E = \int E_{int}(v(s)) + E_{ext}(v(s)) ds, \quad (2)$$

and depends on an *internal energy* term and an *external energy* term. The *internal energy*  $E_{int}$  depends on the snake itself. It is made up of two energy terms, representing *tension energy* and *bending energy*; minimising  $E_{int}$  makes the snake shrink and straighten (which are to some extent conflicting requirements).

These drive the snake inwards, and also prevent it from locally bending in and out too much. The *external energy*  $E_{ext}$  depends on the mesh. It creates extra forces which move and deform the snake to make it best fit the features sought; these forces are described in detail later. The forces arising from these various energy terms drive the snake to a position of lower overall energy. Ultimately, this should make the snake stabilise at the desired contour, located at a local minimum of energy.

In image processing, *variational calculus* [Kass et al. 1988] is widely used to determine the minimum energy of Eqn. 2. However, previous formulations cannot immediately be applied to the mesh case as this formalism does not permit the inclusion of the necessary *hard constraint* that the snake must lie on the mesh and not move off it. Lee and Lee [2002] handles this problem by parameterising the mesh surrounding the snake and then minimising the energy in 2D. However, parametrisation introduces an additional complexity, and may lead to unstable or inaccurate results in highly curved regions. Note that such regions at the boundary of the relief are precisely those we wish to find in our application!

A greedy algorithm for image snakes proposed by Williams and Shah [1992] provides a formulation for incorporating hard constraints quickly and easily. In this approach, the energy function is computed for every snaxel and each of its neighbouring pixels in the image, and the pixel (neighbour or current snaxel) which has the lowest energy is chosen as the new position of the snaxel. The energy of the snake is minimised one snaxel at a time and the whole snake stabilises when every snaxel is located at a local minimum of energy. Such a greedy method has also been used by Milroy et al. [1997] on a 3D mesh. The energy is evaluated at two adjacent locations on the surface perpendicular to the current snaxel boundary direction: this is justified in that any force parallel to the tangent direction of the snake affects the distribution of snaxels along the snake but does not change the snake's shape. The new positions of the snaxels are computed on a quadratic surface which locally approximates the orthogonal cross-section surface at every vertex. A greedy algorithm was also used by Jung and Kim [2004] on triangle meshes; they consider as candidate new snaxel positions the 1-ring neighbourhood vertices for each snaxel (which in their algorithm lie at mesh vertices).

We adopt Milroy's method, and use a greedy algorithm. For each snaxel, we consider two new candidate snaxel positions, at fixed distances from the snake and approximately perpendicular to it. In practice these candidate locations are chosen on the mesh along the directions which bisect the two edges linking the current snaxel, and the previous and next ones along the snake. We allow snaxels to be anywhere on the mesh, not at mesh vertices or on mesh edges. The snaxel  $v_i$  and the two candidate points,  $v_i^*$  and  $v_i^{**}$ , are shown in Figure 3. Snaxel  $v_i$  either moves to one of these two points or remains still depending on which of these three points has the lowest energy.

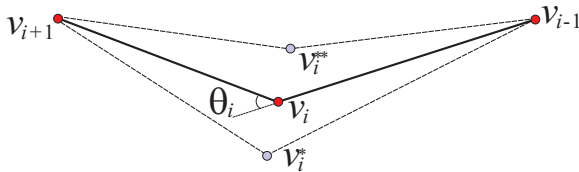


Figure 3: Candidate new snaxel positions

## 5.2 Internal Energy

The internal energy imposes internal shrinking and straightening forces on the snake which make the snake act both as a spring and as a flexible rod. It includes a *tension energy* term,  $E_t$ , and a *bending energy* term,  $E_b$ :  $E_{int} = E_t + E_b$ . We follow Milroy et al. [1997] in using the following energy terms associated with  $v_i$ : a first-order term which encourages the snake to be short,

$$E_t(i) = \frac{1}{4} \alpha (|v_{i-1}v_i| + |v_iv_{i+1}|), \quad (3)$$

and a second-order term representing the curvature of the snake, which discourages a large angle between snaxel segments, keeping it locally smooth,

$$E_b(i) = \beta \frac{1 - \cos \theta_i}{(|v_{i-1}v_i| + |v_iv_{i+1}|)^2}. \quad (4)$$

The constants  $\alpha$  and  $\beta$  are used to normalise the energy terms, giving each energy a value between 0 and 1.  $\theta_i$  is the angle between  $v_{i-1}v_i$  and  $v_iv_{i+1}$  as shown in Figure 3.

## 5.3 External Energy

The external energy's purpose is to drive the snake towards the relief boundary and localise the snake at it. Hence, the external energy should have a minimum at the boundary such that the snake converges towards it. For this we utilise a *feature energy* term  $E_f$  and a *deflation energy* term  $E_d$ . The external energy is:  $E_{ext} = E_f + E_d$ . The deflation energy helps the snake to contract from the user-drawn initial contour which may be at a large distance from the relief boundary and also to elongate the snake into concavities. The feature energy helps to create a counter-force to the deflation force to stop the snake at the relief boundary.

We first consider the feature energy  $E_f$  designed to locate the relief contour. We do so by presenting three versions of the feature energy. A general representation is given first, and then we discuss how a revised one based on *bilateral filtering* can overcome local noise. Finally we present the version using in our program which is specifically tailored to finding the particular feature at the edge of a relief, which is at a higher level the underlying surface. We finish this section by considering the deflation energy term.

### 5.3.1 General Definition of Feature Energy

In order to detect general geometric features in a mesh, energies can be defined in terms of properties such as planarity, variation in normal direction, or curvature, measured at vertices of the mesh. The planarity  $P(v)$  of a vertex  $v$  is defined as the signed distance between  $v$  and some plane fitted to its neighbours, where a positive value represents a vertex in a concave region and a negative value represents a vertex in a convex region. Given the planarity at mesh vertices, the planarity of points inside mesh faces can be determined by linear interpolation. Planarity is a simple measure to compute and provides an easy way to find such features as sharp edges.

For such general features, based on the idea of image feature energy, we may define  $E_f$  at a snaxel  $v_i$  as

$$E_f^1(i) = -\gamma (P(v_i))^2, \quad (5)$$

where  $\gamma$  is a normalisation parameter. This gives a large negative energy where the surface is locally non-planar (as might occur near the edge of a relief).

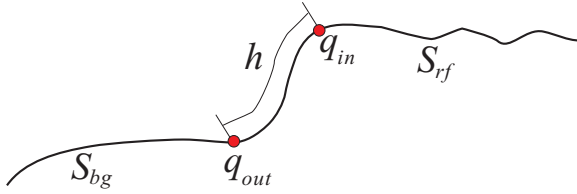


Figure 4: Cross-section of mesh near relief contour

### 5.3.2 Feature Energy for Noisy Surfaces

To reduce the effects of noise, mesh smoothing generally needs to be done before the properties like planarity are calculated. *Bilateral filtering*, proposed for images, is a one-sided filter derived from Gaussian blurring, which on the one hand smooths similar data while avoiding averaging dissimilar values, preserving steps in values [Tomasi and Manduchi 1998]. Extending this idea to meshes, Fleishman et al. [2003] yields a successful mesh-smoothing method which removes noise while preserving features. In the particular case of planarity, the bilaterally filtered value of planarity,  $\hat{P}(v)$ , at point  $v$  is defined by

$$\hat{P}(v) = \frac{\sum_{v' \in N(v)} W_c(|v - v'|) W_s(P(v) - P(v')) P(v')}{\sum_{v' \in N(v)} W_c(|v - v'|) W_s(P(v) - P(v'))}, \quad (6)$$

where  $N(v)$  is a neighbourhood of  $v$ ,  $W_c$  is a Gaussian filter with standard deviation  $\sigma_c$ , given by  $W_c(x) = \exp(-x^2/(2\sigma_c^2))$ , and  $W_s$  is a similarity weight function, with standard deviation  $\sigma_s$  that penalises large variation in planarity, given by  $W_s(x) = \exp(-x^2/(2\sigma_s^2))$ . Settings for these parameters are discussed in Section 7.2.1.

We may now revise the feature energy and replace Eqn. 5 by

$$E_f^2(i) = -\gamma (\hat{P}(v_i))^2, \quad (7)$$

which will reduce the tendency of the snake to get stuck at local minima, caused by noise in the background region due to measurement errors.

### 5.3.3 Relief Step Feature Energy

When applied to a relief contour, Eqn. 7 can readily detect a sharp step boundary between a relief and the background, but does not work so well if the relief meets the background in a more gentle step. Consider how a cross section of the mesh might appear near the boundary of a relief; an example is shown in Figure 4.  $S_{bg}$  represents the background surface and  $S_{rf}$  is the relief surface. A small step joins the two surfaces and causes two local feature energy extrema at the boundary of the relief: the outer edge of the step located at  $q_{out}$  is represented by a local *maximum* of positive planarity in the sectional view, while the inner edge of the step located at  $q_{in}$  has a local *minimum* of negative planarity. This leads to  $E_f^1$  or  $E_f^2$  having two minima in the vicinity of the edge of the relief. Thus, we make a further modification to Eqn. 7 to give the final version of feature energy used in our algorithm. The key idea is to take advantage of the planarity signature expected near the edge of a relief. Although the relief itself may have quite a variable height, usually it will have a characteristic *step height* above the background surface where it *meets the background*. Suppose this height is  $h$ —see Figure 4. We assume that it can be measured, or estimated by the user. We use

this as a typical distance in the further modified definition of the feature energy function:

$$E_f^3(i) = \begin{cases} -\gamma (\hat{P}(v_i) - \hat{P}(q_i))^2 & \text{if } \hat{P}(v_i) > \hat{P}(q_i); \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

Suppose the point  $q_i$  is a point on the mesh which is located inside the snake at geodesic distance  $h$  from snaxel  $v_i$ , measured perpendicular to the snake at  $v_i$ . When the snake is at some point of the background surface,  $v_i$  and  $q_i$  are expected to have nearly the same planarity, leading to a feature energy close to zero. When the snake is at the outer contour of the relief boundary,  $v_i$  will be the point having the local maximum planarity and  $q_i$  will have the local minimum planarity, leading to a very large negative feature energy.

However, in practice the relief may have a complicated cross-section—for example, it may rapidly go down again just inside the contour. Thus, just choosing one particular position for  $q_i$  may not lead to stable results. Instead, we use a small set of candidate positions for  $q_i$  at distances  $(1 - \epsilon)h$  to  $(1 + \epsilon)h$  from  $v_i$ , and select the location of  $q_i$  as the point with minimum planarity value for use in determining the energy. Generally,  $\epsilon$  is given a fixed value 0.2 in our algorithm. Note that even if a model has large variations of relief step height, satisfactory results can be obtained by setting  $h$  as the minimum relief step height: note that point  $q_{out}$  in Figure 4 will still be the first location of local minimum energy encountered by the snake.

In practice we observe that using the function  $E_f^3$  as the feature energy locates the relief contour very well when either the bottom or the top of the step is a sharp edge, and still produces good results when the relief blends into the background. It works well in practice as shown by the examples in Figures 6–8.

### 5.3.4 Deflation Energy

For the *external energy*, in addition to the feature energy, we also use a *deflation energy*  $E_d$  producing an extra force which tries to force the snake inwards. If global optimisation were used, the tension energy term in Eqn. 3 would make the snake shorter, but unfortunately this does not occur in a greedy algorithm which moves one snaxel at a time and only considers its immediate neighbours. The tension energy in this case simply tends to make the snake flat rather than shrink.

Our deflationary force is similar to that used in the *balloon model* [Cohen 1991], but it dynamically balances the internal energy taking into account the length of each snake segment and the speed of movement of the snake. Generally, the contour of a relief, when considered as a curve on an underlying surface, is composed of convex regions (e.g. around the beak of the duck in Figure 1), concave regions (e.g. around the neck of the duck in Figure 1) and flat regions (neither especially convex nor concave). Let us reconsider  $E_{int}$ . Snaxel  $v_i$  gets a smaller internal energy if it is moved towards the line between its neighbours. The result is that the snake does not move further inwards in concave regions, or even in flat regions, if we only use the internal energy alone: it would be moving to a position of higher energy. We must add a deflationary energy term to overcome this problem.

The deflationary energy  $E_d$  is defined to produce a corresponding force which acts inwards. The minimum deflationary force at a given point  $v_i$  is required when  $v_{i-1}$ ,  $v_i$  and  $v_{i+1}$  lie in a straight line, and in such cases the force should be just sufficient to disrupt the flat status, while not being so large as to cause the snake to overrun the relief boundary. If correctly balanced, the snake will continue

to move until it meets the relief boundary, but will not enter concavities. (We discuss later how the snake is forced into concavities). For conciseness, let  $l_i$  be the length  $0.5(|v_{i-1}v_i| + |v_iv_{i+1}|)$ ; we start evolution with all snaxels having values  $l_i = l$ , some desired initial value, and add or remove extra snaxels as evolution proceeds to keep snaxels about  $l$  apart, as explained in detail later. Suppose the movement step size for the snaxel  $v_i$  is  $\tau l_i$ . The just sufficient deflationary energy required can be derived from Eqns. 3 and 4. Taking into account that the background may be somewhat noisy, we define  $E_d$  at the inside candidate point for snaxel  $v_i$  to be:

$$E_d(i) = -\frac{1}{2}\alpha(\sqrt{1+\tau^2}-1)l_i - 2\beta\frac{\tau^2}{(1+\tau^2)^2l_i^2} - \gamma\sigma^2, \quad (9)$$

where  $\sigma$  is the standard deviation of the planarity of the background surface, which can be estimated using points on the initial contour on the background as indicated by the user, and  $\gamma$  is the same parameter as in Eqn. 7. This equation comprises three terms. The first two resist the internal energy and the term  $-\gamma\sigma^2$  eliminates the influence of any noise on the background surface.

## 6 Evolution

We now discuss how the snake moves from its initial contour to the final relief contour, which we call its *evolution*. The evolution of the snake occurs in three phases: firstly, *coarse evolution*, secondly, *contour refinement*, and thirdly, *final stabilisation*. Coarse evolution is used to approach the relief contour quickly. The snake stabilises close to most of the relief boundary except near concavities. Coarse evolution is carried out using large snake segments for efficiency. Contour refinement is then used both to explore the concavities, and to better capture fine detail of the contour. Smaller snake segments are used, and the internal energy is decreased. Final stabilisation is then used to accurately locate the snake at the relief contour and produce a smooth result.

In one evolutionary step, every snaxel is moved inwards or outwards a step, or kept still, depending on which of these three positions has minimum energy. Then a postprocessing process is applied before the next step to keep the snake regular, as done by Milroy et al. [1997]. First, any sharp notches which are smaller than a given sharpest angle are removed. Snaxels can be randomly distributed so without this it would be possible for the snake to fold over itself: there is no explicit control of the topology of the snake in our algorithm. Secondly, the length of each snake segment is kept within a desired range by merging any snaxels which are too short, and subdividing ones which are too long. Evolution stops after the snake has converged to a position when no snaxels move (or may be aborted if the iteration count exceeds some large number).

### 6.1 Coarse Evolution

Because the initial contour is far from the relief contour, we initially aim to move quickly across the background triangles on the mesh, while avoiding becoming stuck on local noise. During this phase, we define energy to be the internal energy, plus the feature energy and deflation energy. After the snake has stabilised, it will have reached a position which matches the relief boundary except near some of the concavities, as shown for example in Figure 6(b).

*Long* snake segments are used at this stage, causing the snake to move quickly and capture the relief contour coarsely. Given a relief with step height  $h$ , we use a discrete curve with segment lengths about  $h$  to coarsely represent the relief boundary.

For efficiency, the status of whether each snaxel has moved or not during the current iteration is stored. If a given snaxel and its two nearest neighbours did not move on the previous iteration, we know that the centre snaxel must remain still during the current iteration, so we can save time by not bothering to compute its energy.

It is possible for small oscillations to occur in the positions of snaxels. We check after every 10 iterations whether the distance moved by each snaxel is less than a small amount (we use  $2\tau l$ , twice the movement step size), and if so, the snaxel is locked in position, to avoid it oscillating forwards and backwards.

### 6.2 Contour Refinement

After the snake has stabilised at the coarse contour, we now refine it, both to more accurately capture the details of the model, and to drive it into any concavities. We change the energy terms used, and the snake segment length, in this step.

Using the energy terms used for the coarse contour does not permit the snake to explore concavities, as its internal energy term prevents it from locally bending or expanding enough. Thus, we now modify the energy used to drive evolution.

Firstly, we adjust the deflation energy given in Eqn. 9, replacing the term  $-\gamma\sigma^2$  by 0. This term is useful when coarsely finding the contour, as it allows us to step over noise in the background. However, when we are near the relief contour, we deactivate this term to prevent the snake from crossing the relief boundary. This is important to prevent snake entering the relief where its boundary is weakly defined.

We also add weights to the energy terms. Initially, during the coarse phase, each energy term had unit weight. Now the energy is redefined as

$$E = W_{int}(E_t + E_b) + W_f E_f + W_d E_d. \quad (10)$$

Decreasing  $W_{int}$  reduces the internal energy, allowing the snake to bend more, while having less tendency to shrink. Increasing  $W_f$  helps to force the snake to stop at the relief boundary. Increasing  $W_d$  helps to drive the snake into concavities. Clearly, it is the relative sizes of these weights which are important. In practice we fix  $W_f$  at 1.  $W_{int}$  is chosen by the user, as described later.  $W_d$  is then set to be larger than one, keeping the sum of the weights  $2W_{int} + W_f + W_d = 4$ .

During the refinement step, we set the initial snake segment length appropriate to the resolution of the mesh: half the average triangle edge length. The inserted snaxels are computed by resampling the snake on the mesh between the coarse snaxels.

### 6.3 Final Stabilisation

After the the refined snake has converged near the relief boundary, using the approach described above, we perform a final adjustment step to restabilise the snake, to remove unnecessary noise and yield a smoother boundary. We turn off the deflationary force by setting  $W_d$  to 0, to stop pushing the snake inwards—we have now driven the snake into the concavities, and now it is important not to overrun the boundary.  $W_f$  is again set to 1, to localise the snake at the relief boundary. We set  $W_{int} = 1$  to help make the final snake smooth. The snake is re-adjusted until its position converges.

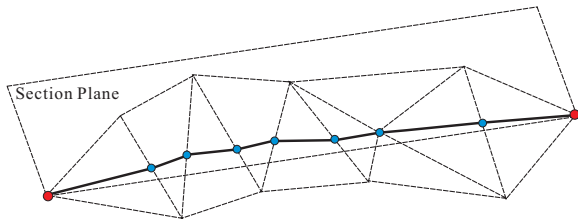


Figure 5: Intersecting the mesh with a section plane

## 7 Implementation Details

We now give various further implementation details of our algorithm.

### 7.1 Initial Contour

The user selects a few (say 4, or 6) mesh points around the outside of the relief, which are then joined across the mesh to give the initial snake. Normally this is a closed curve, but an open curve can also be used, for example if the relief runs to the edge of the mesh. We could use a shortest path algorithm to construct the initial contour [Kanai and Suzuki 2001; Surazhsky et al. 2005]. However, our snake is insensitive to the initial contour, so any cheap and simple method for constructing a reasonable connecting path between each pair of vertices can be used. We use the simple method (illustrated in Figure 5) of creating a section plane through the two vertices and the average of their normals, and intersecting the plane with the mesh. This is simple, fast and effective.

These intersection curves are sampled as described earlier to get the initial snake segments. Though the snake segments initially have equal lengths, there is no need to keep this restriction during evolution as our energy expressions properly take into account the lengths of individual snake segments. However, we do merge or divide any that become too long or too short.

### 7.2 Parameters

Snake-based methods are usually sensitive to choice of parameters. We have tried to make our method depend on few parameters, or at least to operate in a way which is insensitive to precise choice. The main parameter required is a user estimate for the relief step height  $h$  at the boundary of the relief; if this is variable, an estimate for its minimum value should be given.

Other factors affecting the discretised snake are adjusted according to the mesh model, as described next.

#### 7.2.1 Parameters for Planarity Calculation

The number of neighbours of a given point to use when calculating planarity at a point depends on the resolution of the input mesh. We use all neighbours within a geodesic distance equal to the relief step height for this planarity calculation, as this is a length scale appropriate to the problem. The fast-marching method [Kimmel and Sethian 1998] is used to compute the geodesic distance to the neighbours of every vertex. For planarity bilateral filtering, the Gaussian filter  $W_c$  in Eqn. 6 decreases quickly as the distance increases. As further neighbours do not affect the filtering result much, just the 2-ring neighbours are used. Following Fleishman et al. [2003],  $\sigma_c$

is set to half of the maximum distance to any of the two-ring neighbours, and  $\sigma_s$  is set to the variance of the planarity.

#### 7.2.2 Parameters for Snaxel Control

The relevant parameters here are,  $l$ , the initial length of each snake segment, the range of permitted snake segment lengths, and the sharpest permissible angle between segments (used to avoid self-intersections). The length of the snake segments affects speed of movement towards the contour, as well as its ability to precisely locate details of the contour and to explore concavities. As noted earlier, at the start of the coarse evolution phase,  $l$  is set to  $h$ , while at the start of the refinement phase,  $l$  is set to half of the average triangle edge length.

Segment lengths can vary in length as evolution proceeds, and although small changes are unimportant, they may change greatly wherever the snake shrinks or lengthens significantly. Redundant snaxels slow the algorithm down, while snaxels which are too sparse will not capture enough details of the contour. Thus, during both phases, extra snaxels are added, or snaxels are removed, if the snaxel spacing varies outside the range  $[0.5l, 1.5l]$ .

If two neighbouring segments form a sharp angle, self-intersection of the snake may occur after a few more steps. Thus, snaxels are removed if  $\theta_i$  (see Figure 3) is larger than some maximum angle, set to  $160^\circ$  in our program.

#### 7.2.3 Parameters for the Energy Terms

Various parameters are used to control the relative importance of the different energy terms. These are three normalisation parameters  $\alpha$ ,  $\beta$ ,  $\gamma$  (see Eqns. 3, 4 and 7) and three weights  $W_{int}$ ,  $W_f$  and  $W_d$  (see Eqn. 10).

The parameters  $\alpha$ ,  $\beta$  and  $\gamma$  are used to normalise the different energy terms. The snake segment length is initialised to  $l$  and it ranges between  $l_{min} = 0.5l$  and  $l_{max} = 1.5l$ . Thus  $E_t$  lies in the range  $0.5\alpha l_{min} \leq E_t \leq 0.5\alpha l_{max}$ . From this, it is easy to choose  $\alpha$  to normalise  $E_t$  to lie in the range  $[0, 1]$ . Note, however, that different values are used in the coarse evolution and refinement phases.  $\beta$  and  $\gamma$  can be determined in a similar way.

The deflation energy is dynamically defined by snaxel length and speed of movement according to the internal energy at concavities, and is not normalised.

The weights  $W_{int}$ ,  $W_f$  and  $W_d$  are used during refinement and stabilisation to adjust the relative importance of the energy terms as explained earlier. As noted, the only free parameter is  $W_{int}$  which is chosen by the user. Typically, the user only has to try a *very* small number of values, each of which should be half the previous value, before satisfactory results are obtained.

#### 7.2.4 Other Parameters

Our method is relatively insensitive to choice of other parameters, and these are fixed in the algorithm. These include:

- $\tau$ : the movement step size: 0.2;
- $\varepsilon$ : for locating candidate points for feature energy: 0.2;
- length tolerance used for oscillation detection:  $2\tau l$ .

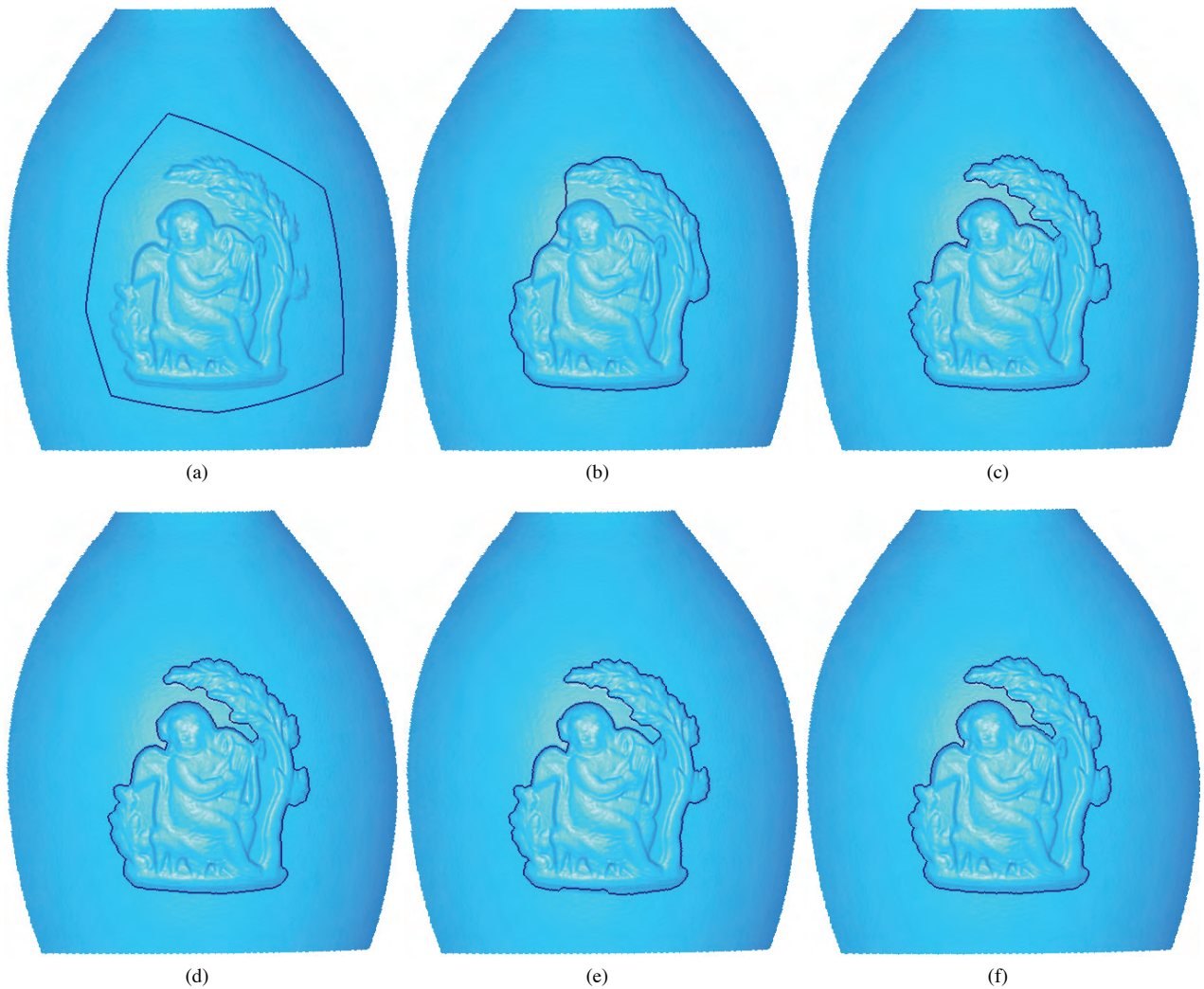


Figure 6: Relief segmentation of a model with a deep concavity: (a) initial contour, (b) coarse result, (c) refined result, (d) final contour ( $h = 1.0$ ), (e) final contour with  $h = 0.6$ , (f) final contour with  $h = 1.4$

The step height of the relief  $h$  need not be estimated very accurately by the user. In our experiments with the duck model (see Figure 8), good results were obtained using estimates in the range from 0.4mm–1.0mm. We provide the user with an interactive tool to calculate the geometric distance between two user-defined vertices on the mesh, allowing the user to easily estimate  $h$ .

Overall, our algorithm simply requires the user to estimate the parameter  $h$  and to tune the parameter  $W_{int}$ . All other parameters are fixed, automatically calculated or adaptively change according to the model and the snake evolution process. It is not hard for a non-technical user to produce good results.

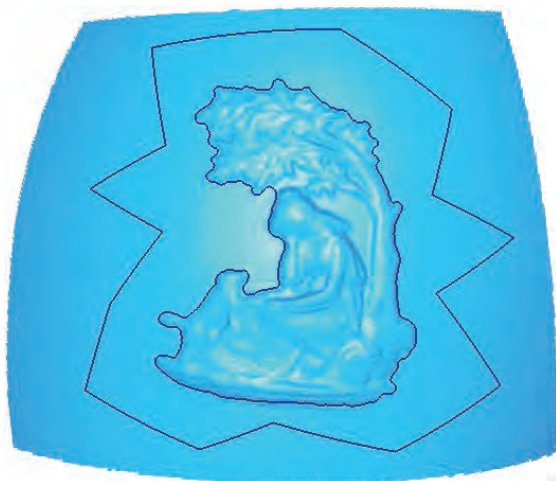
## 8 Experimental Results

A variety of real scanned geometric models have been tested, having different relief contour shapes and with different resolutions and characteristics. All examples were tested on a computer with a 2.4GHz CPU and 1GB RAM.

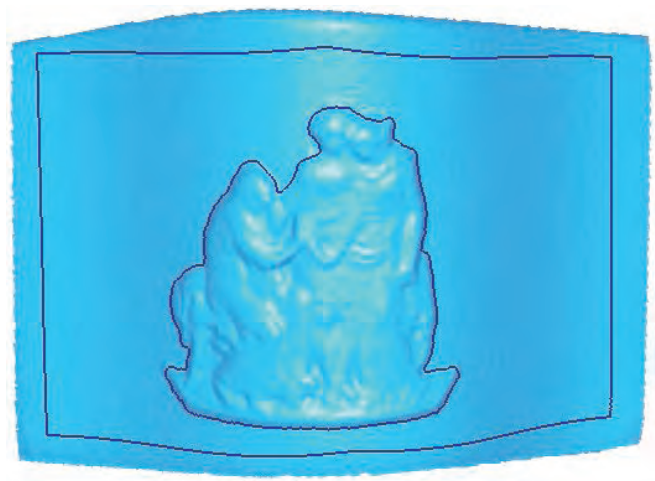
### 8.1 Examples

Figure 6 shows the process of relief segmentation for a relief whose contour has a deep concavity. The model has 287446 triangles, with average edge length 0.21mm. Figure 6(a) shows the initial contour as created from a typical set of 7 user-specified points. Figures 6(b–d) illustrate the snake evolution process. A coarse relief contour with 175 snaxels is obtained after 131 iterations of energy minimisation during the coarse step, taking 21 seconds, as shown in Figure 6(b). Figure 6(c) shows the result after the refinement step; in this case the weight  $W_{int}$  was set to 0.25. It took 60 seconds using 459 iterations to produce the contour which comprises 1136 snaxels. Figure 6(d) is the result after final stabilisation. This process took 10 seconds and 94 iterations. Overall, the total process took 91 seconds. Throughout Figures 6(a–d), the parameter  $h$  was set to 1.0mm. Figures 6(e) and 6(f) show other final results when  $h$  was set to 0.6mm and 1.4mm respectively. Note that the final result does not change greatly.

Relief segmentation results for a variety of reliefs on a variety of backgrounds are shown in Figure 7. Figure 7(a) shows a model where the initial contour has deliberately been drawn as a zigzag



(a)



(b)

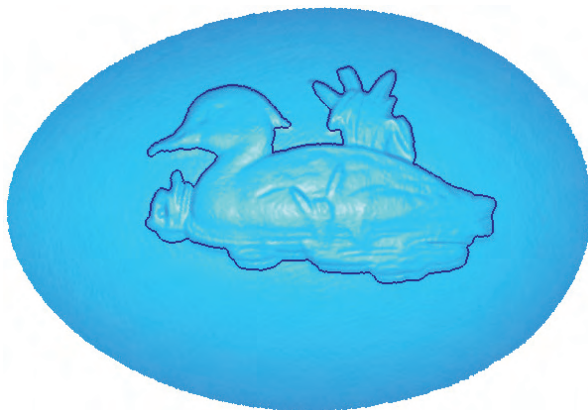


(c)

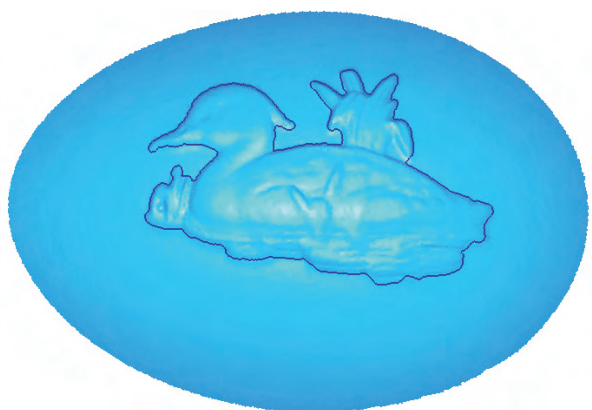


(d)

Figure 7: Relief segmentation for various relief and background shapes: (a) snake evolved from a zigzag initial contour, (b) background of varying curvature, (c) an open snake, (d) horse relief



(a)



(b)

Figure 8: Segmentation results for duck relief with different mesh resolutions: (a) scanned by Minolta VI-910, (b) using decimated mesh

shape to demonstrate that our method is insensitive to choice of initial contour. The snake both explores the concavities very well, and also represents the shape of the convex part in detail. For this model, it took 20 seconds to get the coarse contour using 194 snaxels and another 66 seconds to obtain the final results using 1525 snaxels. Figure 7(b) shows an example where the initial contour is far from the relief, and where the background surface has both positive and negative Gaussian curvature. In Figure 7(c), the relief boundary is an open contour, ending at the mesh boundary; for this example the background is also very noisy. Again we get a good result. Figure 7(d) is another example which shows that the snake goes into deep concavities while not overrunning the convex parts.

Figure 8 shows segmentation results for a duck relief using 2 different mesh resolutions. The mesh used in Figure 8(a) was scanned using a Minolta VI-910, and has 168700 triangles with average length 0.227mm. The initial model for Figure 8(b) was produced by decimation of the mesh used in Figure 8(a), and has 47327 triangles with average edge length 0.555mm.

## 8.2 Discussion

Our testing has shown that the coarse evolution phase readily produces reasonable and stable results however the initial contour is chosen. During the coarse phase, sensitivity to  $h$  and other parameters is low, but it is higher in the later phases. For example, for Figure 6,  $h$  can be successfully set anywhere in the range  $[0.3, 1.6]$  during the coarse phase, but must be in the range  $[0.6, 1.4]$  during the refinement phase.

Our method can deal with most concavities except those having very narrow entrances. Adjusting the method to force the snake into such concavities also generally results in the snake overrunning the relief contour elsewhere. Generally, the snake cannot enter concavities whose widths at their mouths are less than 4 to 6 times the average edge length of the mesh.

## 9 Conclusions and Future Work

In this paper, we have shown how to use snakes for segmenting isolated reliefs lying on a smooth and slowly varying background surface. After user delimitation of a few background points surrounding the relief, the algorithm automatically creates an initial contour and actively evolves it to the relief boundary through a coarse phase, a refinement phase and a stabilisation phase. The process is controlled by a user estimate of relief step height, and one other parameter.

For future work, we intend to deal with reliefs with small internal holes, as well as coping with concavities with narrow mouths, by first estimating a continuation of the background surface. We also intend to extend our algorithm to cope with more complicated reliefs such as those having textured backgrounds, and cyclic reliefs. For textured background, we hope to drive the snake using an energy based on classification of the mesh into texture and relief. Closed cyclic reliefs, necessitating analysis of repeating items.

## 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.

## References

- BISCHOFF, S., AND KOBBELT, L. 2004. Parameterization-free active contour models with topology control. *The Visual Computer* 20, 4, 217–228.
- BISCHOFF, S., WEYAND, T., AND KOBBELT, L. 2005. Snakes on triangle meshes. <http://www-i8.informatik.rwth-aachen.de/publications/publications.html>.
- COHEN, L. D., AND COHEN, I. 1993. Finite element methods for active contour models and balloons for 2d and 3d images. *IEEE Trans. Pattern Analysis and Machine Intelligence* 15, 11, 1131–1147.
- COHEN, L. D. 1991. On active contour models and balloons. *CVGIP: Image Understanding* 53, 2, 211–218.
- FLEISHMAN, S., DRORI, I., AND COHEN-OR, D. 2003. Bilateral mesh denoising. *ACM Transactions on Graphics* 22, 3, 950–953.
- FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. *ACM Transactions on Graphics* 23, 3, 652–663.
- JUNG, M., AND KIM, H. 2004. Snaking across 3d meshes. *Proceedings of the Computer Graphics and Applications* 12, 87–93.
- KANAI, T., AND SUZUKI, H. 2001. Approximate shortest path on a polyhedral surface and its applications. *Computer-Aided Design* 33, 11, 801–811.
- KASS, M., WITKIN, A., AND TERZOPOULOS, D. 1988. Snakes: Active contour models. *International Journal of Computer Vision* 1, 4, 321–331.
- KIMMEL, R., AND SETHIAN, J. A. 1998. Computing geodesic paths on manifolds. *Proceedings of National Academy of Sciences* 1995 15, 8431–8435.
- LEE, Y., AND LEE, S. 2002. Geometric snakes for triangular meshes. *Computer Graphics Forum* 21, 3, 229–238.
- MILROY, M. J., BRADLEY, C., AND VICKERS, G. W. 1997. Segmentation of a wrap-around model using an active contour. *Computer-Aided Design* 29, 4, 299–320.
- SURAZHISKY, V., SURAZHISKY, T., KIRSANOVAND, D., GORTLER, S., AND HOPPE, H. 2005. Fast exact and approximate geodesics on meshes. *Proceedings of ACM SIGGRAPH 2005* 24, 553–560.
- TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. *Proceedings of the 1998 IEEE International Conference on Computer Vision*, 839–846.
- WILLIAMS, D. J., AND SHAH, M. 1992. A fast algorithm for active contours and curvature estimation. *CVGIP: Image Understanding* 55, 1, 14–26.
- XU, C., AND PRINCE, J. L. 1998. Snakes, shapes, and gradient vector flow. *IEEE Trans. Image Processing* 7, 3, 359–369.