Curve Analysis with Applications to Archaeology

Research Thesis

In Partial Fulfillment of the

Requirements for the

Degree of Doctor of Philosophy

Michael Kolomenkin

Submitted to the Senate of

the Technion – Israel Institute of the Technology

Tamuz 5773 Haifa June 2013

The Research was done under the supervision of Prof. Ilan Shimshoni from the department of Information Systems of Haifa University and Prof. Ayellet Tal from the Faculty of Electrical Engineering of the Technion

Acknowledgement

I thank my parents, Tamara and Victor Kolomenkin, for their help and support

I thank Prof. Ilan Shimshoni and Prof. Ayellet Tal for their guidance during all the stages of the research

The generous financial help of the Technion and the Ollendorff fund is gratefully acknowledged

Contents

List of Figures

List of Tables

Abstract

In this thesis we discuss methods for the definition, detection, analysis, and application of curves on surfaces. While doubtlessly as important as curves in images, curves on surfaces gained less attention. A number of definitions of curves on surfaces has been proposed. The most famous among them are ridges and valleys. While portraying important object properties, ridges and valleys fail to capture the shape of some objects, for example of objects with reliefs. We propose a new type of curves, termed relief edges, which addresses the limitations of the ridges and the valleys, and demonstrate how to compute it effectively. We demonstrate that relief edges portray the shape of some objects more accurately than other curves. Moreover, we present a novel framework for automatic estimation of the optimal scale for curve detection on surfaces. This framework enables correct estimation of curves on surfaces of objects consisting of features of multiple scales. It is generic and can be applied to any type of curve. We define a novel vector field on surfaces, termed the prominent field, which is a smooth direction field perpendicular to the object's features. The prominent field is useful for surface enhancement and visualization. In addition, we address the problem of reconstruction of a relief object from a line drawing. Our method is able to automatically reconstruct reliefs from complex drawings composed of hundreds of lines. Finally, we successfully apply our algorithms to archaeological objects. These objects provide a significant challenge from an algorithmic point of view, since after several thousand years underground they are seldom as smooth and nice as manually modelled objects.

Chapter 1

Introduction

Since the early days of humanity, curves have been employed to convey information about the shape. From cave drawings and tattoos, through hieroglyphs and maps, to engineering drawings and arts, curves have accompanied humans and helped them to communicate. Today curves continue to serve as an important tool for portraying scenes and visualizing ideas.

Curves posses several properties that make them so attractive. First, curve drawings are easy and cheap to produce. They do not require expensive materials and long preparations. Second, the meaning of curve drawings is often intuitive and can be grasped regardless of cultural background. Third, curves are very informative – a small number of curves is sufficient to depict a complex structure.

The sparsity of curves makes them attractive not only for humans, but also for computer algorithms. Sparsity reduces storage and may result in faster algorithms which are able to run on cheaper hardware. This fact is emphasized by the large number of curve-based applications in the field of computer graphics and computer vision – segmentation [118, 124], tracking [10, 27], retrieval [54], non-photorealistic graphics [123], and navigation [99], to name a few. The above applications typically employ curves to describe features, which are the prominent structures in 2D images or on surfaces.

Curves in images have been a lively topic of research for several decades. There exists a range of different methods for their detection and usage. Although curves on objects are doubtlessly as important as curves in images, they have gained less attention. However, several definitions of curves on surfaces have been proposed, each suitable for a different problem. The most common family of curves on surfaces is ridges and valleys [98], which are the extrema of principal curvatures. Ridges and valleys indicate sharp creases on surfaces. Other types of curves are parabolic curves, which partition the surface into hyperbolic and elliptic regions, and zero-mean curvature curves, which classify sub-surfaces into concave and convex shapes [67]. They correspond to the zeros of the Gaussian and the mean curvature, respectively. While portraying important object properties, ridges and valleys are not suitable for surfaces without creases and for surfaces with reliefs.

There are also attempts to define view dependent curves [24, 56]. These curves depend not only on the differential geometric properties of the surface, but also on the viewing direction. These curves are often aesthetically pleasing and thus are applicable for non-photorealistic rendering in computer graphics. However, these curves may be inappropriate in applications that require a stable representation of a feature from all viewing directions, as needed, for example, in shape analysis in archaeology and medicine.

In this thesis we propose a new type of curves, termed *relief edges*, which addresses the limitations of the commonly used ridges and valleys. Intuitively, a surface can be considered as an unknown smooth manifold, on top of which a local height image is placed. Relief edges are the edges of this local image. We show how to compute these edges from the local differential geometric properties of the surface.

We show how to apply our curves to capture 3D shape information visually. The curves can be used straightforwardly, by simply drawing them on top of the 3D objects. However, better representation is obtained when our curves are combined with curve-based coloring. In this scheme, a color of a point on the object is proportional to the curvature in the direction perpendicular to the nearest curve feature. This coloring increases the contrast on the feature curves, thus enhancing them.

Moreover, we propose a general framework for automatically estimating the optimal scale at each point on the surface. This general scheme can then be applied to every type of 3D curve, assuming it's strength can be defined by the curvature and its derivatives. Our framework eliminates the need for user intervention and enables to detect features of different scales on a single object.

In addition, we suggest a novel framework for enhancing objects, based on a definition of a new direction field (a normalized vector field), termed the *prominent field*. Intuitively, the direction of this prominent field, termed the *prominent direction*, is perpendicular to the surfaces feature curves and is smooth. The object is enhanced by smoothing it in the direction of the features (of the prominent field) and keeping it intact in the perpendicular direction.

Finally, after drawing curves from surfaces, we turn to the inverse problem of constructing surfaces from line drawings. Understanding the 3D shape of an object from its line drawing is a basic human ability. Even young children can easily recognize and reconstruct the shape in their minds from a handful of lines [131]. However, automatic reconstruction from a line drawing is very challenging [21]. First, the lines are usually sparse and thus, the object is not fully constrained by the input. Second, the line drawings are often ambiguous, since the lines may have different geometric meanings they can indicate 3D discontinuities, surface creases, or 3D step edges. Third, the input may consist of a large number of strokes that need to be specified by the user and handled by the algorithm efficiently. Fourth, these strokes are inter-related.

We propose an algorithm that is able to reconstruct a relief from a complex drawing that consists of many inter-related strokes. The algorithm is based on two key ideas. First, the inter-dependencies between the strokes of the line drawing can be exploited to automatically generate a good initial interpretation of the line drawing. Second, given an interpretation, it is possible to reconstruct a consistent surface.

As an application in this thesis, we specifically focus on the domain of archaeology. This domain is important, because analysis of archaeological artifacts, such as seals, ceramic vessels, coins, etc. is a major source of our knowledge about the past. From our point of view, the domain is highly challenging, since after spending several thousand year underground the artifacts are often broken and eroded.

Hence, our thesis makes several contributions. First, we propose a new type of curves, termed relief edges (Chapter 4). Relief edges portray step-like features more accurately than other curves. Second, we present a general framework for automatically detecting curves at the optimal scale (Chapter 5). This general scheme can then be applied to every curve type. Third, we demonstrate how to process objects with prominent field (Chapter 6). The field helps us to enhance, denoise, and visualize objects. Fourth, we present an algorithm for reconstruction of reliefs from complex line drawings (Chapter 7). The algorithm can reconstruct a model semi-automatically from a single drawing. Last, but not least, we apply our algorithm to highly challenging archaeological objects.

1.1 Related work

In this section we survey the various types of curves on surfaces and the algorithms for their detection. Before we proceed to discuss them, let us briefly discuss edges in images. Initial approaches treat the edge detection locally. They provide mathematical definitions of curves or propose methods to compute them efficiently. Later approaches are more global in nature. Some of them show how to calculate curves at a point more accurately and robustly by using information from the point neighbourhood. Others distinguish between real and spurious curves by combining smaller curves into larger structures and checking whether they agree with each other. For additional information about curves in images see the surveys [6, 115, 150]. In images, the focus has been on a single curve type - the intensity edge. This is so since edges are considered to be the most important and widely used low-level curves.

On surfaces, there is no single curve that is suitable for all applications [20]. Every curve has its strengths and weaknesses. Below, we describe various curve types and present the algorithms for their detection.

Zero lines of curvature-based functions: The first curves to be defined on a surface are the zero lines of curvature-based scalar functions [67]. Specifically, the *zero mean curvature* and the *parabolic (Gaussian)* curves were used. The Mean curvature H is the average of the principal curvatures κ_1 and κ_2, and the Gaussian curvature G is their product:

$$H = \frac{1}{2}(\kappa_1 + \kappa_2),$$

$$G = \kappa_1 * \kappa_2.$$

The Gaussian curvature separates the surface into flat ($G = 0$), convex ($\kappa_1 > 0, \kappa_2 > 0$), concave ($\kappa_1 < 0, \kappa_2 < 0$), and Hyperbolic ($\kappa_1 \kappa_2 < 0$) areas. While mathematically elegant, parabolic curves are hardly used in practice [24]. This is so since they are noisy and seldom correspond to visually meaningful data.

Zero mean curvature curves are sometimes used for feature detection [101]. These lines can be thought of as an extension of the 2D Laplacian edge detector to surfaces. When the surface is represented as a function defined on a plane (as in 2D images), they are equal to the Laplacian edges [133, 134].

The main drawback of the zero mean curvature curves is their sensitivity to the underlying surface. When a feature resides on a surface, we expect the feature be portrayed by the same curves whatever the underlying surface looks like. However, the mean curvature changes with the change of the curvature of the underlying surface.

Ridges and valleys: Undoubtedly, the most popular surface curves are ridges and valleys. which are also known as “surface creases” and “normal discontinuities”. Intuitively, ridges and valleys are identical to their geographical counterparts. They reside along sharp discontinuities of surface orientation. Ridges correspond to convex discontinuities and valleys correspond to concave ones.

Formally, ridges (valleys) are defined as the loci of points where the curvature obtains maximum (minimum) in the first principal direction. They are not defined at umbilic points, i.e. points where the principal directions are not defined. Let $\mathbf{p}$ be a point on a surface, $\kappa_{\max}$ and $\kappa_{\min}$ be the principal curvatures at $\mathbf{p}$, and $t_{\min}$ and $t_{\max}$ be the corresponding principle directions. Then, $\mathbf{p}$ is a ridge point if $\kappa_{\max}$ attains a local positive maximum at $\mathbf{p}$ along the associated integral curve of $t_{\max}$. And $\mathbf{p}$ is a valley point if $\kappa_{\min}$ attains a local negative minimum at $\mathbf{p}$ along the associated integral curve of $t_{\min}$.

The definitions of the ridges and valleys are dual: if we change the surface orientation then the ridges turn into the valleys and vice versa. Therefore, without loss of generality we will consider only the ridges.

Many methods have been proposed for ridge computation on different surfaces. For example, [9, 98] show how to calculate them on implicit surfaces, [15, 147, 46] – on triangular meshes, [58, 26, 100] – on point clouds, [8] – on range data, and [92] – on B-spline surfaces. In addition to general calculation, the latter paper also shows how to compute ridges in the neighbourhood of umbilic points.

These methods are concerned with the basic, point-wise computation of ridges. They concentrate on finding accurate and efficient methods for the computation of the curvature and the curvature extrema. While they are a vital step in ridge detection, by themselves they are insufficient for capturing large features. They are influenced by outliers and noise.

Another class of algorithms focuses on addressing the problems mentioned above. They attempted to make ridges smoother and more robust to noise by considering sets of points and examining the data in the points' neighbourhoods. For example, [63] employ a voting technique to identify salient structures, [109] use morphological operators to detect and remove holes, [93] propose to track ridges to fill discontinuities in the data, [46] add a regularization component to the definition, and [136] combine segmentation and edge detection in a region-growing algorithm.

A third class of algorithms is inspired by the formal definition of ridges, but does not realize it accurately. These algorithm define curves as the loci of point of high surface variation. For instance, [102, 49] compute the ratio of surface eigenvectors and use minimum spanning tree algorithms to build features from all the points whose ratio is above the predefined threshold. [96] first warp the surface normals onto a 2D map and then search for points with high 2D derivative. [11] generalizes the definition of features lines as regions that correspond to arbitrary cross sections extruded along straight or circular lines. In practice it uses the definition to propagate lines from an initial location of high curvature.

The considerable research efforts devoted to extraction of ridges and valleys resulted in accurate and effective computation. However, the creases themselves are sometimes insufficient for depicting shape [20]. For example, they may be less suitable for smooth features, because of the very definition of ridges & valleys as “creases” which indicate discontinuity.

View dependent curves: Another family of curves on surfaces concerns view-dependent curves. As their name implies, these curves depend not only on the surface, but also on the viewing direction. The curves can be classified according to the order of the surface properties that they use. Silhouettes, or contours, are first order curves [67]. They are defined as the points where the normal is perpendicular to the viewing direction. Suggestive contours and suggestive highlights are second order curves [24, 25]. They are the zero lines of the view dependent curvature. Apparent ridges are third order curves [56]. They are the ridges of the view-dependent curvature, i.e. the points where view dependent curvature obtains maximum.

Recently proposed view-dependent curves also incorporate the light direction. Photic Extremum Lines (PELs) extend the famous Canny image edge detector by defining curves as the loci of points for which the variation in illumination reaches maximum [143]. Laplacian lines are the extension of the Lapalcian edge detector [149]. They are points for which the Laplacian of the illumination is zero.

View-dependent curves are often more aesthetically pleasing and therefore appropriate for non photorealistic applications in computer graphics. They correspond nicely to artistic drawings. However, they are less suitable for shape analysis applications that require accurate detection of surface features.

Chapter 2

Research Methods

Our research methods are driven from two characteristics of our data. The first is that models are the archaeological artifacts. Thus, we expect the input to be noisy, eroded, and often incomplete. Moreover, the output has to fit the requirements of our end user – the archaeologist. The second characteristic is that our objects are reliefs and can therefore be viewed as a generalization of images. Reliefs can be represented as functions defined on a general smooth surface, while images are functions defined on a plane.

Below we describe the archaeological data that we use and show how it influences our work. Then, we explain the common principles and ideas behind the algorithms for surfaces with reliefs.

2.1 Archaeological data

From the very start, our goal was to investigate techniques applicable to real archaeological problems. This implies on certain requirements from the input and the output of our algorithm. We took care to use only objects provided by archaeologists. Simpler, synthetic objects were valuable only for preliminary testing of ideas. In addition, we aimed to create useful output and cooperated with archaeologists to validate the results.

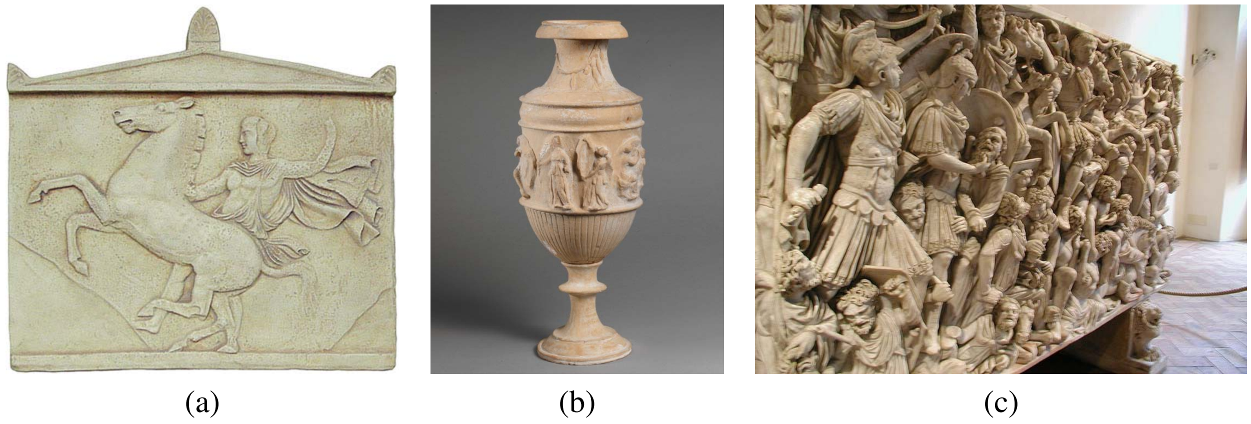


Figure 2.1: **Examples of reliefs.** (a) part of Rome Pantheon low relief, (b) Greek terracota vase with medium relief, (c) Roman sarcophagus with high relief.

Among the various types of archaeological artifacts, we concentrated on objects with reliefs. These objects consist of a smooth base surface (such as lamps, vases, seals, etc.) with protruding or immersed details. The details may represent anything from simple geometric ornaments to real 3D objects drawn on the surface. The degree of object depth can vary from shallow reliefs, where the base surface is only slightly scratched and the depth of the 3D object is distorted, to high-reliefs, where the depth is apparent, as shown in Figure 2.1.

We decided to focus on archaeology for several reasons. First, the archaeological artifacts provide very interesting and unique challenges. This is so, since after several thousands years underground, they are seldom as complete and smooth as man-made objects are. Second, archaeology offers us a clear criterion of success, since the usefulness of our results could be quickly verified with the archaeologists. Third, our work can significantly boost up the archaeological research. Recording and retrieving information about artifacts is done manually, and is therefore a slow and expensive process which has not changed much for the last century. It is still one of the main bottlenecks of archaeological research. Introducing state-of-the-art computer vision and computer graphics algorithms can considerably speed up and improve their work. Our focus on reliefs is especially important since reliefs represent the footprints of a specific period or an area. The shapes of the base surfaces usually remain constant over time, while the reliefs were subject to more frequent changes. Finally, solving our problems on archaeological data yields principles that apply to other types of data.

We had to take a special step to adapt our algorithms to the archaeological data. We employed a very accurate 3D scanner that ensured high quality scans of even the smallest objects. We used a Polygon scanner [1] with raw measurement accuracy of $0.01mm$ and surface restoration accuracy of $0.1mm$.

The output of the algorithm is also influenced by the requirements of our human user. We collaborated with archaeologists to better understand these requirements and developed algorithms that accommodated the output to fit them. For example, the output of the feature detection methods is a set of curves that capture the shape. While the curves portray the features accurately and may be preferred for subsequent automatic processing, they are not the optimal representation for a human observer. Instead, we developed a coloring algorithm that colors the surface in grey level tones in according with the feature position. The visualization turned out to be much easier to quickly grasp the shape of the surface then representing it using curves.

2.2 Handling relief objects

Our methods for processing relief objects are based on two observations. The first observation is that reliefs can be viewed as a generalization of images. Therefore, some of our algorithms are inspired by the corresponding algorithms for images. The second observation is that the information in reliefs is sparse. The surface of the relief is piecewise smooth and our algorithms focus on the curves that indicate transitions between the smooth areas.

Below, we first discuss the relief representation common for all our methods. Then, we describe the above observations and their influence on our work in more detail.

2.2.1 Relief representation

We represent a relief as a function defined on the underlying surface. We assume that the base surface is locally a manifold and that its curvature has a smaller value than the curvature of the function. We term the function *the local image* and the underlying surface *the base*.

In many cases, we consider only the local image and not the base. This is so since the base is often the same for different objects and the interesting information is only in the image. Thus, processing a general surface is reduced to processing an image defined on an unknown manifold and our tasks become generalizations of the standard image processing tasks. Actually, in the special case of the underlying surface being a plane, the local image is reduced to the common 2D image.

We do not assume that either the image or the base is known and therefore we do not make any assumption regarding the structure of the base.

2.2.2 Relief-oriented techniques

In this section we describe the properties of reliefs that influence our algorithms in Chapters 3-7.

Relief as image generalization: While being similar to images, reliefs posses two important differences. These differences explain why applying algorithms that fail on natural images succeed on reliefs and vice versa. The first difference is that the base-image decomposition in reliefs is unknown. Thus, even the most basic approaches for images processing, which treat the image as a function (like gradient computation), cannot be applied to reliefs in a straightforward manner. Instead, we look for the image characteristics that are "base-invariant", i.e. can be computed independently of the underlying base.

The second difference is that natural images may include high frequency structures, such as textures, edges resulting from 3D occlusions or illumination effects. Conversely, reliefs physically cannot have high frequencies.

The absence of high-frequencies lets us to successfully utilize higher order properties of the surface, such as its second and third derivative. Similar approaches in natural images usually did not gain popularity because the high frequency components in the image made them unstable and prone to noise [43, 64, 106].

Sparsity Reliefs are smooth everywhere except along *edges* – the sharp changes of the intensity of the local image. Therefore, the information representing the shape of the relief is sparse. As the important information resides only along edges, it is natural for our algorithms to focus on them. The edges are important both for the human observers and for various processing tasks.

In particular, feature detection algorithms search for edges (Chapters 3,4, and 5). They search for locations on the surface whose geometry resembles that of the ideal edge in all the possible scales and orientations. The edges detected by these algorithms are employed by the shape processing algorithm in order to determine the strength and the direction of the surface filtering (Chapter 6). The object is smoothed in the direction of the edges and enhanced in the perpendicular direction. The surface reconstruction algorithm estimates the size and the location of the edges from a line drawing and then reconstructs the rest of the surface (Chapter 7). The reconstruction is performed by an integration algorithm that keeps the whole surface smooth while keeping the edges intact. The edges serve as the boundary conditions for the integration algorithm.

Chapter 3

Demarcating curves for shape illustration

Abstract

Curves on objects can convey the inherent features of the shape. This paper defines a new class of view-independent curves, denoted *demarcating curves*. In a nutshell, demarcating curves are the loci of the “strongest” inflections on the surface. Due to their appealing capabilities to extract and emphasize 3D textures, they are applied to artifact illustration in archaeology, where they can serve as a worthy alternative to the expensive, time-consuming, and biased manual depiction currently used.

This paper appeared in SIGGRAPH ASIA (ACM TOG) 2008.

3.1 Introduction

Curves drawn on objects convey prominent and meaningful information about the shape. They can therefore be utilized in a large spectrum of applications, including non-photorealistic rendering [123], segmentation [124], robot navigation [99], simplification [102], brain analysis [4], registration of anatomical structures [105], and the recovery of archaeological and architectural information [84]. Recent user studies [19] do not conclusively choose one of the current types of curves as the best for all cases. Therefore, the search for additional curves continues. Moreover, this search could be guided by specific application areas, where certain types of curves are preferred.

Feature curves can be classified as *view-dependent* or *view-independent* curves. View-dependent curves depend not only on the differential geometric properties of the surface, but also on the viewing direction. They change whenever the camera changes its position or orientation [67, 24, 25, 56]. View-independent curves do not change with respect to the viewing direction [52, 57, 98, 102, 145]. One criticism of view-independent curves is that they can appear as markings on the surface [24]. Even so, we believe there is merit to using such curves, in particular for applications such as archeology, architecture and medicine. We support this idea with a small study on artifact illustration in archeology.

This paper defines a new class of view-independent curves, termed *demarcating curves*. They are the loci of points for which there is a zero crossing of the curvature in the curvature gradient direction. Demarcating curves can be viewed as the curves that typically separate valleys and ridges on 3D objects (hence the name *demarcating*).

Our results demonstrate that demarcating curves effectively manage to capture 3D shape information visually. For instance, Figure 3.1 demonstrates its ability to depict the 3D texture of an object, such as the facial features and the hair, when comparing it to other well-known curves. They are as quick to compute as ridges and valleys and suggestive contours. Moreover, they can be combined with a shading model to jointly convey the details of the shape.

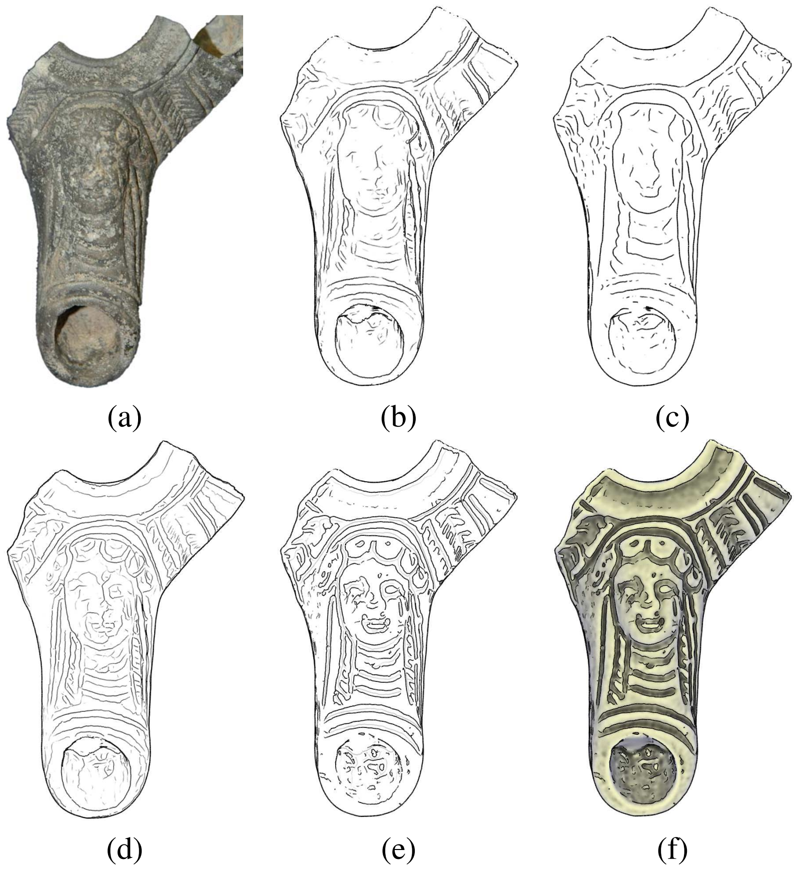


Figure 3.1: **A late Hellenistic lamp (150-50 BCE) rendered with different feature curves.** (a) Original object, (b) Apparent ridges, (c) Suggestive contours, (d) Valleys & ridges, (e) Demarcating curves with valleys, (f) Demarcating curves with shading.

Archaeology has attracted a lot of attention of researchers in computer graphics and visualization [110, 69, 12]. This paper focuses on one aspect of archaeological research – relic illustration. Traditionally, archaeological artifacts are drawn by hand and printed in the reports of archaeological excavations – an extremely expensive and time-consuming procedure (e.g., Figure 3.2, [122]). The main purpose of these drawings is to depict the features of the 3D object so that the archaeologist can visualize and compare artifacts without actually holding them in her hand. Such drawings are often inaccurate, since the precision of the drawn curves depends on the qualifications of the artist. In addition, this technique does not always suffice due to space limitations that force the archaeologist to choose which objects will be drawn and decide on a small fixed set of viewing directions. Digitizing the

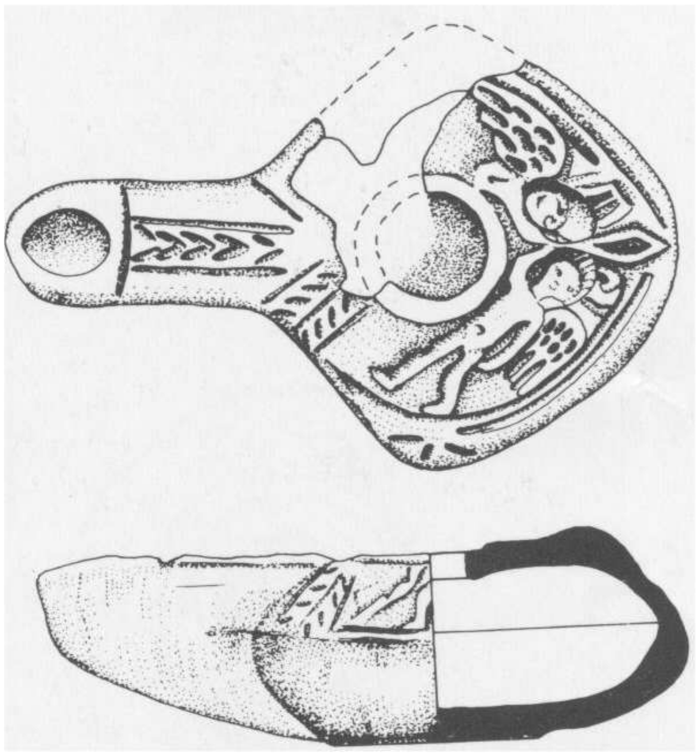


Figure 3.2: **Lamp drawing in archaeology [Stern 1995]**

findings by a high resolution scanner and drawing the curves directly on the scanned objects is a welcome alternative. This enables the archaeologist to study the artifact from all directions, with the 3D features highlighted.

The contribution of this paper is threefold. First, the paper presents demarcating curves, a new class of non-photorealistic view-independent curves on meshes. Second, some relationships of these curves to other well-known families of curves are discussed. Last but not least, these curves are applied to a real application – artifact illustration in archaeology. A preliminary user study indicates that archaeologists prefer for this purpose using demarcating curves to other types of curves or to manual drawing.

The paper is structured as follows. Section 3.2 reviews related work. Section 3.3 defines demarcating curves and describes the algorithm for computing them. Section 3.4 discusses relations of demarcating curves to other curves. Section 3.5 presents some results. Section 3.6 discusses the use of the curves for artifact illustration in archaeology. Section 3.7

concludes the paper.

3.2 Related work

The approaches for drawing curves characterizing objects in 3D can be categorized according to whether they depend on the viewpoint. A variety of view-dependent curves has been proposed. *Contours (silhouettes)*, which represent the "object outline," are the loci of points at which the object normal is perpendicular to the viewing direction [67, 40, 45]. *Suggestive contours* are the loci of points at which occluding contours appear with minimal change in viewpoint [24, 23]. They correspond to true contours at nearby viewpoints. *Highlight lines* extend the suggestive contours [25]. They roughly correspond to ridges of intensity in diffuse-shaded images. *Apparent ridges* are defined as the ridges of view dependent curvature [56]. *Photic extremum lines* are the set of points where the variation of illumination in the direction of its gradient reaches a local maximum [143].

Other view-dependant approaches utilize image edge detection algorithms by drawing the curves on the projections of the objects to the image [78, 104, 53, 113]. These approaches assist in correct scale selection and may reduce the computational complexity. However, pixel-based representation of image edges might yield low precision. View-dependent curves look visually pleasing and hence suit non-photorealistic rendering applications.

There are a number of view-independent curves. The most common curves are *ridges and valleys* [52, 57, 98, 102, 145], which occur at points of extremal principal curvature. Ridges and valleys portray important object properties. However, drawing only valleys (or ridges) is often insufficient, since they do not always convey the structure of the object. Drawing both will overload the image with too many lines. Moreover, coloring these lines so as to differentiate between them might be cumbersome [52]. Other view-independent curves are *parabolic lines*, which partition the surface into hyperbolic and elliptic regions, and *zero-mean curvature curves*, which classify sub-surfaces into concave and convex shapes [68].

3.3 Demarcating curves

Given a surface in 3D, we can imagine it locally as a terrain with ridges and valleys. Intuitively, *demarcating curves* run on the slopes between the ridges and the valleys. Figure 3.3 shows an example of such a local terrain, where the magenta cross section transverses from concave (valley) to convex (ridge) and the demarcating curve point (green) is the transition point. In other words, demarcating curves are the loci of the "strongest" *inflections* on the surface (i.e., where the transition from convex to concave is the fastest). The challenge is to find them. Below, we define this notion formally.

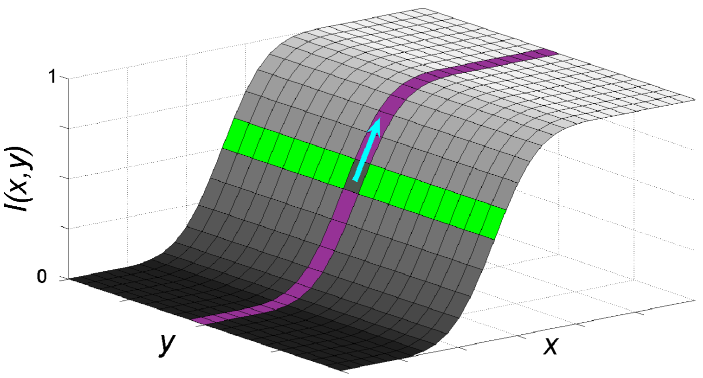


Figure 3.3: **Local terrain (smoothed step edge)**; the demarcating curve in green; the cross section orthogonal to it in magenta; its local direction $\mathbf{g_p}$ in cyan.

3.3.1 Defining demarcating curves

Before defining the curves, we review the definitions of the normal section, normal curvature, the second fundamental form, and the derivatives of curvature [30]. The motivation for using these quantities is that they are intrinsic properties of the surface and are therefore invariant to rigid transformations.

The *normal section* of a regular surface at point $\mathbf{p}$ in tangent direction $\mathbf{v}$ is the intersection of the surface with the plane defined by the normal to the surface at $\mathbf{p}$ and $\mathbf{v}$.

The *normal curvature* at point $\mathbf{p}$ in direction $\mathbf{v}$ is the curvature of the normal section at $\mathbf{p}$, where the curvature of a curve is the reciprocal of the radius of the circle that best approximates the curve at $\mathbf{p}$.

For a smooth surface, the normal curvature in direction $\mathbf{v}$ is $\kappa(\mathbf{v}) = \mathbf{v}^T \mathbf{II} \mathbf{v}$, where the symmetric matrix $\mathbf{II}$ is the *second fundamental form* (which is a special case of the Weingarten matrix, where the first fundamental form is the identity matrix).

The *derivatives of the curvature* are defined by a $2 \times 2 \times 2$ tensor with four unique numbers [111]:

$$\mathbf{C} = (\partial_{u_1}\mathbf{II}; \partial_{u_2}\mathbf{II}) = \left[\begin{pmatrix} a & b \\ b & c \end{pmatrix}; \begin{pmatrix} b & c \\ c & d \end{pmatrix} \right], \tag{3.1}$$

where $\mathbf{u_1}$ and $\mathbf{u_2}$ are the principal directions. Multiplying $\mathbf{C}$ from its three sides by a direction vector $\mathbf{v}$, $\mathbf{C}_{ijk}\mathbf{v}^i\mathbf{v}^j\mathbf{v}^k$ gives a scalar, which is the derivative in the direction $\mathbf{v}$ of the curvature in this direction.

As noted above, we are seeking the loci of the "strongest" inflections, i.e., loci where the curvature derivative is maximal. We therefore define the following.

Definition 3.3.1. *The* curvature gradient *is the tangent direction of the maximum normal curvature variation. Hence, this direction maximizes the following expression:*

$$\mathbf{g_p} = \arg\max_{\mathbf{v}} \mathbf{C}_{ijk}\mathbf{v}^i\mathbf{v}^j\mathbf{v}^k, \;\; s.t \;\; \|\mathbf{v}\| = 1. \tag{3.2}$$

Having defined the curvature gradient direction, we can now proceed to define a demarcating curve point, which is the zero crossing of the normal curvature in the curvature gradient direction.

Definition 3.3.2. $\mathbf{p}$ *is as a demarcating curve point if the following holds at* $\mathbf{p}$*:* $\kappa(\mathbf{g_p}) = \mathbf{g_p}^T \mathbf{II} \mathbf{g_p} = 0$.

3.3.2 Computing demarcating curves on meshes

First, for each vertex, the gradient direction is computed, in accordance with Definition 3.3.1, as well as the value of the curvature in the $\mathbf{g_p}$ direction $\kappa(\mathbf{g_p}) = \mathbf{g_p}^T \mathbf{II} \mathbf{g_p}$. Then, the zero crossings of $\kappa(\mathbf{g_p})$ on the mesh faces are computed according to Definition 3.3.2, to create the demarcating curves. We elaborate on these stages below.

Calculation of $\mathbf{g_p}$: To calculate $\mathbf{g_p}$, the second fundamental form $\mathbf{II}$ and the curvature derivative tensor Equation 3.1 are first found for every vertex [111].[1] Then, $\mathbf{g_p}$ can be either computed analytically or estimated numerically (by sampling). Below, we provide the analytic derivation. A slightly different derivation appears in [88].

To compute $\mathbf{g_p}$ the expression $\mathbf{C}_{ijk}\mathbf{v}^i\mathbf{v}^j\mathbf{v}^k$ is differentiated with respect to $\mathbf{v}$ and compared to zero, as follows. Let $\mathbf{v} = [\cos(\theta), \sin(\theta)]$ be the vector of a unit length, and let a, b, c, d be the coefficients of the curvature derivative tensor (Equation 3.1). Then, Equation 3.2 can be written as:

$$\begin{aligned}\theta_{\mathbf{g}_p} = \arg\max_{\theta}(a\cos^3(\theta) + 3b\cos^2(\theta)\sin(\theta) + \\ +3c\cos(\theta)\sin^2(\theta) + d\sin^3(\theta)).\end{aligned} \tag{3.3}$$

Equation 3.3 is differentiated with respect to θ and compared to zero. After applying some simple algebraic manipulations, we obtain:

$$\begin{aligned}3b\cos^3(\theta) + 3(2c-a)\cos^2(\theta)\sin(\theta) + \\ +3(d-2b)\cos(\theta)\sin^2(\theta) - 3c\sin^3(\theta) = 0.\end{aligned} \tag{3.4}$$

Next, the sin term is isolated and the high order cos terms are substituted by $\cos^2(\theta) =$

[1]implemented using the trimesh2 library by S. Rusinkiewicz

$1-\sin^2(\theta)$ to obtain:

$$\cos(\theta)=\sin(\theta)\frac{(a-3c)\sin^2(\theta)+2c-a}{(3b-d)\sin^2(\theta)-b}. \tag{3.5}$$

After squaring Equation 3.5 and eliminating $\cos^2(\theta)$, the resulting equation depends only on $\sin(\theta)$:

$$\begin{aligned}[(-3c+a)^2+(3b-d)^2]\sin^6(\theta)+\\+[2(2c-a)(-3c+a)-(3b-d)^2-2b(3b-d)]\sin^4(\theta)+\\+[(2c-a)^2+2b(3b-d)+b^2]\sin^2(\theta)+-b^2=0.\end{aligned} \tag{3.6}$$

This is a third order polynomial in $\sin^2(\theta)$. Therefore, its roots can be found analytically. There can be either one or three real roots, which create two or six extremal angles. If there is a single root, the extremal angle corresponding to the maximum is used to determine $\mathbf{g_p}$. Otherwise, the function in Equation 3.3 is smoothed with a Gaussian before selecting the global maximum. In this way, close maxima are merged together, giving the larger maximum a bigger weight. Consequently, all the maxima are considered explicitly. In practice, less than 5% of the curve points have two significant roots with a ratio of their values greater than 0.9, and these cases are handled well. The case in which all three maxima have high function values and a demarcating point should be detected (i.e. satisfying Definition 3.3.2) has not been found in practice.

Calculating demarcating curves: Computing $\mathbf{g_p}$ at every vertex does not suffice for determining the points that satisfy Definition 3.3.2, since the gradient direction is known only for the vertices and not for all the other points on the mesh. An additional problem is that the direction of the gradient $\mathbf{g_p}$ at every vertex of a mesh face might differ, and thus computing the zero crossing of the curvature along a mesh edge would be inappropriate (as we are looking for zero crossing at a certain direction). Since these problems occur in the calculation of other types of mesh curves, our solution is a variation on [98, 25, 56] and is briefly described below.

The demarcating curve points are first estimated along the mesh edges. A mesh edge $[\mathbf{p}_1, \mathbf{p}_2]$ contains a demarcating curve point if $\kappa(\mathbf{g}_{p_1})$ and $\kappa(\mathbf{g}_{p_2})$ have opposite signs (i.e., a zero crossing). The exact location of the demarcating curve point is obtained by linear interpolation of the curvature values. Neighboring demarcating curve points are then connected on a face by a straight line to create the demarcating curve itself.

To solve the second problem, faces whose three gradient vectors differ considerably are eliminated from further consideration. (In our implementation, this happen when the angles between the gradients $> \pi/4$.) For faces in which the gradients of only two vertices are similar, the average gradient of the two similar vertices is selected and the curvature of the third vertex is computed in this direction. Obviously, when this gradient is used for the third vertex, it should be rotated so as to coincide with the vertex's tangent plane, as in [111, 98]. Now, the zero-crossing interpolation can be applied as described above.

It is important to note that the computation described above is performed offline, prior to interaction with the user. The only operation performed during the actual rendering is the elimination of weak curves. The user provides a *strength parameter*, which is the only parameter that the system requires. This parameter is used as a threshold for the precomputed value of the curvature derivative in the gradient direction $(\mathbf{C}_{ijk}\mathbf{g}_p^i\mathbf{g}_p^j\mathbf{g}_p^k)$.

3.4 Relations to other curves

This section discusses relations between demarcating curves and other well-known curves, in particular valleys and ridges, parabolic lines, zero-mean curvature curves, and suggestive contours.

Relation to valleys and ridges: A ridge (valley) point is a point on a manifold, where the positive (negative) principal curvature obtains a maximum (minimum) along its principal direction. Recall that we expect demarcating curves to run between ridges and valleys.

Mathematically, this idea can be modeled by locating the curves on a local smooth step edge – a step edge function convolved with a Gaussian (Figure 3.3). Moreover, demarcating curves run in parallel to ridges and valleys. This is so since in 3D step edges, all normal sections in the $\mathbf{g_p}$ direction are identical, and thus their maxima, minima and zero crossings are equal.

In practice, a demarcating curve will not lie between a valley and a ridge. This is demonstrated in Figure 3.4, where the ridges fail to capture the round structure of the “bumps” on leg of the Armadillo, yet demarcating curves bound these “bumps” (Figure 3.4(c)).

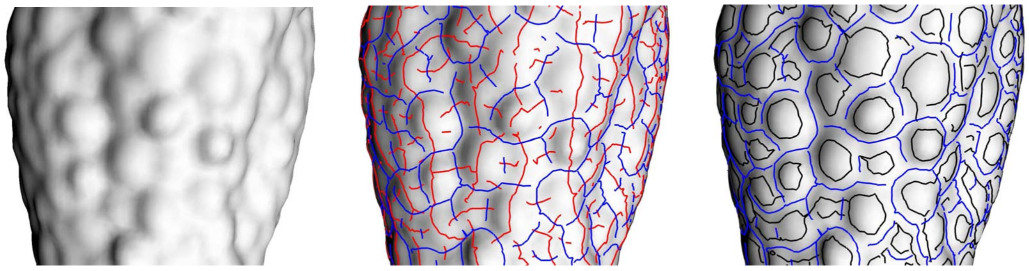


Figure 3.4: **Relation between valleys, ridges, and demarcating curves** (on the Armadillo leg). (a) Zoom in of Armadillo’s leg, (b) Valleys (blue) and ridges (red), (c) Valleys (blue) and demarcating curves (black). The ridges are not well-defined, the valleys do not bound the bumps, whereas demarcating curves perform much better.

Relation to parabolic lines & zero-mean curvature curves: Parabolic (zero-mean curvature) curves are the loci of points with zero Gaussian (mean) curvature. In an ideal surface, where the curves pass through true step edges, zero-mean curvature curves and demarcating curves coincide, since it can be shown that in this case $\mathbf{g_p}$ is a principal direction and both principal curvatures vanish. Moreover, the set of demarcating curve points is a subset of the parabolic curve points, since the Gaussian curvature is zero at demarcating curve points. However, as can be seen in Figure 3.5 (left & middle), demarcating curves are less sensitive to deviations from the ideal surface.

Figure 3.6 shows parabolic lines with increasing threshold values of the curvature derivative in the direction orthogonal to the curve. It can be seen that even with no threshold (left)

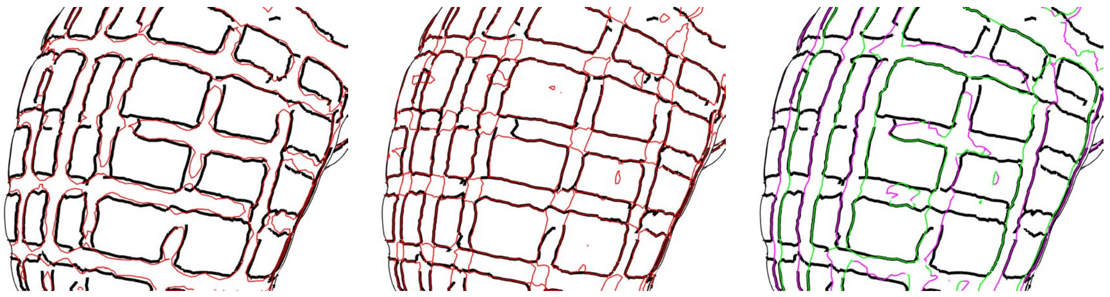
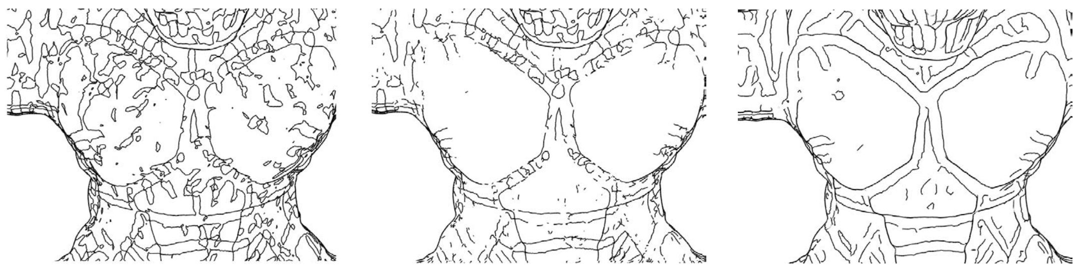


Figure 3.5: **Relation to other curves.** Demarcating curves in black, curves of zero-mean curvature (left) in red, parabolic lines (middle) in red, suggestive contours (right) in green, and suggestive highlights (right) in magenta, on the Armadillo's thigh. The thresholds are all set to zero in order to compare the curves as they are defined. The demarcating curves are closely aligned with the rectangular 3D texture, in contract to the other curves.



(a) Parabolic lines with different threshold values (b) Demarcating curves

Figure 3.6: **Parabolic lines vs. demarcating curves**: zoom into the armadillo's chest

some of the most important lines do not appear. Moreover, as the threshold increases, some of the "good" lines disappear along with the clutter.

Relation to suggestive contours: Given a viewing direction, let $\mathbf{w}$ be its projection onto the tangent plane. The suggestive contour points are the set of all points on a surface at which the curvature $\kappa(\mathbf{w})$ is zero and the directional derivative of $\kappa(\mathbf{w})$ is positive [24].

The set of demarcating curve points is a subset of the union of all the suggestive contour points, viewed from all possible viewing directions. This relationship between the curves simply follows from the fact that they both lie on hyperbolic regions (having negative Gaussian curvature) of the surface. This can also been shown constructively by choosing $w = \mathbf{g_p}$, i.e., the projection of the viewing direction coincides with the gradient direction $\mathbf{g_p}$.

Similarly, it can be shown that the set of demarcating curve points is a subset of the union of all the suggestive highlight points [25].

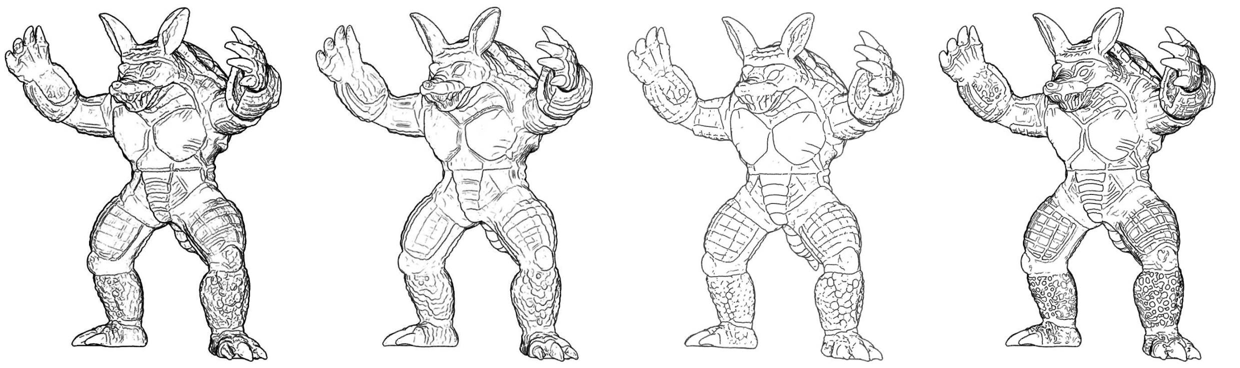
Figure 3.5 (right) demonstrates the relations between the curves. It can be seen that many of the suggestive contours (highlights) coincide with the demarcating curves. However, some of the horizontal curves are missing from the suggestive contours (highlights) in this viewpoint. Moreover, when the suggestive highlights appear noisy, demarcating curves usually do not follow.

3.5 Results and analysis

This section shows results of demarcating curves and compares them to other major curve families. All these curves have only one parameter the user should set. In the examples below, for each of the curves shown, we tried to choose the value that produces the best-looking result for that curve type.

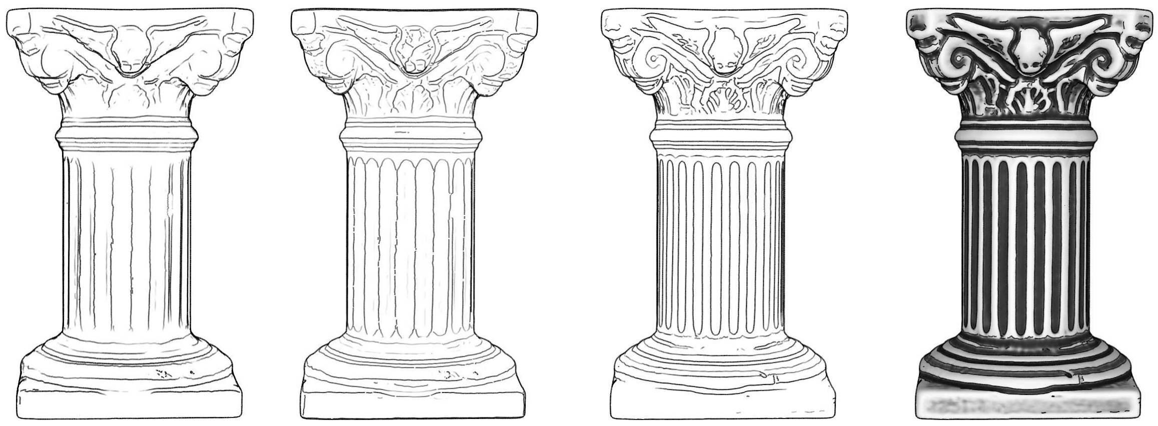
Figure 3.7 compares different curves drawn on the Armadillo (silhouettes were added to all of them). Apparent ridges and suggestive contours do not convey some important features, especially the circular and rectangular “bumps” on the legs and arms, and the teeth. Suggestive contours are biased towards lines parallel to the viewing plane, and thus lines in certain directions are missed. Apparent ridges may ignore curve points whose normal directions are parallel to the viewing direction, since their employed local maximal curvature tends to be larger near the silhouettes. In this example, valleys better illustrate the 3D structure on the thighs. (Adding ridges degrades the drawing.) Demarcating curves are capable of extracting not only this structure, but also the circular 3D structures on the lower legs.

Figure 3.8 shows another comparison between the curves. It can be seen that apparent



(a) Apparent ridges (b) Suggestive contours (c) Valleys (d) Demarcating curves

Figure 3.7: **Armadillo model.** Apparent ridges and suggestive contours do not convey many important features, as can be seen on the upper and lower legs, teeth, and eyes. Even valleys do not convey some of the rectangles on the upper legs and the bumps on the lower legs. Ridges are not shown, since they degrade the drawing. Demarcating curves perform better on this example.



(a) Apparent ridges (b) Valleys & ridges (c) Demarcating curves (d) Shaded curves

Figure 3.8: **Column model.** While lines on the shaft disappear in the apparent ridges drawing and lines on the capital disappear in both apparent ridges and valleys & ridges, they both appear in the demarcating curve drawing.

ridges (and similarly suggestive contours, as shown in [56]) do not detect the structures on the midsection of the column. Valleys & ridges manage to extract these structures, but fail to accurately detect the curves on the upper section. Demarcating curves better carry the shape structure. Figure 3.8(d) illustrates how shading can be used to emphasize the demarcating curves – a topic discussed in the next section.

In contrast to valleys & ridges, demarcating curves convey the shape information without resorting to employ different hues. The application of different hues to distinguish between valleys and ridges is somewhat cumbersome [52]. Moreover, valleys & ridges are less effective for detecting closed curves. Finally, as demonstrated in the top section of the column in Figure 3.8, they do not always convey the structure.

Figure 3.9 shows an example where the view-dependent curves are more appealing and thus may be considered more pleasing for some non-photorealistic applications. Another limitation of demarcating curves is their inability to highlight protruding or depressing features, which lie at surface curvature extremalities.

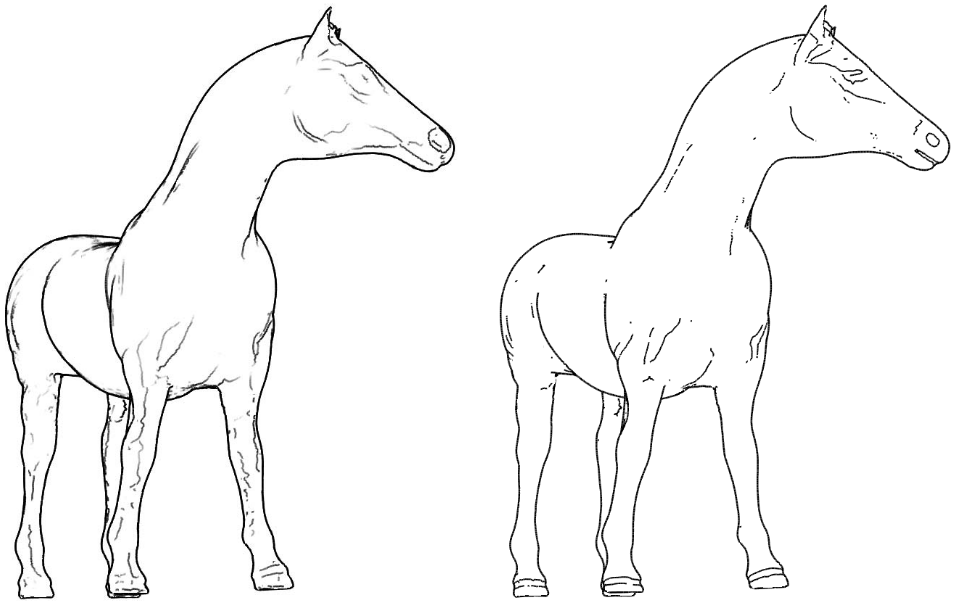

(a) Apparent ridges (b) Demarcating curves

Figure 3.9: **Horse model**

Performance evaluation: Demarcating curves are as quick to compute as ridges and valleys and suggestive contours, since they can be computed prior to rendering. Apparent ridges are more expensive to compute since they rely on view-dependent curvature, which needs to be computed for each viewpoint. On a 2.66 GHz Intel Core 2 Duo PC, our unoptimized C++ implementation computed the demarcating curves in 0.15 seconds for 50K polygon meshes and in 1.1 seconds for 500K polygon meshes.

3.6 Artifact illustration in archaeology

Analysis of archaeological artifacts, such as ceramic vessels, stone tools, coins, seals, figurines etc., is a major source of our knowledge about the past. Traditionally, artifacts are documented and published in 2D photographs, which convey little information about the actual shape (and none about the inner structure) of the objects. The latter properties are described by conventional drawings (Figure 3.2), which contain sections across the artifacts. These are produced manually – by artists – an extremely time-consuming and expensive procedure, prone to inaccuracies and biases.

Digital archaeological reports are slowly spreading around the globe. When scanned 3D representations replace the 2D ones, accurate, automatic curve drawing will be needed.

Demarcating curves are highly beneficial for models that consist of smooth surfaces overlaid with 3D textures (reliefs). Intuitively, this is so since 3D textures, by their very nature, can be considered locally as "almost images." Therefore, the characteristics of the demarcating curves make them especially appropriate.

The current research is conducted as an interdisciplinary effort with several archaeologists, who defined their needs and evaluated intermediate results. Below we present some results of archaeological relics.

Figure 3.10 shows a 3D scan of a handle stamped by a Greek official from which it is impossible to read the text. Suggestive, Apparent, and Ridges & Valleys (Figure 3.11 (a-c)) do not help either. With demarcating curves, we can identify the Greek letters (d). Since the letters are convex and the background is concave, and since demarcating curves demarcate them, it is possible to add a shading scheme to highlight the letters (e-f). It is now possible to read the text as ΜΑΡΣΥΑ ΑΡΤΑΜΙΤΙΟ, where ΜΑΡΣΥΑ[Σ] is the name of an eponym (an official who had the year named after him) and ΑΡΤΑΜΙΤΙΟ[Σ] is the name of a month in the Greek (Rhodian) calendar.

The variant shown in Figures 3.11(e-f) can be generally used for drawing artifacts. Various types of shading schemes can be employed, such as mean-curvature shading [65]

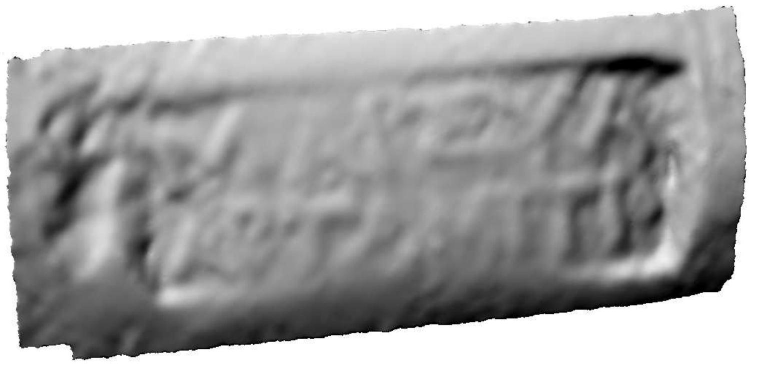


Figure 3.10: **An Hellenistic stamped amphora handle from the first century BCE.**

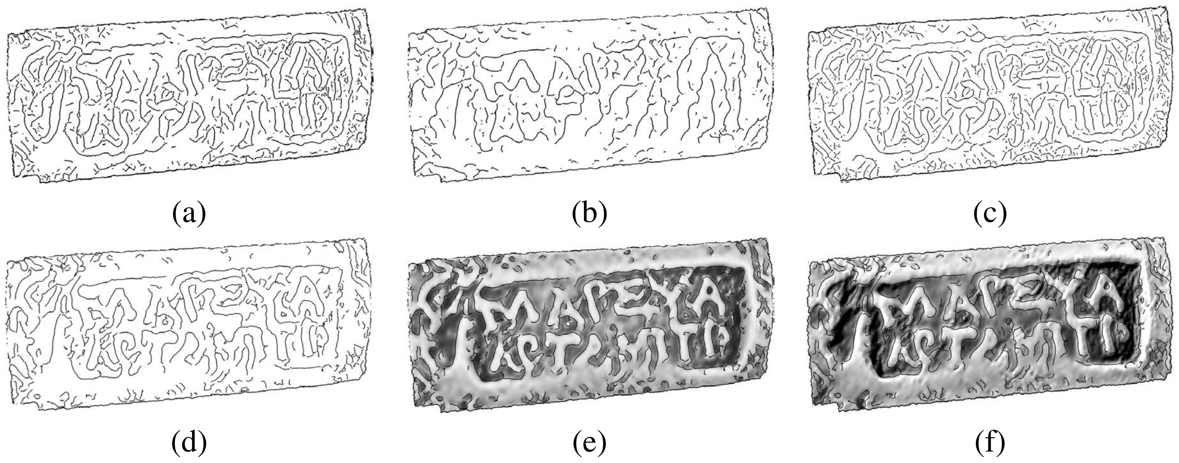
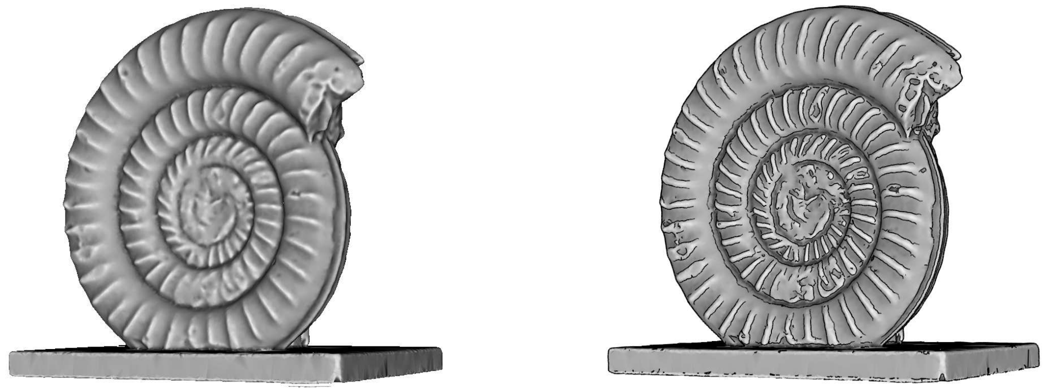


Figure 3.11: **A Hellenistic stamped amphora handle from the first century BCE.** (a) Suggestive contours, (b) Apparent ridges, (c) Ridges & valleys, (d) Demarcating curves, (e) Demarcating curves & mean-curvature shading, (f) Demarcating curves & exaggerated shading. The letters (e.g., Σ) are only visible in (e)-(f)

or exaggerated shading [112]. As discussed in Section 3.4, zero-mean curvature curves and demarcating curves are close to each other. Therefore, using demarcating curves with mean curvature (and often with exaggerated) shading yields eye-pleasing results. The color palette used can vary. Both gray-level shading (Figure 3.11(e-f)) and the palette suggested by [40] (Figure 3.1) are shown.

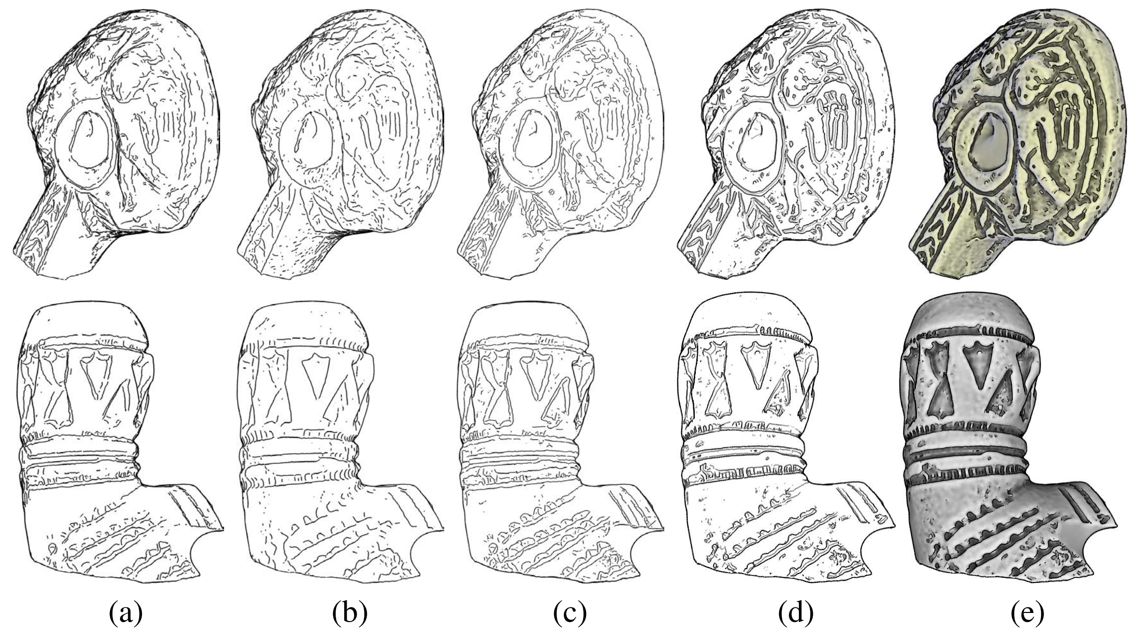
Figure 3.12 compares mean-curvature shading alone with demarcating curves painted on top of the shaded scanned model of a 65 million year old fossil. It can be noted that the



(a) Mean curvature shading (b) Demarcating curves (with mean curvature shading)

Figure 3.12: **Shading options for an Ammonite fossil.**

demarcating curves better emphasize the 3D features, yielding crisper images and making them closer to the way archaeological artists portray artifacts.



(a) (b) (c) (d) (e)

Figure 3.13: **Comparison.** Demarcating curves enhance the wings of Cupid, his naval, and the V-shaped decorations on the Hellenistic lamp (top), and the fine vertical decorations on the Ottoman pipe (bottom). (a) Apparent ridges, (b) Suggestive contours, (c) Ridges & Valleys, (d) Demarcating curves with valleys, (e) Demarcating curves with shading.

Figures 3.13–3.14 show additional results. Demarcating curves enhance the features that

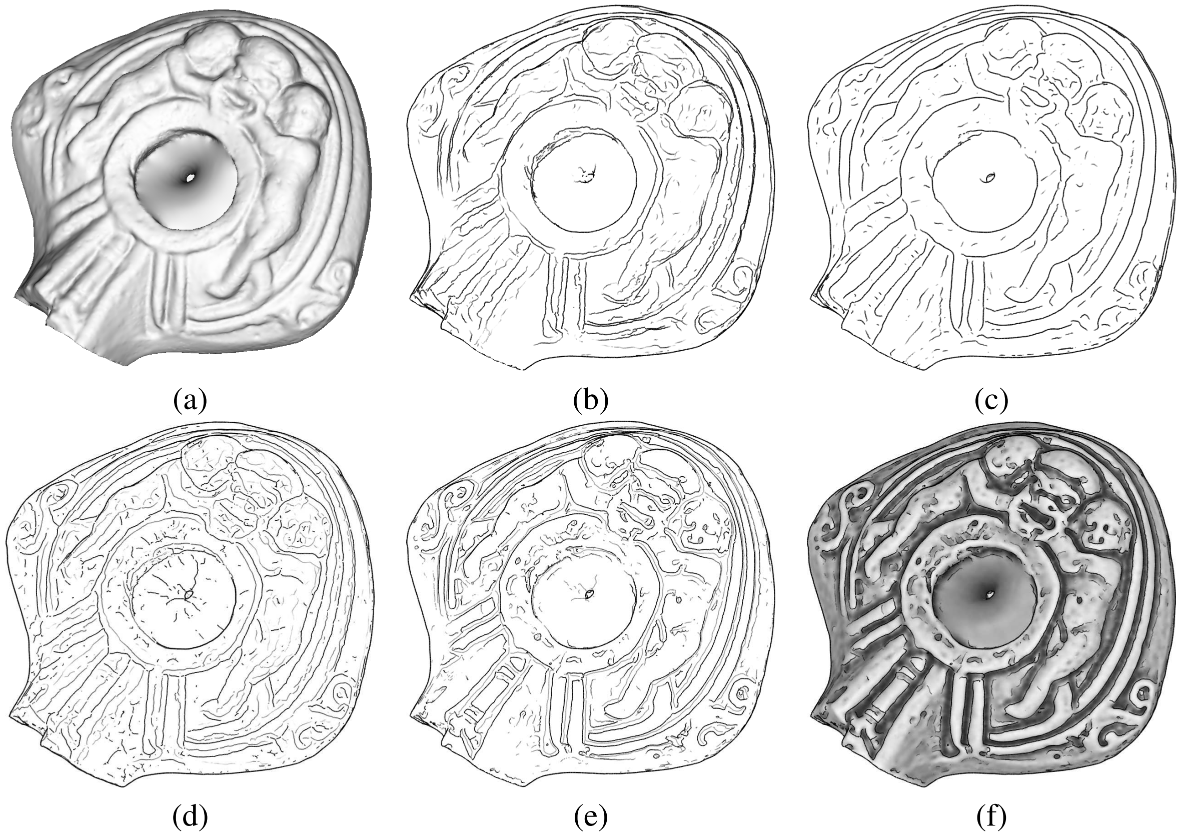


Figure 3.14: **Hellenistic lamp** – Note that the small features, such as the facial features, are difficult to visualize even in the scanned object. Demarcating curves make them visible. (a) Scanned object, (b) Apparent ridges, (c) Suggestive contours, (d) Ridges & valleys (e) Demarcating curves with valleys, (f) Demarcating curves with shading.

are sometimes difficult to visualize with the other curves (and even in the scanned object). Examples include the wings of Cupid, his naval, and the V-shaped decorations in the top Hellenistic lamp; the fine vertical decorations on the bottom Ottoman pipe; and the facial features on the lamp in Figure 3.14.

Figures 3.13(d) & 3.14(e) illustrate a second variant of drawing, where valleys (or ridges) are used to complement demarcating curves. Here, valley lines are also drawn in gray, in order to portray the concave regions.

The results illustrate the robustness of the algorithm to noise. These archaeological objects are all noisy, not only due to the scanning process but also because of their very nature, found after spending more than 2000 years underground.

To compare the suitability of the different curve types to archaeological illustration, we conducted a preliminary user study. Twenty two professional archaeologists from different universities, attending an international conference on *Computer Applications in the Archaeology of the Levant*, participated in the study. Each person was presented with four pages – each page devoted to a single relic (Figures 3.1,3.13,3.14). Each relic was described by six images: the original scanned object and five different drawings, similarly to Figure 3.14. (The images with demarcating curves also included valleys in gray.) The order of the five (untitled) drawings changed from page to page. The archaeologists were asked to rank the drawings according to their appropriateness for replacing the traditional manual illustration. Among the four non-shaded line drawings, 71.5% preferred demarcating curves to the other types, 12.5% preferred valleys & ridges and apparent ridges, and 3.5% preferred suggestive contours. In second place valleys & ridges were preferred to apparent ridges (40% vs. 29%). Moreover, 72% preferred the shaded demarcating curves to the non-shaded line drawings.

In an open discussion, the archaeologists indicated that they prefer our drawings to the traditional manual drawings, both aesthetically and because it is also possible to view the drawings interactively in 3D. Manipulation in 3D enables them to see all the important features, as if they held the artifact in their hand. Moreover, they find view-dependent curves less suitable, since the stability of the curves is paramount. These encouraging results suggest that demarcating curves can be a basis for an illustration tool for archaeology.

3.7 Conclusions

This paper has presented a new class of view-independent curves – *demarcating curves*, defined as the loci of points for which there is a zero crossing of the curvature in the curvature gradient direction. Relations to other types of curves have been discussed.

The utility of the curves for artifact illustration in archaeology has been demonstrated. The results show that demarcating curves effectively capture the 3D information visually. It was welcomed wholeheartedly by the archaeologists.

Since these curves convey meaningful shape information compactly, we intend to utilize them in the future for shape analysis applications, such as similarity based retrieval. In addition, we would like to explore the utility of other types of drawings in archaeology, such as [29].

Acknowledgements: We are grateful to the anonymous referees for their thoughtful comments. This research was supported in part by the Israel Science Foundation (ISF) 628/08, the A.M.N Foundation, and the Joint Technion – Haifa University Research Foundation. We thank Dr. A. Gilboa & Dr. I. Sharon from the Departments of Archaeology at the University of Haifa and the Hebrew University for invaluable discussions and for providing the artifacts and scanning them.

Chapter 4

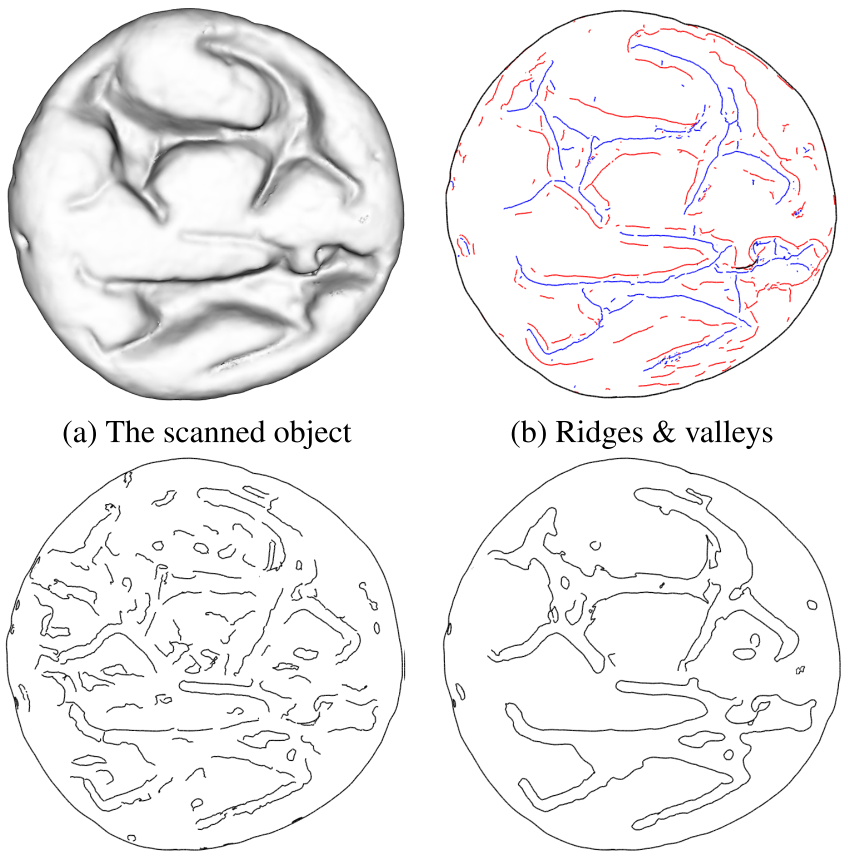
On edge detection on surfaces

Abstract

Edge detection in images has been a fundamental problem in computer vision from its early days. Edge detection on surfaces, on the other hand, has received much less attention. The most common edges on surfaces are ridges and valleys, used for processing range images in computer vision, as well as for non-photorealistic rendering in computer graphics. We propose a new type of edges on surfaces, termed *relief edges*. Intuitively, the surface can be considered as an unknown smooth manifold, on top of which a local height image is placed. Relief edges are the edges of this local image. We show how to compute these edges from the local differential geometric surface properties, by fitting a local edge model to the surface. We also show how the underlying manifold and the local images can be roughly approximated and exploited in the edge detection process. Last but not least, we demonstrate the application of relief edges to artifact illustration in archaeology.

This paper appeared in CVPR 2009.

4.1 Introduction



(a) The scanned object (b) Ridges & valleys

(c) Demarcating curves (d) Relief edges

Figure 4.1: **A seal from the early Iron Age, 11th century BCE**

Edges in images provide low-level cues, which can be utilized in higher level processes, such as object detection, recognition, and classification, as well as motion detection, image matching, and tracking [7, 90]. They are more resilient to image formation parameters than the image intensity values, while containing less information than the whole image.

Edges on surfaces can be used in a similar way [5, 41]. While edges in images can have a variety of causes, such as depth discontinuities, textures, shadows, and other lighting effects that might hinder their use for higher level processes, edges on surfaces are the outcome of the surface geometry only (see Figure 4.1). This paper focuses on the problem of accurately detecting edges on surfaces.

Many of the existing algorithms detect *ridges and valleys*, which are the extrema of principal curvatures [98, 91, 61]. Other types of curves are *parabolic curves*, which partition the surface into hyperbolic and elliptic regions, and *zero-mean curvature curves*, which classify sub-surfaces into concave and convex shapes [68]. They correspond to the zeros of the Gaussian and mean curvature, respectively. Finally, *demarcating curves* are the zero-crossings of the curvature in the curvature gradient direction [71]. While portraying important object properties, the aforementioned curves sometimes fail to capture relevant features, such as weak edges, highly curved edges, and noisy surfaces. As shown in [19], no specific curve fits all applications.

This paper proposes a novel type of surface edges, termed *relief edges*, which addresses these limitations. Consider a surface as an unknown smooth manifold (*base*), on top of which a local height function is defined (e.g., a relief). The function can be considered locally as a standard image defined on the tangent plane of the base. Relief edges are the edges of this local image, i.e., a surface point $\mathbf{p}$ is a relief edge point if it is an edge point of this image.

We demonstrate that relief edges are smoother and more accurate than the other types of curves. They are better suited for certain surfaces, such as reliefs prevalent in archaeological artifacts.

The main contributions of the paper is thus threefold. First, we extend the definition of edges from functions on a plane to functions on an unknown manifold. Second, we describe an algorithm that extracts these edges. Finally, we demonstrate the utility of these edges in archaeological artifact illustration.

Algorithm overview: Relief edges are defined as the zero crossings of the normal curvature in the direction perpendicular to the edge. Initially, the edge direction is estimated for every point by fitting a step edge model to the surface. Given the edge directions, the precise edge localization is obtained (Section 4.3).

The quality of the estimation of the edge directions is further improved (Section 4.4). First, a rough estimation of the base normal is employed to limit the range of possible edge directions. Second, the edge directions are smoothed, while maintaining the properties of relief edges.

4.2 Related work

The paper proposes an extension of edge detection in images to arbitrary 2D surfaces. Hence, this section presents related work both on images and on surfaces. It does not describe volumetric edges [151] that are mere extensions of 2D edges to a higher dimension.

Edge detection in images: Edge detection has been extensively investigated [37]. Our work is most closely related to gradient-based edge detection, which can be generally classified into two classes.

The first class defines edges as the maximum of a smoothed first derivative or zero crossings of a smoothed second derivative. These methods differ in the manner in which they smooth and the way the derivatives are calculated. Examples include the maximum of the derivative of the Gaussian filter [14], the zero crossings of the Laplacian of the Gaussian [86], and the cubic spline filter [134].

Other methods attempt to implicitly fit the data to an edge model, such as a parametric-feature model [3] or a 1D polynomial [94]. The fitting determines both the orientation and the strength of the edge. These algorithms strongly rely on the edge model and thus might fail when the underlying assumption of the edge is unsuitable.

Our approach most resembles [87], which combines both types of edge detection algorithms. Canny edge detection is used for the initial edge estimation, followed by verification that is based on the correlation of the data with an edge template. This significantly increases the ability of the detector to eliminate spurious edges and deal with weak edges.

Edge detection on surfaces: There are two classes of edges on surfaces. The first includes ridges and valleys [98, 91, 61], which are the loci of points at which the curvature obtains extrema along the principal direction. They occur at surface normal discontinuities. Ridges and valleys portray important object properties. However, illustrating the object only by valleys (or ridges) is often insufficient, since they do not always convey its structure. Drawing both will overload the image with too many lines. It should be noted that relief edges do not compete with ridges and valleys, but rather complement them, since they portray locations with different geometric properties.

The second class includes curves that are defined as the zero crossings of some function of curvature. Examples include parabolic lines (zeros of Gaussian curvature) [68], curves of zero-mean curvature [68], and demarcating curves (zeros of the normal curvature in the curvature gradient direction) [71]. Parabolic curves are demonstrated to be noisy and unreliable [24, 71]. The curves of zero-mean curvature depend on the curvatures both along the edge and in the direction perpendicular to it; hence their error is high when the curvature in the edge direction is large. Both types of curves are isotropic operators and suffer from similar flaws as isotropic edges in images (e.g., Laplacian), such as poor behavior at corners and inexact edge localization [37]. Demarcating curves might be noisy when the curvature along the edge varies. The relief edges proposed in this paper belong to this class and address these problems.

Other kinds of curves are view dependent, i.e., they change when the viewpoint changes [24, 56, 143]. These curves are often aesthetically pleasing and thus are applicable for non-photorealistic rendering in computer graphics.

4.3 Relief edges

Given a surface $S(u,v) : \mathbb{R}^2 \to \mathbb{R}^3$, we assume that it consists of a smooth base surface $B(u,v) : \mathbb{R}^2 \to \mathbb{R}^3$ and a function (local image) $I(u,v) : \mathbb{R}^2 \to \mathbb{R}$ defined on B:

$$S(u,v) = B(u,v) + \bar{\mathbf{n}}(u,v)I(u,v), \tag{4.1}$$

where u and v are the coordinates of a planar parametrization and $\bar{\mathbf{n}}(u,v) : \mathbb{R}^2 \to \mathbb{S}^2$ is the normal of B ($\mathbb{S}^2$ is the unit sphere). We assume that B is locally a manifold and that its curvature has a smaller value than the curvature of I (Figure 4.2). The decoupling of S into B and I is unknown. Note that in the special case of an image, B is the image plane, $\bar{\mathbf{n}}(u,v)$ is constant, and I is the image intensity.

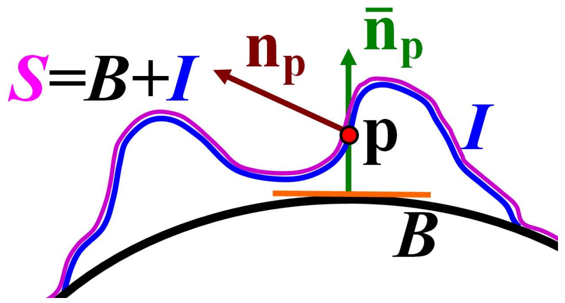


Figure 4.2: **Surface representation.** The surface S (magenta) is composed of a smooth base B (black) and a function I (blue). Function I at point $\mathbf{p}$ can be locally viewed as an image defined on the tangent plane (orange) of the base. Point $\mathbf{p}$ is a relief edge point if it is an edge point of this image. The normal $\mathbf{n_p}$ (brown) is the normal of S and $\bar{\mathbf{n}}_\mathbf{p}$ (green) is the normal of B corresponding to $\mathbf{p}$.

The goal is to detect edges on S that correspond to edges on the local images I. We consider the common definition of edges in images, as points at which the derivative obtains a maximum in the gradient direction. We will show that the edges can be detected without accurately estimating B or its normal $\bar{\mathbf{n}}$ – a rough estimate suffices.

In the following we first provide the necessary mathematical background and then describe the computation of the relief edges.

4.3.1 Background

Before defining the curves, we review some definitions in differential geometry [30]. The *normal section* of a surface at a point $\mathbf{p}$ in a tangent direction $\mathbf{v}$ is the intersection of the surface with the plane defined by $\mathbf{v}$ and the normal to the surface at $\mathbf{p}$. The *normal curvature* at point $\mathbf{p}$ in direction $\mathbf{v}$ is the curvature of the normal section at $\mathbf{p}$.

For a smooth surface, the normal curvature in direction **v** is

$$\kappa(\mathbf{v}) = \mathbf{v}^T \mathbf{II} \mathbf{v}. \tag{4.2}$$

The symmetric matrix **II** is the second fundamental form:

$$\mathbf{II} = \begin{bmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{bmatrix}, \tag{4.3}$$

where κ_1 and κ_2 are the principal curvatures.

The derivatives of the curvature are defined by a $2 \times 2 \times 2$ tensor with four unique numbers:

$$\mathbf{C} = (\partial_u \mathbf{II}; \partial_v \mathbf{II}) = \left[\begin{pmatrix} a_1 & a_2 \\ a_2 & a_3 \end{pmatrix}; \begin{pmatrix} a_2 & a_3 \\ a_3 & a_4 \end{pmatrix} \right], \tag{4.4}$$

where ∂_u and ∂_v are the derivatives along the principal directions. Multiplying C from its three sides by a direction vector **v**, $C_{ijk}\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k$ gives a scalar, which is the derivative in direction **v** of the curvature in this direction.

The *Monge form* is a polynomial approximation of a surface S on the tangent plane at a given point, expressed as:

$$\begin{aligned} S(\mathbf{v}) = \frac{1}{2}\mathbf{v}^T \mathbf{II} \mathbf{v} + \frac{1}{2} C_{ijk}\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k = \\ \tfrac{1}{2}(\kappa_1 u^2 + \kappa_2 v^2) + \tfrac{1}{6}(a_1 u^3 + 3a_2 u^2 v + 3a_3 u v^2 + a_4 v^3), \end{aligned} \tag{4.5}$$

where u and v are the coordinates of **v** in the principal directions. We will be using the Monge form to locally estimate the surface, utilizing the state-of-the-art techniques developed for estimating the curvature and its derivative [111, 76, 42].

4.3.2 Computing relief edges

Relief edges are computed in two steps: estimating the edge direction at every point and determining the relief edge points using this estimation. We elaborate on these steps below.

Estimating the edge direction: The edge direction is estimated by fitting an edge model that best approximates the surface locally. Below we first describe our edge model and then the process of fitting it to the surface.

We utilize the commonly-used smoothed step edge to model relief edges. Since B is unknown, all our computations are performed with respect to the local tangent plane of S at $\mathbf{p}$, where the principal directions define the coordinate system on this plane. To locally approximate surface S, we use the Monge form polynomial (Equation 4.5). A smoothed step edge E passing through point $\mathbf{p}$ in direction $(-\sin(\theta), \cos(\theta)) \equiv (-s, c)$ can be approximated by a cubic polynomial as:

$$\begin{aligned} E(\theta, \alpha, u, v) &= \frac{1}{6}\alpha(cu+sv)^3 = \\ &= \frac{1}{6}\alpha(c^3u^3 + 3c^2su^2v + 3cs^2uv^2 + s^3v^3), \end{aligned} \tag{4.6}$$

where α is the edge intensity, and u and v are the local coordinates. (Note that the other coefficients of the polynomial are zero because the step edge is constant along its direction and antisymmetric in the perpendicular direction.)

Using polar coordinates: $(u, v) = (\rho\cos(\phi), \rho\sin(\phi)) \equiv (\rho\tilde{c}, \rho\tilde{s})$, Equations 4.5 and 4.6 can be rewritten as:

$$\begin{aligned} S &= \frac{\rho^2}{2}(\kappa_1\tilde{c}^2 + \kappa_2\tilde{s}^2) + \frac{\rho^3}{6}(a_1\tilde{c}^3 + 3a_2\tilde{c}^2\tilde{s} + 3a_3\tilde{c}\tilde{s}^2 + a_4\tilde{s}^3), \\ E(\theta, \alpha) &\equiv \alpha\tilde{E}(\theta) = \alpha\frac{\rho^3}{6}(c^3\tilde{c}^3 + 3c^2s\tilde{c}^2\tilde{s} + 3cs^2\tilde{c}\tilde{s}^2 + s^3\tilde{s}^3). \end{aligned}$$

We define the orientation of the edge as the direction that best fits the edge model. In other words, we seek $(\hat{\theta}, \hat{\alpha})$ that minimize the difference between $E(\hat{\theta}, \hat{\alpha})$ and S. We define the approximation error as:

$$\text{Err}(\theta,\alpha) = \int \|E(\theta,\alpha) - S\|^2 \rho d\rho d\phi, \tag{4.7}$$

where the integral is defined over a neighborhood of $\mathbf{p}$ and ρ is the Jacobian of the polar coordinates substitution. The optimal edge is determined by $(\hat{\theta},\hat{\alpha}) = \arg\min \text{Err}(\theta,\alpha)$.

We reformulate Equation 4.7 in terms of vectors in the polynomial space of cos and sin. This formulation allows us to represent our optimization problem as the problem of finding the roots of a third-order polynomial of $\sin^2(\theta)$, as explained below.

Let the basis vectors and their inner product be:

$$\mathbf{x}_1 = \tilde{c}^3,\ \mathbf{x}_2 = \tilde{c}^2\tilde{s},\ \mathbf{x}_3 = \tilde{c}\tilde{s}^2,\ \mathbf{x}_4 = \tilde{s}^3,\ \mathbf{x}_5 = \tilde{c}^2,\ \mathbf{x}_6 = \tilde{s}^2,$$

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \int_{\phi=0}^{2\pi} \mathbf{x}_i \mathbf{x}_j d\phi. \tag{4.8}$$

Surface S, the step edge E, and the error $\text{Err}(\theta,\alpha)$ can be rewritten in terms of the basis vectors $\mathbf{x}_i$ as:

$$\begin{aligned} S &= \frac{\rho^3}{6}(a_1\mathbf{x}_1 + 3a_2\mathbf{x}_2 + 3a_3\mathbf{x}_3 + a_4\mathbf{x}_4) + \frac{\rho^2}{2}(\kappa_1\mathbf{x}_5 + \kappa_2\mathbf{x}_6) \\ &= \frac{\rho^3}{6}S_1 + \frac{\rho^2}{2}S_2, \\ \tilde{E}(\theta) &= \frac{\rho^3}{6}(c^3\mathbf{x}_1 + 3c^2s\mathbf{x}_2 + 3cs^2\mathbf{x}_3 + s^3\mathbf{x}_4) = \frac{\rho^3}{6}E_1(\theta), \\ \text{Err}(\theta,\alpha) &= \int_\rho \|\alpha\tilde{E}(\theta) - S\|^2 \rho d\rho, \end{aligned} \tag{4.9}$$

where the norm is calculated according to the inner product in Equation 4.8.

$$\begin{aligned}\mathrm{Err}(\theta,\alpha) = & \\ \alpha^2 \int \|\tilde{E}\|^2(\theta)d\rho + \int \|S\|^2 d\rho - 2\alpha \int \langle \tilde{E}(\theta), S\rangle d\rho = & \\ = \alpha^2 \|E_1\|^2 \int \rho \tfrac{\rho^6}{36} d\rho + \int \|S\|^2 d\rho - & \\ -2\alpha \langle E_1, S_1\rangle \int \rho \tfrac{\rho^6}{36} d\rho - 2\alpha \langle E_1, S_2\rangle \int \rho \tfrac{\rho^4}{4} d\rho. &\end{aligned} \tag{4.10}$$

Appendix A proves that $\langle E_1, S_2\rangle = 0$. The value of $\|S\|^2$ is independent on θ and α and can be removed. Therefore, the optimal parameters need to minimize:

$$(\hat{\theta},\hat{\alpha}) = \arg\min\ (\alpha^2\|E_1\|^2 - 2\alpha\langle E_1, S_1\rangle) \int \rho \frac{\rho^6}{36} d\rho. \tag{4.11}$$

It is interesting to note that by Equation 4.11, $\hat{\theta}$ and $\hat{\alpha}$ are independent on the size of the region on which the integral is computed. Since the magnitude $\|E_1\|$ of the edge is independent of its direction θ, $\hat{\theta}$ should maximize the edge–surface correlation $\langle E_1(\theta), S_1\rangle$:

$$\hat{\theta} = \arg\max \langle E_1(\theta), S_1\rangle. \tag{4.12}$$

In Appendix A we show that:

$$\hat{\theta} = \arg\max(c^3C_1 + c^2sC_2 + cs^2C_3 + s^3C_4), \tag{4.13}$$

where the C_is are scalars depending on the parameters of the curvature derivative tensor.

After $\langle E_1(\hat{\theta}), S_1\rangle$ has been computed, $\hat{\alpha}$ is:

$$\hat{\alpha} = \frac{\langle E_1(\hat{\theta}), S_1\rangle}{\|E_1(\hat{\theta})\|^2}. \tag{4.14}$$

Equations 4.13 and 4.14 determine the orientation and the intensity of the best fitting edge. In [71], it is shown that the maxima of an equation of the type of Equation 4.13 correspond to the roots of a cubic polynomial in $\sin^2(\theta)$, and thus the polynomial may have up to three maxima. Multiple maxima appear when there are several step edges that can locally fit the surface. Section 4.4 describes our method for choosing the appropriate one.

Determining the relief edges: The previous step computed $\hat{\theta}$ and $\hat{\alpha}$ for every point on the surface. Our goal is to find the edge points, which are the loci of points where the gradient obtains maximum in the gradient direction.

In [134] it is shown that the maximum of the gradient in the gradient direction corresponds to the zeros of the normal curvature in this direction. For a smoothed step edge, the gradient direction is perpendicular to the edge direction. Therefore, the loci of the relief edges are the zero crossings of the curvature in the direction perpendicular to the edge direction $\hat{\theta}$.

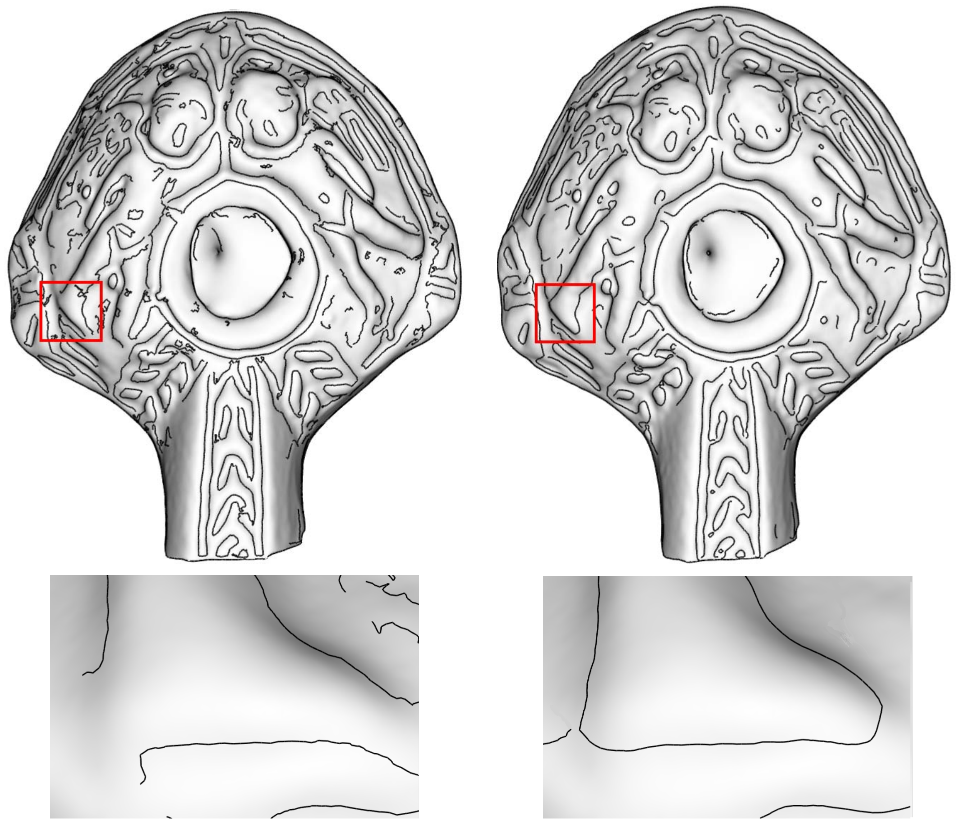
We can now formally define a relief edge point. Let $\mathbf{g_p} = [\cos(\hat{\theta}), \sin(\hat{\theta})]$ be a vector perpendicular to the edge direction at point $\mathbf{p}$, and let $\mathbf{G_p} \equiv \mathbf{g_p}^T \mathbf{II} \mathbf{g_p}$ be the value of the normal curvature in the gradient direction at $\mathbf{p}$.
Definition 4.3.1. *Point* $\mathbf{p}$ *is a relief edge point iff* $\mathbf{G_p} = 0$.

The algorithm is applied to meshes. To achieve sub-vertex accuracy, points on the mesh edges satisfying the constraint are found. We use the method in [71, 98], which accurately estimates the zero curves of a function on a mesh, given the function values on the vertices.

In the implementation, we threshold the error defined in Equation 4.7, normalized by $||S||^2$, which reflects the dissimilarity of the surface to the edge model. This removes points that satisfy Definition 4.3.1, but do not resemble step edges.

4.4 Enhancing relief edges

The algorithm proposed in the previous section usually produces high quality edges, as can be seen in Figure 4.1. However, when the surface is very noisy or deviates from the step edge model, the resulting curves might be noisy or incorrect, as illustrated in Figure 4.3. This section describes how to handle these cases, by utilizing the relief surface model illustrated in Figure 4.2.



(a) Relief edges (b) Enhanced relief edges

Figure 4.3: **A late Hellenistic lamp (150-50 BCE)**: top, full object; bottom, zoom in. Note the closing of the outline of Cupid's foot due to correcting the edge orientation and the smooth edges resulting from the smoothing procedure.

Choosing edge orientations: The orientation obtained by Equation 4.13 is optimal for edges well-approximated by the step model. Deviations from the model, such as when several maxima exist in Equation 4.13, might lead to erroneous orientations. Though these deviations are rare, we present a method that aids in choosing the correct orientation.

The method uses a rough estimation of B's normal $\bar{n}_p$ to determine the possible orientations. Obviously, if $\bar{n}_p$ were known, the edge orientation could be computed precisely, by using an edge detector on the local image (the plane perpendicular to the normal). Though our normal's rough estimation is insufficient for a precise calculation of the edge, it suffices to limit the range of possible orientations.

To do that, we modify the function maximizing Equation 4.13. Let $f_{\text{corr}}(\theta) = c^3C_1 + c^2sC_2 + cs^2C_3 + s^3C_4$ be the original correlation of the edge and the surface and let $f_{\text{base}}(\theta)$ be a function that limits the range of orientations, using the estimated base normal (defined below). We define the modified function as:

$$f_{\text{mod}}(\theta) = f_{\text{corr}}(\theta) \cdot f_{\text{base}}(\theta). \tag{4.15}$$

If $\bar{n}_p$ were known, the edge direction θ_{base} could be calculated as the projection of $\bar{n}_p$ on the local tangent plane. Then, $f_{\text{base}}(\theta) = \delta(\theta - \theta_{\text{base}})$, where δ is the Kronecker delta function. Since $\bar{n}_p$ is known only approximately, we use:

$$f_{\text{base}}(\theta) = r_w(\theta - \theta_{\text{base}}), \tag{4.16}$$

where

$$r_w(x) = \begin{cases} 1 & \|x\| \leq w \\ 0 & \|x\| > w. \end{cases} \tag{4.17}$$

Below we describe how to calculate the rough estimation of $\bar{n}_p$ and the width w.

To calculate $\bar{n}_p$, the surface (S) normals are smoothed at the neighborhood of the point. This neighborhood should be sufficiently large, so as to reduce the influence of the local features on the estimated normal. Our approach utilizes an adaptive Gaussian filter, similarly to [97]. However, since the σ of the smoothing Gaussian in [97] estimates the size of the local feature, it is unsuitable for estimating the base surface. We therefore use a three times larger σ . This enables us to average the normals of several features and thus achieve a better approximation.

To calculate width w, we first compute the directions θ_{base} and $\hat{\theta}$ (Equation 4.13) for all the vertices. Then, w is set to the standard deviation of the histogram of the error $||\hat{\theta} - \theta_{\text{base}}||$. Assuming that most of the values of $\hat{\theta}$ are correct, w is statistically meaningful. When the edge is weak, its gradient estimation is unreliable, and thus it is removed, by setting $f_{\text{base}}(\theta) \equiv 1$. In the implementation, weak edges are characterized by a small angle between the base normal ($\angle(n_p, \bar{n}_p) \leq 11^\circ$). The value $1/(n_p \cdot \bar{n}_p)$ is proportional to the magnitude of the local image gradient. This measure is most commonly used to threshold edges.

Since we are utilizing two thresholds – one that measures the similarity to the edge model (Section 4.3.2) and one that measures the edge strength, we can combine them to produce better results using a two-dimensional hysteresis.

Edge smoothing: While the method described above captures the features correctly, scanning noise and edge direction estimation errors may cause the edges to become jagged. In this case, smoothing should be applied. While smoothing could be applied to the edges themselves, this correction would not relate to the geometry of the surface. Therefore, a smoothing scheme which indirectly smoothes the edges is proposed.

This is done by first smoothing the function $\mathbf{G_p}$, which is defined at every vertex, yielding $\hat{\mathbf{G}}_\mathbf{p}$. Then, we compute the updated edge directions $\hat{\mathbf{g}}_\mathbf{p}$ that satisfy:

$$\hat{\mathbf{G}}_\mathbf{p} = \hat{\mathbf{g}}_\mathbf{p}^T \mathbf{II} \hat{\mathbf{g}}_\mathbf{p}. \tag{4.18}$$

When such a direction does not exist (e.g., when $\hat{\mathbf{G}}_\mathbf{p}$ is required to have a negative value at a point with two positive principal curvatures), the direction that minimizes the error $||\mathbf{G_p} - \hat{\mathbf{G}}_\mathbf{p}||$ is chosen. Finally, $\mathbf{G_p}$ is recalculated according to $\hat{\mathbf{g}}_\mathbf{p}$.

We observed that good results are achieved when simple Gaussian smoothing is used to smooth $\mathbf{G_p}$. The smoothing parameter can be controlled by the user. In all our experiments, the σ of the Gaussian is equal to 0.8 of the median edge length of the mesh.

4.5 Results

This section shows results of relief edges and compares them to other major edge families. While relief edges can be used on any object, as shown in Figure 4.4, we focus on the challenging archaeological artifacts, which are noisy and contain edges which are difficult to detect.

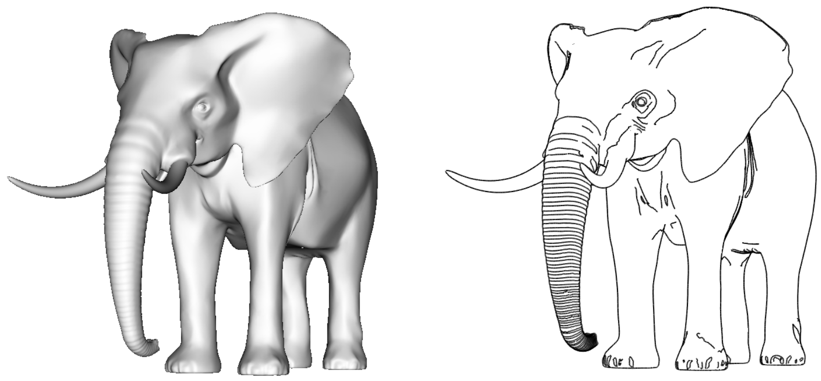
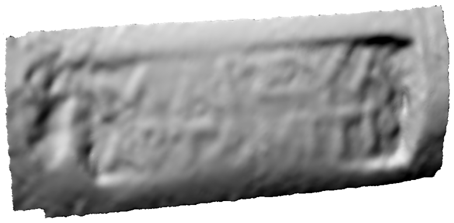
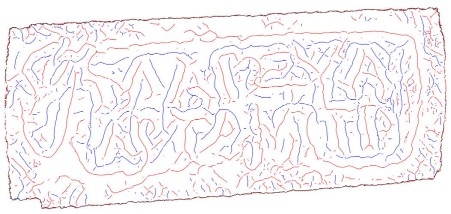


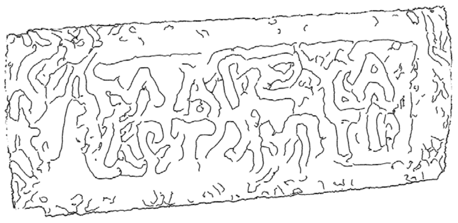
Figure 4.4: **Elephant model**. Note that the relief edges are shown together with the surface contours.

Analysis of archaeological artifacts such as ceramic vessels, stone tools, coins, seals and figurines is a major source of our knowledge about the past. Traditionally, archaeological artifacts are drawn by hand and printed in the reports of archaeological excavations. These are produced manually by artists, in an extremely time-consuming and expensive procedure, prone to inaccuracies and biases. The main purpose of these drawings is to depict the features of the 3D object so that the archaeologist can visualize and compare artifacts. Thus, all the major features (edges) have to be detected. When, in the near future, digitization of the findings by high resolution scanners will replace the 2D representations, accurate, automatic curve drawing will be needed.

Figures 4.5–4.8 show some results. Figures 4.5–4.6 demonstrate the importance of choosing the correct orientation of the edges. Since the edges pass on almost flat surfaces, the


(a) The object


(b) Ridges & valleys


(c) Demarcating curves

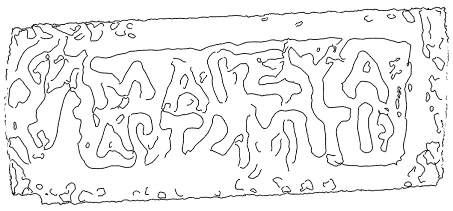
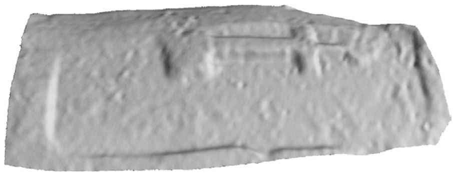
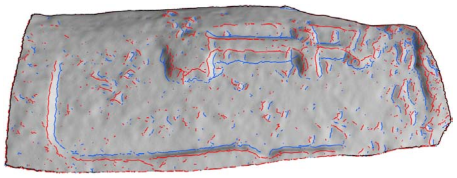
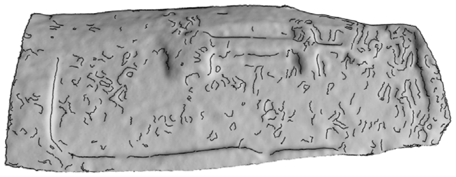

(d) Relief edges

Figure 4.5: **Hellenistic stamped amphora handle from the first century BCE.** While the text is hardly legible in the 3D object, relief edges make most of the letters visible and improve on the alternatives. The text reads ΜΑΡΣϒΑ ΑΡΤΑΜΙΤΙ∘.


(a) The object


(b) Ridges & valleys


(c) Demarcating curves

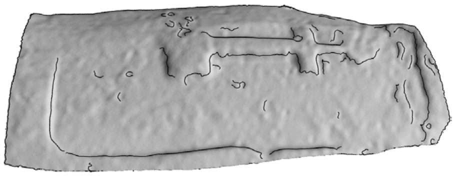

(d) Relief edges

Figure 4.6: **Hellenistic stamped amphora handle from the first century BCE.** This is an example of a noisy surface. Only relief edges manage distinguish between the edges and the noise utilizing the approximated base surface.

base normal can be calculated accurately and aid in estimating the edge direction. Figure 4.6 is a difficult object, due to the high level of noise. Locally true edges and noisy surfaces look similar and therefore demarcating curves fail to differentiate between them. Relief edges on the other hand exploit the approximated base for estimating the local image gradient. In addition, relief edges perform better at places where the curve curvature is not constant.

Figures 4.7–4.8 demonstrate models having non-planar bases. In particular, the base surface of Figure 4.8 is quite complex. As can be seen, relief edges outperform the other types of edges.

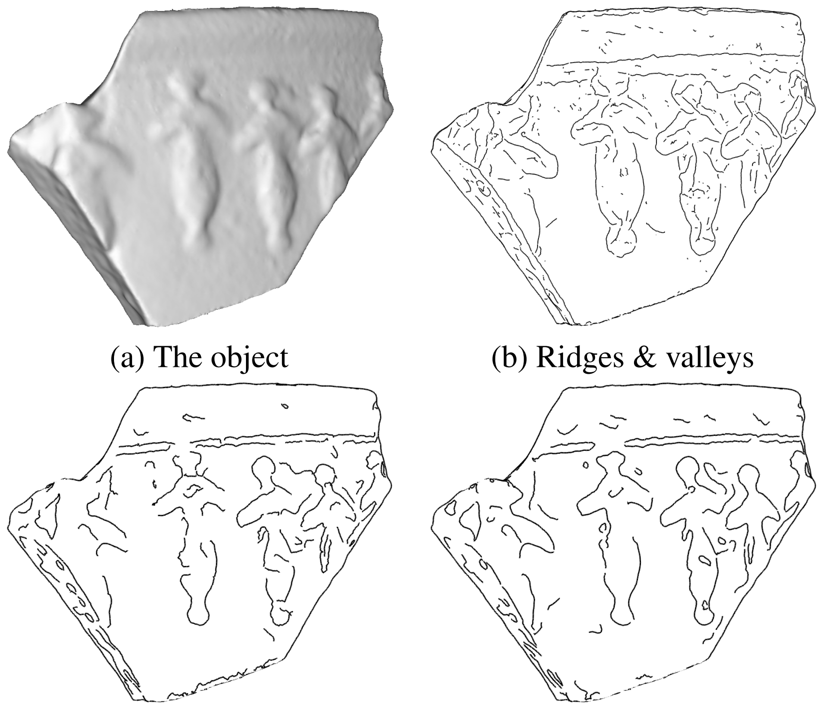
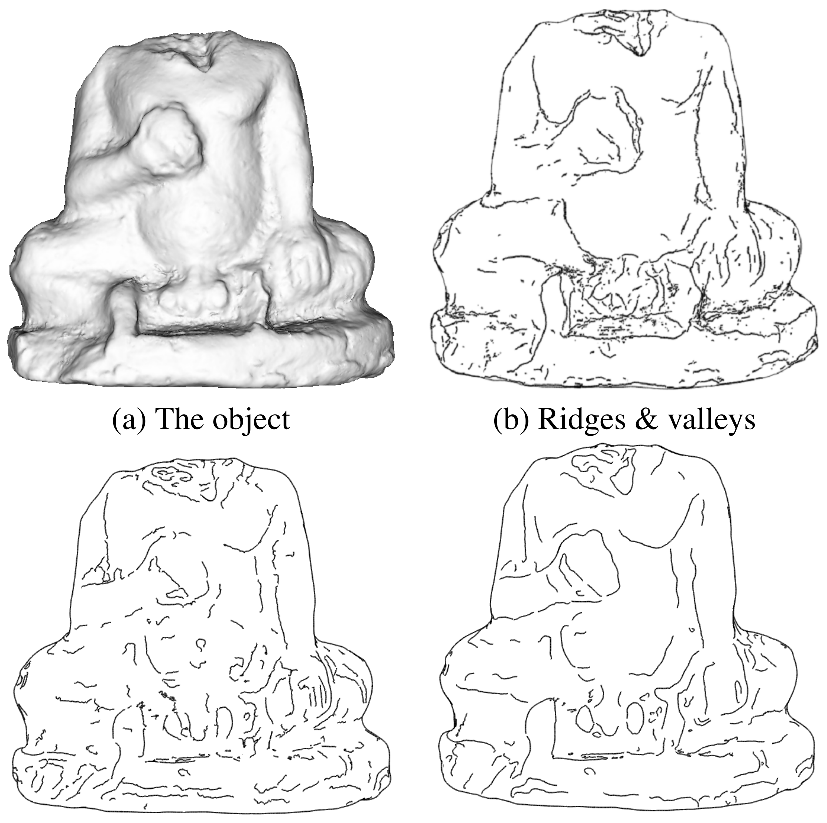

(a) The object (b) Ridges & valleys (c) Demarcating curves (d) Relief edges

Figure 4.7: **Hellenistic vase.** The figures are well-depicted with long meaningful edges. Note especially the quality of the recovered arms where the curvature of the edges change considerably .

The algorithm was implemented in C++ using the trimesh2 library by S. Rusinkiewicz. On a 2.66 GHz Intel Core 2 Duo PC all the steps of the algorithm run in real time except the estimation of the base normal which currently takes 16 seconds for a surface of 50K vertices and 55 seconds for a surface of 140K vertices.


(a) The object (b) Ridges & valleys

(c) Demarcating curves (d) Relief edges

Figure 4.8: **Figurine from the Persian period (4th c. BCE).** The relief edges are continuous and smoother than the alternatives. Noisy edges have been successfully removed.

4.6 Conclusion

This paper has extended the definition of edges from images to surfaces, for which image edges are a special case. These edges, termed relief curves, use local intrinsic surface properties together with a rough approximation of the base surface to produce superior results.

The results show that relief edges manage to capture the 3D features. They have been utilized to draw edges on scanned objects for artifact illustration in archaeology. In the future we intend to utilize these edges for shape-matching applications, which is an important challenge in archaeology, as well as in computer vision in general.

Acknowledgements: This research was supported in part by the Israel Science Foundation (ISF) 628/08, the Ollendorff foundation, and the Joint Technion University of Haifa Research Foundation. We thank Dr. A. Gilboa and the Zinman Institute of Archaeology at the University of Haifa.

Appendix A: Inner products of polynomials

In Equation 4.8, the inner products of the basis functions need to be computed. When the exponent of $\sin(\phi)$ or $\cos(\phi)$ is odd, the inner product is zero. Otherwise, the inner product is defined by the Euler beta function:

$$\mathfrak{B}(x,y) = 2\int_{\phi=0}^{\pi/2} (\sin(\phi))^{2x-1}(\cos(\phi))^{2y-1}d\phi.$$

Thus, $\langle \mathbf{x}_1, \mathbf{x}_1\rangle = \langle \mathbf{x}_4, \mathbf{x}_4\rangle = \frac{5}{8}\pi \equiv A$,

$$\langle \mathbf{x}_2, \mathbf{x}_2\rangle = \langle \mathbf{x}_3, \mathbf{x}_3\rangle = \langle \mathbf{x}_1, \mathbf{x}_3\rangle = \langle \mathbf{x}_2, \mathbf{x}_4\rangle = \frac{\pi}{8} \equiv B. \tag{4.19}$$

We can now calculate the inner products in Section 4.3.2:

$$\begin{aligned} &\langle E_1, S_2\rangle = \\ &\quad \langle c^3\mathbf{x}_1 + 3c^2s\mathbf{x}_2 + 3cs^2\mathbf{x}_3 + s^3\mathbf{x}_4, k_1\mathbf{x}_5 + k_2\mathbf{x}_6\rangle = 0, \end{aligned} \tag{4.20}$$

$$\begin{aligned} \|E_1\|^2 &= \|c^3\mathbf{x}_1 + 3c^2s\mathbf{x}_2 + 3cs^2\mathbf{x}_3 + s^3\mathbf{x}_4\|^2 = \\ &= c^6\langle \mathbf{x}_1, \mathbf{x}_1\rangle + 9c^4s^2\langle \mathbf{x}_2, \mathbf{x}_2\rangle + 9c^2s^4\langle \mathbf{x}_3, \mathbf{x}_3\rangle \\ &\quad + s^6\langle \mathbf{x}_4, \mathbf{x}_4\rangle + 6c^4s^2\langle \mathbf{x}_1, \mathbf{x}_3\rangle + 6c^2s^4\langle \mathbf{x}_2, \mathbf{x}_4\rangle \\ &= \cdots = ((1 - 3c^2 + 3c^4))A + 3A(c^2 - c^4) = A, \end{aligned}$$

$$\begin{aligned}
\langle E_1, S_1\rangle = c^3 a_1\langle \mathbf{x}_1, \mathbf{x}_1\rangle + 9c^2 s a_2\langle \mathbf{x}_2, \mathbf{x}_2\rangle + \\
+9cs^2 a_3\langle \mathbf{x}_3, \mathbf{x}_3\rangle + s^3 a_4\langle \mathbf{x}_4, \mathbf{x}_4\rangle + \\
+3c^3 a_3\langle \mathbf{x}_1, \mathbf{x}_3\rangle + 3cs^2 a_1\langle \mathbf{x}_1, \mathbf{x}_3\rangle \\
+3s^3 a_2\langle \mathbf{x}_2, \mathbf{x}_4\rangle + 3c^2 s a_4\langle \mathbf{x}_2, \mathbf{x}_4\rangle = \\
= c^3(a_1 A + 3a_3 B) + 3c^2 s(3a_2 B + a_4 B) \\
+3cs^2(3a_3 B + a_1 B) + s^3(a_4 A + 3a_2 B) = \\
= c^3 C_1 + 3c^2 s C_2 + 3cs^2 C_3 + s^3 C_4,
\end{aligned}$$

where the C_is are scalars depending on the parameters of the curvature derivative tensor (Equation 4.13).

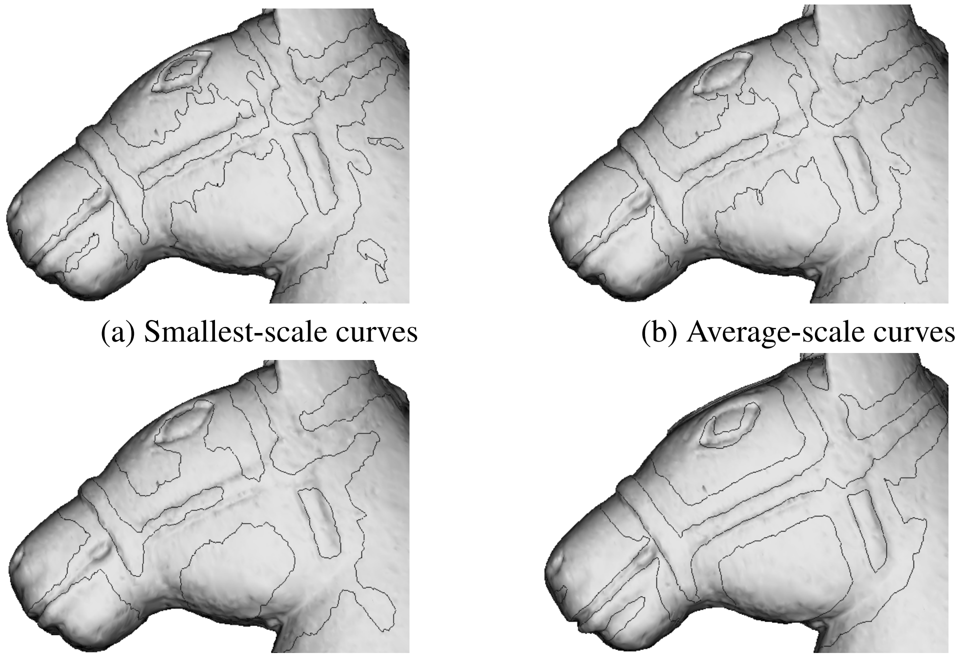
Chapter 5

Multi-scale curve detection on surfaces

Abstract

This paper extends to surfaces the multi-scale approach of edge detection on images. The common practice for detecting curves on surfaces requires the user to first select the scale of the features, apply an appropriate smoothing, and detect the edges on the smoothed surface. This approach suffers from two drawbacks. First, it relies on a hidden assumption that all the features on the surface are of the same scale. Second, manual user intervention is required. In this paper, we propose a general framework for automatically detecting the optimal scale for each point on the surface. We smooth the surface at each point according to this optimal scale and run the curve detection algorithm on the resulting surface. Our multi-scale algorithm solves the two disadvantages of the single-scale approach mentioned above. We demonstrate how to realize our approach on two commonly-used special cases: ridges & valleys and relief edges. In each case, the optimal scale is found in accordance with the mathematical definition of the curve.

This paper appeared in CVPR 2013.



(a) Smallest-scale curves (b) Average-scale curves

(c) Large-scale curve (d) Our multi-scale curves

Figure 5.1: **The benefit of using multi-scale curves.** When relief edges [72] are detected using a single scale, some features are missed and others are inaccurate (a)-(c). Conversely, when using multiple scales, the detected curves are more correct (d).

5.1 Introduction

3D feature curves on surfaces carry important information regarding the shape of the object. Therefore, a lot of effort has been devoted to charactering curves and detecting them. Examples of types of curves include ridges & valleys [98], parabolic curves [68], zero-mean curvature curves [68], demarcating curves [71], and relief edges [72], to name a few. Each type of curve is used to detect a different 3D feature. Curves on surfaces are equivalent to edges in images, which are basic low-level features in images. Consequently, 3D curves are inherently important in 3D shape analysis.

In images, each edge is associated with a scale. This scale is related to the image gradient; the steeper and stronger the edge, the smaller the scale. This is because steep edges are thinner, occupying a smaller area in the image than fuzzy moderate edges. This concept exists also in 3D curves on surfaces. For instance, the eye of the horse in Figure 5.1 has a smaller scale than its harness.

As illustrated in Figure 5.1, no single scale suffices to capture all the features. If the scale is too large, fine details are missed. On the other hand, if the scale is too small, coarse features are localized inaccurately and false features appear. However, most state-of-the-art curve detection algorithms use a single scale [24, 56, 71, 72, 98]. Moreover, the user is required to manually choose the "correct" scale.

Our goal is to propose a general framework for automatically estimating the optimal scale at each point on the surface. This general scheme can then be applied to every type of 3D curve, assuming it can be defined by the curvature and its derivatives. Hence, our technique not only eliminates the needed user intervention, but is also able to detect feaures of different scales on a single object.

A couple of algorithms address scale selection. Pauly et al. [102] propose a scheme that is designed for a single type of curves, defined as the loci of points whose curvature variation is persistent over all scales. It cannot be applied in a straightforward manner to other types of curves. Luo et al. [83] propose a method that is independent of the curve type. Rather, they essentially apply multi-scale smoothing to the object, hopefully leaving the surface features intact. However, the scale must be proportional not only to the surface features, but also to the curve type. When the latter is not taken into account, the extracted curve might be inaccurate. Moreover, their approach is quite slow (might take hours), whereas ours takes only a couple of minutes.

Our technique is inspired by the scale-selection theory developed by Lindeberg [80] for images (on which SIFT is based). We extend this theory to three dimensions. Given a definition of a curve type, we show how to calculate the optimal scale directly from the definition. Briefly, every curve type is associated with a function,whose value indicates the strength of the feature. This function typically depends on the surface curvature and its derivatives. We show how to select a parameter, in order for this function to have a single maximum over the range of scales. We define the scale at which the maximum is obtained as the optimal scale. Using this local optimal scale, the surface is smoothed in a multi-scale manner. On this surface, the curves are detected, utilizing the original curve detection algorithms.

We demonstrate the benefit of our approach by applying it to two popular types of curves: ridges & valleys and relief edges. We show that our curves outperform their counterparts computed with a manually-selected single scale, both in terms of accuracy and in terms of robustness to noise. This is especially evident in objects having features of several scales.

The paper is structured as follows. Section 5.2 provides the essential background. Section 5.3 formally defines the notion of optimal scale. Section 5.4 describes the algorithm for computing the optimal scale at every point. Section 5.5 applies the general method to two specific cases and demonstrates our results. We conclude in Section 5.6.

5.2 Background

This section describes the background on curve detection on surfaces and on multi-scale processing on surfaces.

Curves on surfaces: Curves on surfaces can be classified as view-dependent or view-independent. *View-dependent curves* depend not only on the differential geometric properties of the surface, but also on the viewing direction [24, 56, 67]. They change whenever the camera changes its position or orientation. *View-independent curves* depend solely on geometric properties of the surface [61, 68, 71, 72, 91, 98]. Our approach is general and applies to both categories of curves. For demonstration, we apply our approach to two types of view-independent curves. The first is *ridges & valleys* [98], which are the extrema of principal curvatures and the second are *relief edges* [71, 72], which are the loci of zero crossings of the curvature in the edge direction.

Mutli scale processing of surfaces: Multi-scale processing of surfaces can be divided into two separate, yet related, tasks. The first is the creation of *scale space*, which simultaneously represents the surface at different scales. The second is the *optimal scale selection*, which automatically selects the scale that best represents the surface locally.

1. Scale space representation: Scale space can be intuitively thought of as a collection of smoothed versions of the original surface. While in images, which are defined on a regular grid scale space, smoothing is a matter of consensus [80], for surfaces there are many ways to perform smoothing [47, 74, 96, 102, 103].

Formally, given a surface $S(u,v) : \mathbb{R}^2 \rightarrow \mathbb{R}^3$, its scale space representation is

$$S(u,v,t) : \mathbb{R}^2 \rightarrow \mathbb{R}^3,$$

where t is the scale parameter, which is proportional to the amount of smoothing applied to the object. In our work, we use the diffusion-based smoothing of [130], though other methods could also be used.

2. Optimal scale selection: In images, the optimal scale selection was developed for edge detection and point-based features by Lindeberg [80]. He proved that a function of the image spatial derivatives, which is normalized in a certain way, obtains a single maximum in scale space. The scale at which the maximum is obtained is termed the optimal scale.

Formally, let $g : \mathbb{R}^2 \rightarrow \mathbb{R}$ be an image, $L(\mathbf{x};t)$ be its scale space representation, and $\delta_{\mathbf{x}^k}$ be its k^{th} derivative. Then, the γ-normalized function of the derivatives of the image, defined as

$$t^{\gamma/2} f(\delta_{\mathbf{x}^k} L(x;t)) \tag{5.1}$$

obtains a single maximum at the optimal scale t_o. The function f and the k^{th} derivative depend on the desired feature.

Optimal scale for surfaces was mostly used for interest point detection. At these points, a function of some surface properties, normalized by the scale parameter, obtains a maximum both in the spatial and in the scale domains. These properties can be either normals [48], curvatures [108], or functions of curvatures [96]. These SIFT-like features are applied together to photometric and geometric features in [47].

A different approach, to which our method belongs, explicitly chooses a region on the surface, where the variation of some surface property is small. For instance, [102] compute the ratio between the eigenvalues; [83] compute the area with the minimal descriptor length; [75] compute the optimal size of the support region for computing surface normals.

5.3 Definition of the Optimal Scale

Intuitively, the optimal scale at point **p** is the scale at which the likelihood that a curve passes through **p** is maximal. Therefore, we consider the likelihood to be proportional to the curve strength. However, the more smoothing applied to the surface, the weaker the curves become. To compensate for this, the strength is normalized by a function of the scale. Hence, the optimal scale is reformulated as the scale at which the normalized curve strength obtains a maximum in scale space.

Definition: Let curve c have an associated strength function $f(K)$, which depends on the curvature and the curvature derivatives K. Let t be the scale parameter, which is related to the amount of smoothing applied to the surface. Let $f(K(t))$ be the result of applying f to the smoothed surface. Then, the optimal scale s_c of curve c is defined as the scale at which the function obtains a maximum:

$$s_c = \arg\max_t t^\gamma f(K(t)), \tag{5.2}$$

where t^γ is the normalization coefficient, similar to what is used in Equation (5.1). We denote the expression $t^\gamma f(K(t))$ as the *normalized function*.

The strength function $f(K(t))$ monotonically decreases with scale, since as the surface becomes smoother, the features appear less prominent. Conversely, t^γ monotonically increases with scale. We should therefore choose γ correctly, so as to ensure a single maximum of the normalized function $t^\gamma f(K(t))$ in Equation (5.2). This single-maximum property is proved in Section 5.5 for two commonly-used spatial curve types: ridges & valleys and relief edges.

To get a feeling why this is true, recall that a surface can be presented locally, at every point on the surface, as a third degree polynomial (the *Monge form*) defined on the point's tangent plane. The surface curvature is proportional to the polynomial second derivative. According to Equation (5.2), the normalized second derivative obtains a single maximum in the scale space. Therefore, the surface curvature should also obtain a maximum in scale space.

Note that in featureless areas, the feature strength is zero for all scales. Thus, the optimal scale (the maximum) is undefined there. However, since there exist no curves in a featureless region and our goal is to detect curves, any scale chosen is valid and our definition holds.

5.4 Multi-Scale Curve Detection

The goal of this section is to describe an algorithm that detects the curves, given the definition of the optimal scale, described above. Our algorithm consists of three steps. First, we compute the optimal scale at every point on the surface. This computation depends on the type of curve we want to detect, and specifically on its corresponding strength function f. Second, we create a new surface, where each point is smoothed according its optimal scale. Obviously, this step does not depend on the type of curves. On this smoothed surface, the curves are finally computed using their original detection algorithms. This is in contrast to the regular curve detection algorithms, which perform a uniform smoothing on the whole surface, prior to detection.

Computing the optimal scale at every point: For points with features, we apply Equation (5.2) as is and obtain the optimal scale. For featureless points, any value of scale is valid, and yet we need to choose a single scale. Since our goal is to produce the optimally-scaled surface, the scale should be a continuous function over the surface. We therefore require that the following conditions hold:

1. The scale at a point with high strength is equal to the solution of Equation (5.2).

2. The scale changes smoothly along the surface.

Formally, the first condition is a simple equality constraint. The second condition is realized by requiring that the weighted Laplacian of the scalar scale function, defined on the surface, is zero. These two conditions are combined into a system of linear equations, as follows.

Let $\mathbf{p}$ be a point on the surface and $t_i(\mathbf{p})$ be the scale that solves Requirement 1. The final scale $t(\mathbf{p})$ is found by solving the following system:

$$\begin{aligned} t(\mathbf{p}) &= t_i(\mathbf{p}) \\ w(\mathbf{p})\Delta t(\mathbf{p}) &= 0, \end{aligned} \tag{5.3}$$

where $w(\mathbf{p})$ is a weight function inversely proportional to the strength at $\mathbf{p}$ and Δ is the Laplacian. We define $w(\mathbf{p})$ as $1/(1+af)$, where f is the strength function value at the point and a is chosen so that $af_{\max} = 9$.

In our work, we assume that the surface is represented by a triangular mesh. The Laplacian Δ of a scalar function t at point $\mathbf{p}$ on a mesh is calculated as in [89]:

$$\Delta t(\mathbf{p}) = \frac{1}{2A} \sum_{j \in N(\mathbf{p})} (\cot(\gamma_j) + \cot(\delta_j))(t(\mathbf{p}) - t(\mathbf{p}_j)), \tag{5.4}$$

where $N(\mathbf{p})$ is the set of neighbors of $\mathbf{p}$ on the mesh, A is the Voronoi area of $\mathbf{p}$, and γ_j and δ_j are the angles opposite to $\mathbf{pp}_j$ (see Figure 5.2). We solve the system of equations (5.3) with an SVD-based solver [35].

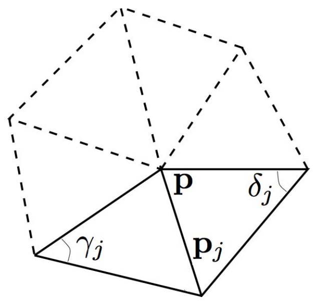


Figure 5.2: **Notations of Equation** (5.4)**.** The Laplacian Δ of a scalar function t at point $\mathbf{p}$ on a mesh is a linear combination of the values of t on the neighbors $\mathbf{p}_j$ of $\mathbf{p}$.

Creating an optimally-smoothed surface: After the optimal scale at each point has been computed, we create a surface in which each point is smoothed to its optimal scale. This is done as follows. First, the surface is smoothed uniformly for each scale, using [130]. For each of these surfaces, we maintain the locations of its vertices. Next, we create a surface for which the coordinates of every vertex are taken from the surface of its corresponding scale. Finally, the resulting surface is smoothed in order to remove artifacts.

5.5 Specific Cases

In this section we demonstrate how to apply our method to commonly-used curves: ridges & valleys and relief edges. We show how to choose the strength function f and the normalization coefficient γ from Equation (5.2), which are used in the first step of our algorithm (Section 5.4).

Recall that the normalization coefficient γ should be chosen such that to ensure a single maximum of the normalized strength function. In order to simplify the task, we make three assumptions. First, we make the standard assumption that the surface in the curve's neighborhood is a function defined on the tangent plane, i.e. the surface is defined by $s(x,y)$. Second, we assume that $s(x,y)$ is constant along the curve, i.e. the surface can be modeled locally by a 1D function. For example, if the curve has direction y, then $s(x,y) \equiv s(x)$. In other words, we assume that $s(x,y)$ is locally developable in the vicinity of the curve. Third, we make the commonly-used assumption that the smoothing process can be approximated by convolving the function with a Gaussian. Our experiments show that this approximation is sufficiently accurate.

Having made these assumptions, we perform the following general steps for finding γ. For each curve type we need to realize these steps differently.

1. Choose the strength function.
2. Choose a 1D function $s(x)$ that represents the surface locally.

3. Derive the expression for the scale at which the normalized strength obtains a maximum. This maximum is the optimal scale.
4. Choose γ values for which the expression in step 3 is maximized by a single scale value.

5.5.1 Ridges & valleys

The most prevalent curves on surfaces are ridges & valleys [98]. They are similar to their geographical counterparts and usually indicate sharp changes in the surface orientation. A ridge (valley) point is a point on a manifold, where the positive (negative) principal curvature obtains a maximum (minimum) along its principal direction. We now discuss the four steps mentioned above, for the case of ridges, whereas the case of valleys is similar.

Strength function f: The strength function f we use is the maximal value of the curvature. This is the standard method to measure the strength of ridges.

Surface representation: We approximate a ridge $s(x;t_0)$ of scale t_0 with a Gaussian of standard deviation $\sigma = \sqrt{t_0}$:

$$s(x;t_0) = \frac{1}{\sqrt{2\pi t_0}} e^{-x^2/2t_0}.$$

The ridge point is obtained at $x = 0$.

We assume that the scale space is represented by convolutions with Gaussians with smoothing parameter t:

$$s(x;t+t_0) = s(x;t_0) * g(x;t). \tag{5.5}$$

Thus, the ridge $s(x;t_0)$ at scale t is

$$s(x;t+t_0) = \frac{1}{\sqrt{2\pi(t_0+t)}} e^{-x^2/2(t_0+t)}.$$

Optimal scale: We want to show that the normalized curvature of $s(x;t_0)$ obtains a single maximum in scale space at $x = 0$. First, we express the normalized curvature at $x = 0$ as a function of t and γ:

$$t^{\gamma}\kappa(x;t_0+t). \tag{5.6}$$

Then, we take its derivative with respect to t and prove that this derivative is equal to zero only at a single t. Let us start with finding the expression for $\kappa(x;t_0+t)$. The curvature $\kappa(x)$ of a curve $s(x)$ is known to be:

$$\kappa(x) = -\frac{s''(x)}{(1+s'(x)^2)^{3/2}}, \tag{5.7}$$

where the derivatives are with respect to x.

We need to compute the derivatives of the surface (curve) $s'(x)$ and $s''(x)$. It is easy to see that the first derivative is:

$$s'(x;t_0+t) = \frac{-x}{t_0+t}\cdot\frac{e^{-x^2/2(t_0+t)}}{\sqrt{2\pi(t_0+t)}}$$

and the second derivative is:

$$s''(x;t_0+t) = \left(\frac{-1}{t_0+t}+\frac{x^2}{(t_0+t)^2}\right)\cdot\frac{e^{-x^2/2(t_0+t)}}{\sqrt{2\pi(t_0+t)}}.$$

Substituting $x = 0$, we obtain

$$s'(0;t_0+t) = 0, \tag{5.8}$$

$$s''(x;t_0+t) = \frac{-1}{t_0+t}\cdot\frac{1}{\sqrt{2\pi(t_0+t)}} = \frac{-1}{\sqrt{2\pi}(t_0+t)^{3/2}}. \tag{5.9}$$

Therefore, by substituting Equations (5.8)-(5.9) into Equations (5.7)-(5.6), we get that the value of the strength function at $x = 0$ for different scales t and different normalization coefficients γ is:

$$t^{\gamma}\kappa(x;t_0+t)\Big|_{x=0} = \frac{t^{\gamma}}{\sqrt{2\pi}(t_0+t)^{3/2}}. \tag{5.10}$$

In order to check when Equation (5.10) obtains a maximum, we set its derivative with respect to t to 0:

$$\frac{d}{dt} t^{\gamma} \kappa(x; t_0 + t) \bigg|_{x=0} = 0, \quad \text{or}$$

$$\frac{\gamma t^{\gamma-1}}{\sqrt{2\pi}(t_0+t)^{3/2}} - \frac{3t^{\gamma}}{2\sqrt{2\pi}(t_0+t)^{5/2}} = 0.$$

After some simple algebraic manipulations, we obtain an equation for the optimal scale:

$$t_{\max} = 2t_0/(3 - 2\gamma). \tag{5.11}$$

Choice of γ: We conclude that when $\gamma < 1.5$, the optimal scale is unique. In our experiments, we use $\gamma = 1$. We observed that small changes of γ do not influence the results.

Results: Figure 5.3 shows ridges and valleys on objects consisting of features of different scales. When the chosen scale is small (b), the results are noisy. When we increase the scale (c), the curves become smoother, but some features disappear. We were unable to find a single scale that yields a good trade-off between smoothness and detectability. Conversely, with our multi-scale approach (d), the curves are smooth and even small details are nicely captured. See the supplementary for additional results.

To provide quantitative evaluation, we created a synthetic example, for which we can calculate the ground truth. It is a cylinder with ridges and valleys of a couple of different scales, to which we added noise, as shown in Figure 5.4. We ran the single-scale algorithm using various single scales and our multi-scale algorithm. When the (single) scale is small, the coarse features are not detected accurately. When it is large, the fine features disappear. With our multi-scale approach, the features are nicely captured. In terms of error, we computed the percentage of the ground-truth valleys for which there exist real valleys within a predefined distance. In practice, we counted the number of faces with ground truth valleys for which there exists a face with a detected valley within 0.5% of the cylinder radius. While the accuracy of the multi-scale results is 91%, for the single scale algorithm the results are between 79% for the finest scale to 42% for the coarsest, The results are: Scale 0

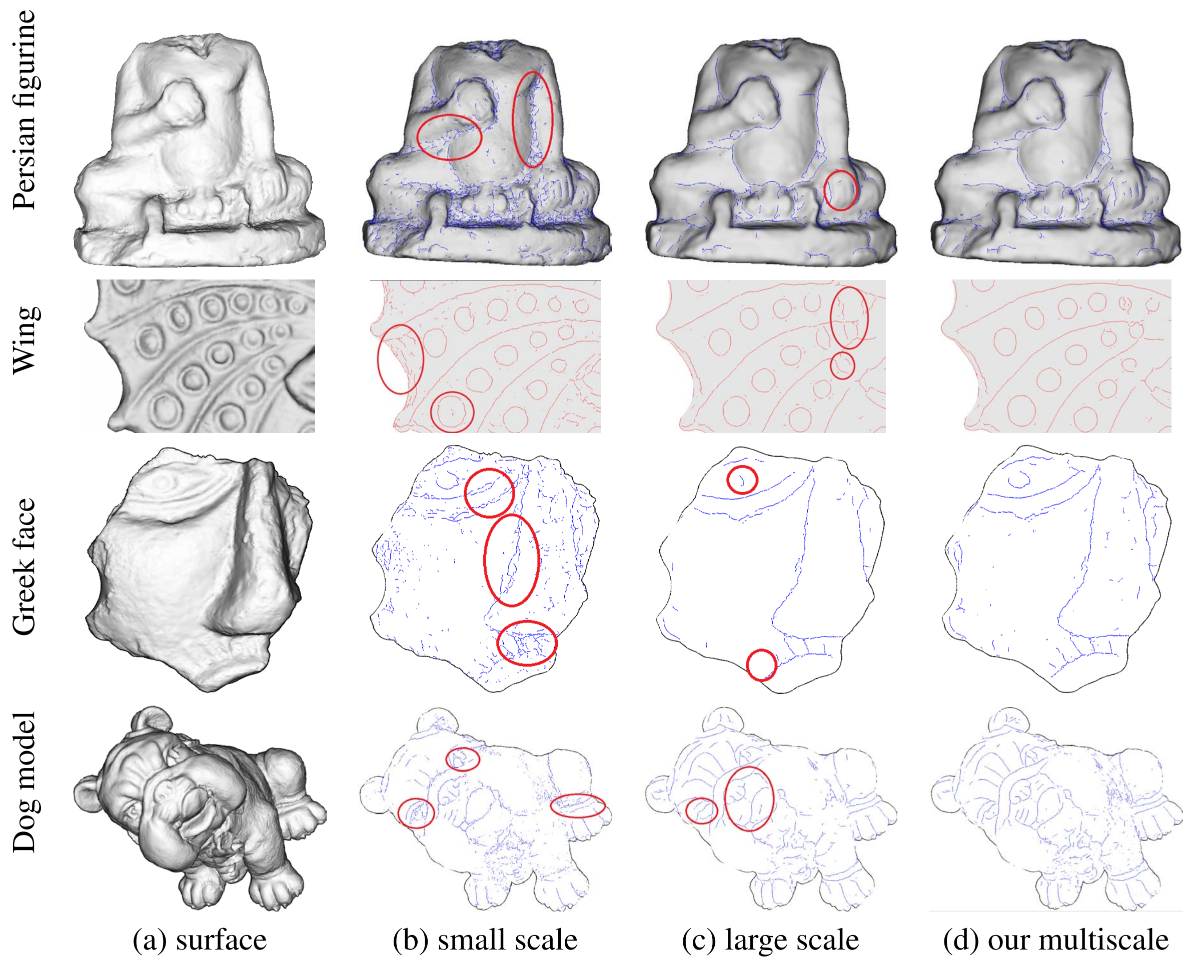
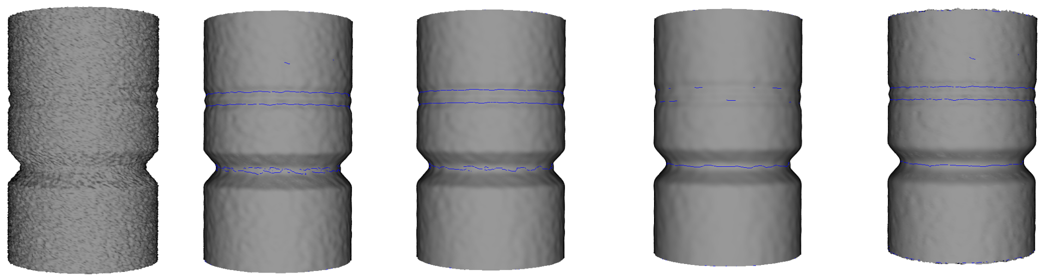


Figure 5.3: **Ridges and valleys on objects with various scales.** When the chosen scale is small (b), the results are noisy. When the scale is large (c), some features disappear. With our multi-scale approach (d), all the features are nicely captured and the curves are smooth. The red circles indicate problematic areas.

(finest): 79%, Scale 1: 85%, Scale 2: 77%, Scale 3 (coarsest): 42%, and Multiscale: 91%.

5.5.2 Relief edges

Relief edges are defined as zero crossings of the curvature in the direction of the step edge model that best approximates the surface locally [72]. They run on the slopes between ridges and valleys and are parallel to them.



(a) Input	(b) fine scale	(c) coarse scale	(d) coarsest scale	(e) multi scale
Accuracy:	85%	77%	42%	91%

Figure 5.4: **Quantitative evaluation.** The noisy cylinder has both fine and coarse valleys (a). The results of our multi-scale approach (e) outperforms those of the single-scale approach (b-d). The accuracy measure is given underneath the images.

Strength function f: As a strength function f, we employ the curvature derivative with respect to the arclength λ in the edge direction, as proposed in [72]. Thus,

$$f = \frac{\partial \kappa(x;t_0)}{\partial \lambda}.$$

Surface representation: A relief edge is represented, by definition, by a smoothed step edge. Let $s(x;t_0)$ be a relief edge of an initial scale t_0 (an ideal step edge smoothed with a Gaussian of standard deviation $\sigma_0 = \sqrt{t_0}$):

$$s(x;t_0) = \frac{1}{\sqrt{2\pi t_0}} \int_{-\infty}^{x} e^{-u^2/2t_0} du.$$

Optimal scale: The proof is similar to that in Section 5.5.1. We want to show that the normalized curvature derivative of $s(x;t_0)$ obtains a single maximum in scale space at $x = 0$. First, we express the normalized curvature at $x = 0$ as a function of t and γ:

$$t^\gamma f = t^\gamma \frac{\partial \kappa(x;t+t_0)}{\partial \lambda}. \tag{5.12}$$

Then, we take its derivative with respect to t and prove that it is equal to zero only at a single t. The curvature derivative is:

$$\frac{\partial \kappa}{\partial \lambda} = \frac{\partial \kappa}{\partial x} \cdot \frac{\partial x}{\partial \lambda}. \tag{5.13}$$

We now show how to compute $\partial k/\partial x$ and $\partial k/\partial \lambda$ and then combine them. To compute $\partial k/\partial x$, we take the derivative of the curvature defined in Equation (5.7). The scale space is generated using a convolution with a Gaussian (Equation (5.5)):

$$s(x;t+t_0) = \frac{1}{\sqrt{2\pi(t_0+t)}} \int_{-\infty}^{x} e^{-u^2/2t_0+t} du.$$

Then, the derivative of $s(x)$ with respect to x is

$$s'(x;t_0+t) = \frac{e^{-x^2/2(t_0+t)}}{\sqrt{2\pi(t_0+t)}}$$

and the second derivative is

$$s''(x;t_0+t) = \frac{-x}{t_0+t} \cdot \frac{e^{-x^2/2(t_0+t)}}{\sqrt{2\pi(t_0+t)}}.$$

We now proceed to compute the curvature $\kappa(x)$

$$\kappa(x) = \frac{-xe^{-x^2/2(t_0+t)}}{(t_0+t)^{3/2}\sqrt{2\pi}} \cdot \frac{1}{[1+e^{-x^2/(t_0+t)}/(2\pi(t_0+t))]^{3/2}}$$

and the curvature derivative

$$\begin{aligned} \left.\frac{\partial k}{\partial x}\right|_{x=0} &= \left. \frac{-e^{-\frac{x^2}{2(t_0+t)}}}{(t_0+t)\sqrt{2\pi(t_0+t)}} \cdot \frac{1}{\left[1+\frac{e^{-\frac{x^2}{t_0+t}}}{2\pi(t_0+t)}\right]^{3/2}} \right|_{x=0} \\ &= \frac{2\pi}{(1+2\pi(t_0+t))^{3/2}}. \end{aligned}$$

After having computed $\partial\kappa/\partial x$ of Equation (5.13), we now derive $\partial x/\partial\lambda$:

$$\partial\lambda = \sqrt{\partial x^2 + \partial y^2}, \frac{\partial\lambda}{\partial x} = \sqrt{1+s'(x)^2},$$

$$\frac{\partial x}{\partial\lambda} = \frac{1}{\sqrt{1+s'(x)^2}}.$$

Equation (5.13) now becomes

$$\frac{\partial\kappa}{\partial\lambda} = \frac{2\pi}{(1+2\pi(t_0+t))^{3/2}} \cdot \frac{1}{\sqrt{1+s'(x)^2}} =$$

$$\frac{2\pi}{(1+2\pi(t_0+t))^{3/2}} \cdot \frac{\sqrt{2\pi(t_0+t)}}{\sqrt{1+2\pi(t_0+t)}} = \frac{4\pi^2\sqrt{t_0+t}}{(1+2\pi(t_0+t))^2}.$$

In turn, Equation (5.12) now becomes

$$t^\gamma \frac{\partial\kappa(x;t+t_0)}{\partial\lambda} = t^\gamma \frac{4\pi^2\sqrt{t+t_0}}{(1+2\pi(t+t_0))^2}. \tag{5.14}$$

Equation (5.14) specifies the value of the strength function at $x=0$ for different scales t and different normalization coefficients γ. We next find when it obtains a maximum. Taking the derivative with respect to t and setting it to zero, we obtain a quadratic equation whose roots are:

$$t_{\max} = (A \pm B)/C, \tag{5.15}$$

where

$$\begin{aligned} A &= 2\gamma - 6\pi t_0 + 8\pi\gamma t_0 + 1, \\ B &= (4\gamma^2 + 40\pi\gamma t_0 + 4\gamma + 36\pi^2 t_0^2 - 12\pi t_0 + 1)^{1/2}, \\ C &= 4(3\pi - 2\pi\gamma). \end{aligned}$$

The derivation of Equation (5.15) is given in Appendix A.

Choice of γ: We show in Appendix A that $(A-B)/C$ is always negative and thus irrelevant, whereas $(A+B)/C$ is always positive for $\gamma < 1.3$. Hence, for $\gamma < 1.3$ the optimal scale is unique.

Results: Figure 5.5 depicts relief edges on surfaces having features of various scales. It compares two single-scale relief edges with our multi-scale edges. The results of the large scale were found to be the best over all scales. Even though the large single-scale result of the Iron Age stamp is pretty, the head is portrayed inaccurately. This is so, since the depth of the relief varies, thus a single scale is insufficient. On the other hand, our multi-scale approach detects the edges accurately. On the bottom row, our multi-scale curves are much smoother than the best single-scale result, when tested on the arm of the figurine from Figure 5.3. To recap, our results are more accurate and smoother than those produced by the single-scale approach.

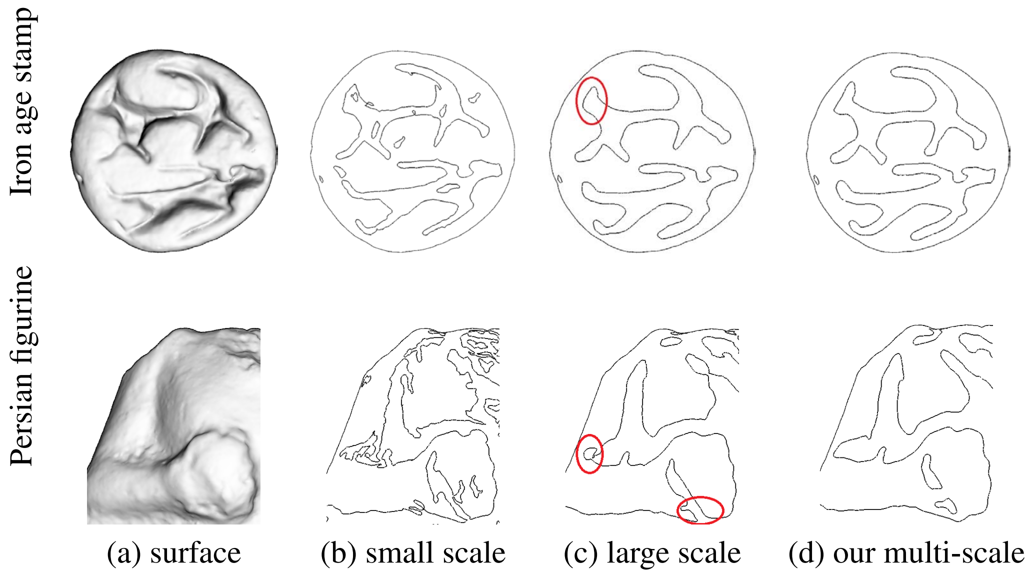


Figure 5.5: **Relief edges on objects having various scales.** When the scale is small (b), the resulting single scale curves are noisy. When the scale is big (c), the single scale curves are smooth but not accurate enough. For example, they do not capture the head of the dog in the top row and create topological mistakes in the arm on the bottom row. Our multi-scale relief edges are more accurate.

Additional benefit of our approach: In addition to the superiority of our approach in cases where the objects have a variety of scales, it is also beneficial in the case when objects have a single scale. While in all the single-scale approaches, the user needs to manually select the scale parameter, in our approach no manual tuning is necessary.

5.6 Conclusion

This paper presented a framework for automatic estimation of the optimal scale for curve detection on surfaces. It can be applied to any curve type, as long as the curve has a strength function based on the curvature and its derivatives. This requirement is satisfied by most curve types.

Our experiments show that on objects composed of features of various scales, the curves obtained by our method outperform those computed by the original single-scale algorithms. On objects that consist primarily of features of a single scale, the benefit of our algorithm is that it does not require manual parameter tuning.

Acknowledgements: This research was supported in part by the Israel Science Foundation (ISF) 1420/12 and the Ollendorff Foundation.

Appendix A: Maxima of normalized curvature derivative of a step edge

Here, we find the values of t at which Equation (5.14) obtains maximum. These are the values at which the derivative of Equation (5.14) is equal to zero.

$$\frac{d}{dt}\frac{t^{\gamma}\sqrt{t+t_0}}{(1+2\pi(t+t_0))^2}=0$$

$$\frac{\gamma t^{\gamma-1}(t+t_0)^{1/2}}{(1+2\pi(t+t_0))^2}+\frac{t^{\gamma}}{2(t+t_0)^{1/2}[(1+2\pi(t+t_0))^2]}-\frac{2\pi 2t^{\gamma}(t+t_0)^{1/2}}{[(1+2\pi(t+t_0))^3]}=0.$$

After several simple manipulations:

$$\frac{\gamma(t+t_0)}{1}+\frac{t}{2}-\frac{2\pi 2t(t+t_0)}{[(1+2\pi(t+t_0))]}=0, \quad \text{or}$$

$$2\pi(2\gamma-3)t^2+(2\gamma-6\pi t_0+8\pi\gamma t_0+1)t+2\gamma t_0(2\pi t_0+1)=0.$$

This is a quadratic equation whose roots t_i are

$$t_i=[2\gamma-6\pi t_0\pm(4\gamma^2+40\pi\gamma t_0+4\gamma+36\pi^2 t_0^2-12\pi t_0+1)^{1/2}+8\pi\gamma t_0+1]/(4(3\pi-2\pi\gamma)).$$

A positive solution to this equation always exists for
$0<\gamma<1.3$.

Chapter 6

Prominent field for shape processing and analysis

Abstract

Archaeological artifacts are an essential element of archaeological research. They provide evidence of the past and enable archaeologists to obtain qualified conclusion. Nowadays, many artifacts are scanned by 3D scanners. While convenient in many aspects, the 3D representation is often unsuitable for further analysis, due to flaws in the scanning process or defects in the original artifacts. We propose a new approach for automatic processing of scanned artifacts. It is based on the definition of a new direction field on surfaces (a normalized vector field), termed the *prominent field*. The prominent field is oriented with respect to the prominent feature curves of the surface. We demonstrate the applicability of the prominent field in two applications. The first is surface enhancement of archaeological artifacts, which helps enhance eroded features and remove scanning noise. The second is artificial coloring that can replace manual artifact illustration in archaeological reports.

The conference version of this paper appeared in ICCV Workshop, 2009 and the journal version was published in IJCV, 2011. The full name is *Prominent field for shape processing and analysis of archaeological artifacts*.

6.1 Introduction

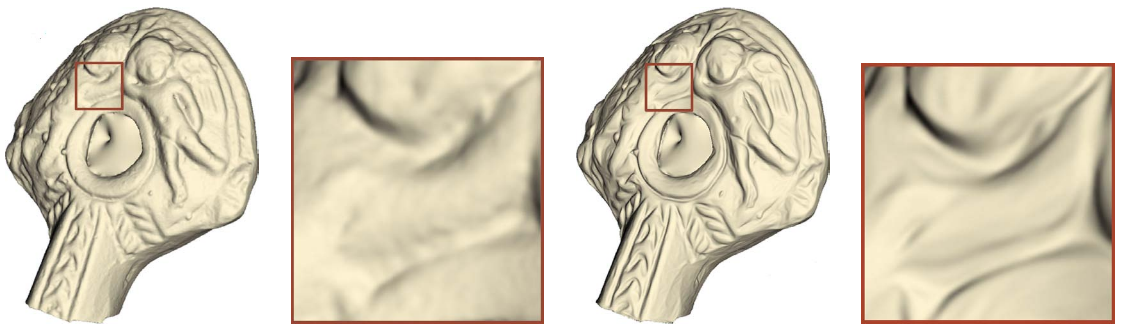


Figure 6.1: **Enhancement of a late Hellenistic oil lamp from the first century BCE.** The red rectangle depicts the zoomed-in part.

Man-made artifacts are a major source of our knowledge about the past. Archaeologists who study assemblages of artifacts seek to identify distinctive patterns in them, which can be used for analysis and comparison. In order to disseminate the information about artifacts, these are either illustrated in reports or scanned by 3D scanners. Nowadays, 3D representations are becoming more popular, since they provide more information by allowing the archaeologist to view the artifact from different viewpoints at various scales and perform comparative measurements. Therefore, this paper focuses on 3D representation of artifacts. Specifically, we concentrate on artifacts with reliefs, which consist of a detailed surface, the *relief*, that resides on the top of a smooth base surface.

The main task facing the archaeologists is the analysis of these artifacts. This is done in several ways – accurately illustrating them by highlighting the surface edges, comparing artifact styles, classifying them etc. All these tasks can be facilitated by applying computer vision and computer graphics techniques [12, 69, 110, 138]. However, the 3D representation is often flawed, either due to defects in the original artifacts or due to the erosion the artifact underwent after spending two thousand years underground. Noise added in the scanning process may also damage the representation. Figure 6.1(a) gives an example of a flaw. The surface of the original object is slightly rippled and looks blurred.

This paper addresses this problem by proposing a novel framework for processing the artifacts. It is based on a definition of a new direction field (a normalized vector field), termed the *prominent field*, defined for every point on the surface. This field is constructed in a manner in which it is smooth on the surface. Intuitively, the direction of this prominent field, termed the *prominent direction*, is perpendicular to the surface's feature curves. With respect to the reliefs, we show that the prominent field is superior to previously proposed vector fields. The existing fields are oriented mostly according to the principal directions, which do not always coincide with the feature curves of the surface.

Since the prominent field is closely related to the feature curves of the relief surface, it is beneficial for a variety of processing applications. We demonstrate its effectiveness in two applications: surface enhancement and artificial coloring. The goal of adaptive filtering is to enhance the features while keeping the surface intact. This may also help to remove the scanning noise. We propose to smooth the surface using the prominent field along the feature curves and enhance it in the prominent direction. Figure 6.1(b) shows the effect of the adaptive filtering. The filtered object is smoother and has crisper, more visible details than the original object.

As a second application, we present a method for artificially coloring objects. The key idea is to color the surface according to its normal curvature in the prominent direction. This coloring increases the color contrast on the feature curves, thus enhancing them.

The contribution of this paper is hence threefold:

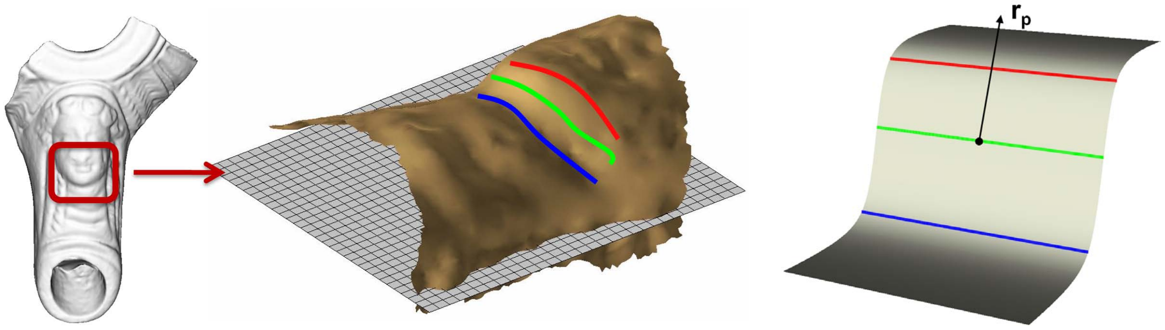
- We define the prominent field and show how to compute it in interactive time.
- We show how to employ the prominent field for surface smoothing and enhancement.
- We propose a method for artificial surface coloring that emphasizes the object features.

The paper continues as follows. Section 6.2 presents the required background. Section 6.3 defines the prominent field and shows how to compute it. Sections 6.4 and 6.5 demonstrate the applications. Finally, Section 6.6 concludes the paper. A preliminary version of this work appeared in [73].

6.2 Background

This section presents the essential background on relief surfaces and on vector fields on surfaces.

Relief surfaces and feature curves on them: A relief surface can be viewed locally as a terrain. Like any other terrain, it has valleys and ridges. Three types of feature curves are defined on it: ridges, valleys, and relief edges, as illustrated in Figure 6.2(a). All these curves run on the slopes of the terrain, which can be approximated locally by the step edge model (Figure 6.2(b)).



(a) Local terrain (b) Step edge model

Figure 6.2: **Three types of feature curves on a relief surface.** (a) The Hellenistic oil lamp can be viewed locally as terrain. The terrain has ridges (red), valleys (blue), and relief edges (green). (b) The step edge model can approximate the slopes of the terrain.

Ridges and valleys are similar to their geographical counterparts and usually indicate sharp changes in the surface orientation. Ridges (valleys) are defined as the maximum (minimum) of the normal curvature in the first principal direction [30].

On ideal step edges, relief edges run on the slopes between ridges and valleys and are parallel to them. They are shown to correspond to the image edges of the local image

I [72]. Equivalently, they are defined as zero crossings of the curvature in the direction of the step edge model that best approximates the surface locally in the L_2 norm. The direction is termed the edge direction.

Figure 6.3 compares different types of curves. Ridges are erroneous for this model; valleys make the dancers wider and do not always follow precisely the figures outline; relief edges provide more accurate results.

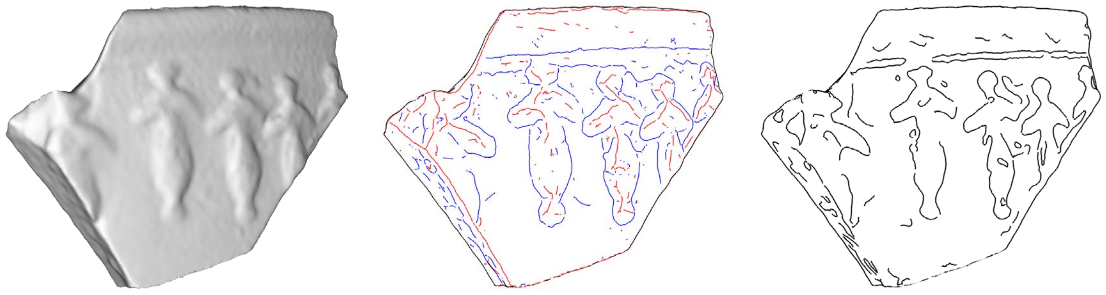


Figure 6.3: **Three types of feature curves on a Hellenistic vase depicting five dancers.** Ridges (red) & valleys (blue) do not follow precisely the dancers, whereas the relief edges (black) are more accurate.

Vector fields on surfaces: A vector field is a vector asociated with every point on the surface. Vector fields on surfaces are essential for many graphics applications, such as texture synthesis [141, 135], non photo-realistic rendering [45], fluid simulation [120], shape deformation [137] and others. There exists a variety of papers on editing, generation, manipulation, and filtering of vector fields [148, 33]. Since the magnitude of the vector field is irrelevant for our work, we will restrict the discussion to direction fields.

The most common candidates for direction fields are the principal directions, which correspond to the directions of the maximal and the minimal curvatures. The principal directions are reliable near ridges and valleys, i.e. at locations where the ratio of the principal curvatures is high, but are ill-defined near umbilical points. Therefore, typically the field is computed so as to coincide with the reliable principal directions and be smooth at other points ([45, 107]). While the principal directions are useful for general objects, they are less suitable for objects with reliefs. This is so since the relief details are often close to umbilical and the principal directions might fail to capture their orientation correctly. Figure 6.4(b) shows that while the principal directions are oriented well near the ridges and the valleys, they are noisy at other places.

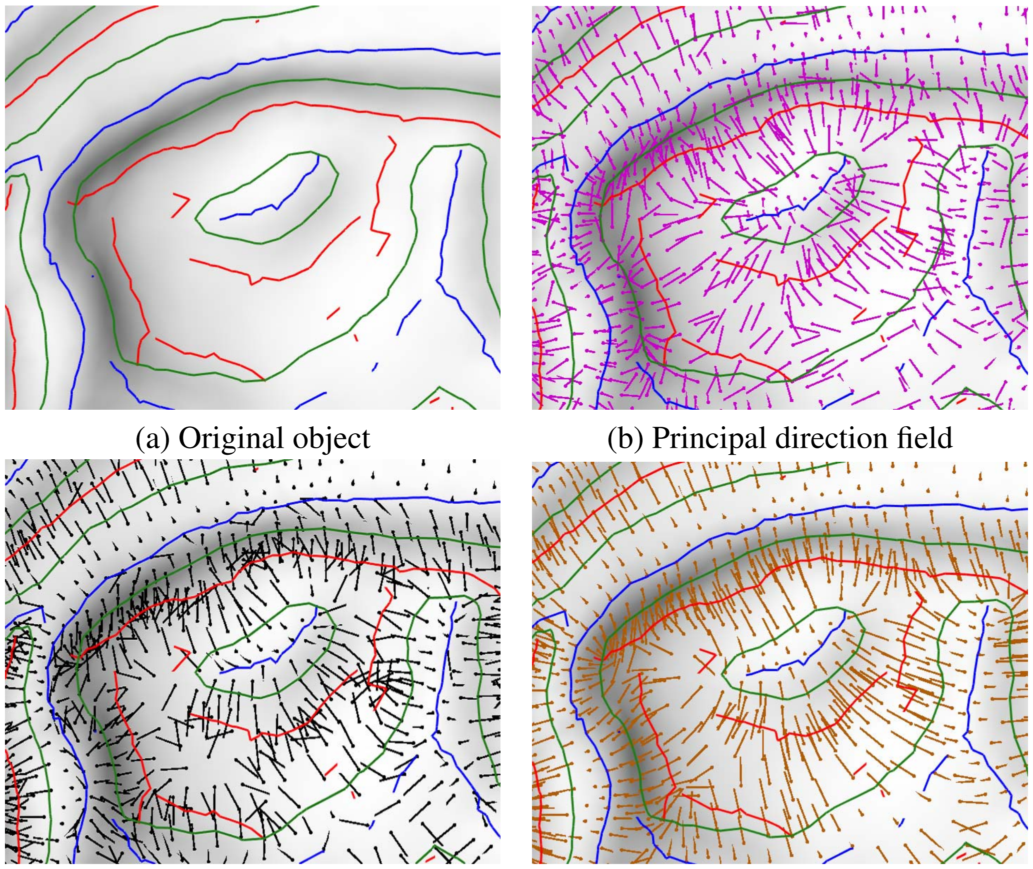

(a) Original object (b) Principal direction field
(c) Relief direction field (d) Prominent direction field

Figure 6.4: **Different direction fields.** In (a) the ridges (red), valleys (blue), and relief edges (green) are depicted. The principal direction field (magenta) is oriented well near the ridges and the valleys, but noisy at other places (b). The relief direction (black) is oriented well on the relief edges, but not at other parts of the surface (c). Our prominent direction (orange) is oriented well everywhere on the surface.

The directions perpendicular to demarcating curves [71] and to relief edges [72] are more appropriate for relief surfaces, since these curves are designed to illustrate relief features. The direction perpendicular to demarcating curves is the direction of the curvature gradient (the direction of the maximal curvature derivative). The direction perpendicular to relief edges is the direction of the locally best-fitting step edge. These directions are well defined on the relief edges, but are meaningless at the other parts of the surface, and thus cannot be used in a straightforward manner. Figure 6.4(c) shows that the relief directions are well oriented near the relief edges, but are randomly oriented at other locations.

6.3 Prominent field

This section presents a novel direction field – the *prominent field*. Intuitively, the prominent field is defined as the smoothest field perpendicular to the object features. A field satisfying this definition can enable us for example to enhance features in the direction of the field, while removing noise in the perpendicular direction.

Below we first present the surface model. Next we define the prominent field, first on the feature curves and then on the whole surface. Finally, we describe the algorithm for computing the prominent field on polygonal meshes.

Surface model: We define a relief surface as a surface composed of a smooth, low-frequency *base* and a high-frequency height function [72]. This function represents the signed distance between the base to the surface in the direction of the base's unit normal. (See Figure 6.5.) This decoupling of the surface into the base and the height function is unknown.

Formally, given a surface $S(u,v):\mathbb{R}^2\to\mathbb{R}^3$, we assume that it consists of a smooth base $B(u,v):\mathbb{R}^2\to\mathbb{R}^3$ and a height function $I(u,v):\mathbb{R}^2\to\mathbb{R}$ defined on B:

$$S(u,v)=B(u,v)+\bar{\mathbf{n}}(u,v)I(u,v), \tag{6.1}$$

where u and v are the coordinates of a parameterization and $\bar{\mathbf{n}}(u,v):\mathbb{R}^2\to\mathbb{S}^2$ is the normal of B ($\mathbb{S}^2$ being the unit sphere). We assume that B is locally a manifold and that its curvature has a smaller value than the curvature of I.

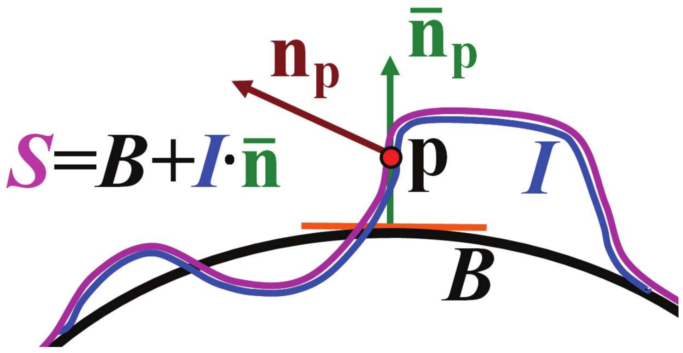


Figure 6.5: **Surface representation.** The surface S (magenta) is composed of a smooth base B (black) and a function I (blue). Function I at point $\mathbf{p}$ can be locally viewed as an image defined on the tangent plane (orange) of the base. Point $\mathbf{p}$ is a relief edge point if it is an edge point of this image. The normal $\mathbf{n_p}$ (brown) is the normal of S and $\bar{\mathbf{n}}_\mathbf{p}$ (green) is the normal of B corresponding to $\mathbf{p}$.

The prominent field on features: We are seeking a smooth direction field – *the prominent field* – that is perpendicular to the object feature curves. This direction is important since many processing tasks are closely related to the feature direction (which should be maintained, enhanced, etc.). Smoothness is required in order to produce smooth results.

Had the surface be an ideal step edge, this direction would be perpendicular to the ridge, valley, and relief edge (Figure 6.2(b)). In practice however, surfaces are not ideal step edges (Figure 6.2(a)) and thus we need to define a direction field more carefully.

To address this problem, we divide the surface points near the feature curves into two fuzzy classes. The first class consists of the points residing near the ridges and the valleys. Here the prominent direction should be equal to the first (maximum) principal direction. The second class includes points residing near the relief edges. In this case the prominent direction should be equal to the relief direction. The classification is fuzzy and hence the regions may overlap.

Formally, let $\mathbf{p}$ be a point on the surface, $\mathbf{g_p}$ be the relief direction, and $\mathbf{t_p}$ be the first principal direction. As mentioned above, on the step edge model, $\mathbf{g_p}$ is meaningful only on the relief edge and $\mathbf{t_p}$ is well-defined near ridges and valleys. Therefore, to define the prominent direction $\mathbf{r_p}$, $\mathbf{g_p}$ and $\mathbf{t_p}$ are combined as a weighted combination, where the weights are proportional to the likelihood of the point to be near a relief edge. Therefore, the weight $\alpha_{\mathbf{p}}$ is 1 at relief edges and 0 at ridge/valley points.

Definition 6.3.1. *The prominent direction is*

$$\mathbf{r}_{\mathbf{p}} = \alpha_{\mathbf{p}}\mathbf{g}_{\mathbf{p}} + (1-\alpha_{\mathbf{p}})\mathbf{t}_{\mathbf{p}},$$

where $\alpha_{\mathbf{p}} \in [0,1]$ is a scalar weight that determines the relative distance of **p** *from the relief edge.*

The question is how to define $\alpha_{\mathbf{p}}$. Let κ_1 and κ_2 be the principal curvatures, k their ratio, and l the median length of mesh edges. Empirically, we observed that good results are obtained when

$$\alpha_{\mathbf{p}} = \begin{cases} 1 & \max(|\kappa_1|,|\kappa_2|) < 3/l \text{ or } |k| < 2 \\ 0 & |k| > 4 \\ \frac{4-|k|}{2} & \text{otherwise.} \end{cases} \tag{6.2}$$

The intuition behind this definition is as follows. On a relief edge ($\alpha_{\mathbf{p}} = 1$), the principal curvatures should be small (on an ideal step edge, the edge point is planar) with respect to the surface resolution (thus, the use of l). If the relief edge bends, the ratio of the principal curvatures is small. On a valley or a ridge ($\alpha_{\mathbf{p}} = 0$), the ratio between the principal curvatures is large.

The prominent direction on the whole surface: In the previous section we defined the prominent field on the feature curves. To extend the definition to the whole surface, we search for the smoothest direction field that satisfies the values of the prominent field on the features.

Utilizing the Laplacian as the smoothness measure, we define the prominent field as the solution of the Poisson equation. The values of the prominent field on the features serve as boundary conditions for the equation.

Formally, we want to compute the prominent field $\mathbf{s}_{\mathbf{p}} = [s^u_{\mathbf{p}}, s^v_{\mathbf{p}}]$, such that the Laplacian $[\Delta s^u_{\mathbf{p}}, \Delta s^v_{\mathbf{p}}]$ of the field components is equal to zero and $\mathbf{s}_{\mathbf{p}} = \mathbf{r}_{\mathbf{p}}$ on the features. Let $\beta_{\mathbf{p}} \in [0,1]$ be our confidence that **p** is a feature point (explained below). At each point **p**, the following should hold:

$$\begin{aligned} \beta_{\mathbf{p}}\mathbf{s}_{\mathbf{p}} &= \beta_{\mathbf{p}}\mathbf{r}_{\mathbf{p}}, \\ (1-\beta_{\mathbf{p}})\Delta s^u_{\mathbf{p}} &= 0, \\ (1-\beta_{\mathbf{p}})\Delta s^v_{\mathbf{p}} &= 0. \end{aligned} \tag{6.3}$$

Thus, on the features ($\beta_{\mathbf{p}} \approx 1$), the first equation enforces the boundary conditions and elsewhere, the two other equations enforce the smoothness of the solution.

We approximate the confidence value $\beta_{\mathbf{p}}$ such that it is close to one when the point is near an edge and zero otherwise. Recall that the points on the edge are characterized either by a high ratio between the principal curvatures (near ridges and valleys) or by a small difference between the surface S and its approximated step edge ε_r [72, Equation 7]. Specifically,

$$\beta_{\mathbf{p}} = \begin{cases} 1 & (|k| > 2 \text{ and } \max(|\kappa_1|,|\kappa_2|) > 3/l) \text{ or } \varepsilon_r < \theta/l \\ 0 & \text{otherwise,} \end{cases} \tag{6.4}$$

where θ is a user controlled threshold. The threshold enables the user to determine what points are considered edges and directly influence the appearance of the prominent field. We observed that setting θ to 2 gives good results in most cases.

Computation of the prominent field: In practice, our surface is given as a triangular mesh. We need to compute $[s^u_{\mathbf{p}}, s^v_{\mathbf{p}}]$ for every vertex of the mesh, and the entire collection will constitute a field. Let $\mathbf{p}$ be a vertex of the mesh and $N(\mathbf{p})$ be the set of the neighbors of $\mathbf{p}$ (i.e., vertices that share a mesh edge with $\mathbf{p}$).

To compute the prominent field $\mathbf{s}_{\mathbf{p}}$ we need to solve Equation 6.3. In order to perform this efficiently, we restrict ourselves to solving a couple of systems of linear equations. We do it by first deriving a linear approximation of the Laplacian $[\Delta s^u_{\mathbf{p}}, \Delta s^v_{\mathbf{p}}]$ of the components of the prominent field and then solving the set of linear equations in $\mathbf{s}_{\mathbf{p}}$.

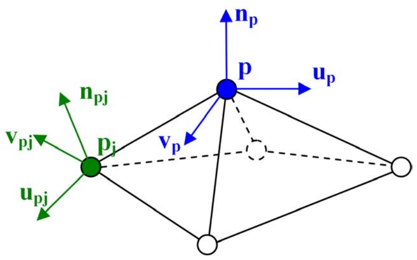
To compute the Laplacian of a scalar function f on a mesh, we follow [89]:

$$\begin{aligned}\Delta f(\mathbf{p}) &= \frac{1}{2A}\sum_{j\in N(\mathbf{p})}(\cot(\gamma_j)+\cot(\delta_j))(f(\mathbf{p})-f(\mathbf{p}_j)) \\ &\equiv \frac{1}{2A}\sum_{j\in N(\mathbf{p})} w_j(f(\mathbf{p})-f(\mathbf{p}_j)),\ w_j\equiv\cot(\gamma_j)+\cot(\delta_j),\end{aligned} \tag{6.5}$$

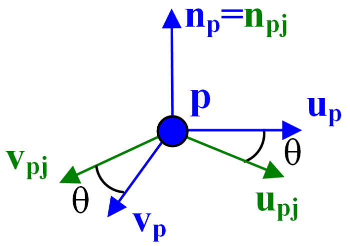
where A is the area of the Voronoi cell of $\mathbf{p}$, and γ_j and δ_j are the angles opposite the edge $[\mathbf{p},\mathbf{p}_j]$ of the triangles sharing this edge.

$[\Delta s^u_{\mathbf{p}}, \Delta s^v_{\mathbf{p}}]$ cannot be computed directly using Equation 6.5, since the components $[s^u_{\mathbf{p}}, s^v_{\mathbf{p}}]$ of the prominent field are not scalar functions of the surface. Rather, they are defined in the local tangent plane, which differs from point to point. To address this problem, we calculate the transformation between the local coordinate systems of the neighboring vertices and utilize it in the computation of the Laplacian.

Given a vertex $\mathbf{p}$ and its neighbor $\mathbf{p}_j$, this transformation is computed as follows. First, the tangent plane of point $\mathbf{p}_j$ is rotated by aligning the normals of $\mathbf{p}$ and $\mathbf{p}_j$. Next, the coordinates systems are aligned in the tangent plane, by applying a 2D rotation by θ: $R = [(\cos\theta,\sin\theta)^T, (-\sin\theta,\cos\theta)^T]$, where θ is the angel between the rotated first principal directions (Figure 6.6).



(a) Initial positions



(b) After aligning the normals

Figure 6.6: **Alignment of the local coordinate systems.** (a) First, we rotate the coordinate system of $\mathbf{p}_j$ so that the tangent plane of $\mathbf{p}_j$ coincides with the tangent plane of $\mathbf{p}$. The rotation is performed around the cross product of $\mathbf{n_p}$ and $\mathbf{n_{p}}_j$. (b) Then, the coordinate systems of $\mathbf{p}_j$ and $\mathbf{p}$ are registered by rotating the rotated tangent plane of $\mathbf{p}_j$ by θ.

Finally, the Laplacian of the prominent field can be written as:

$$\begin{aligned}\Delta s^u_{\mathbf{p}} &= \frac{1}{2A}\sum_{j=N(\mathbf{p})} w_j(s^u_{\mathbf{p}} - \cos\theta s^u_{\mathbf{p}} - \sin\theta s^v_{\mathbf{p}}),\\ \Delta s^v_{\mathbf{p}} &= \frac{1}{2A}\sum_{j=N(\mathbf{p})} w_j(s^v_{\mathbf{p}} + \sin\theta s^u_{\mathbf{p}} - \cos\theta s^v_{\mathbf{p}}).\end{aligned} \tag{6.6}$$

Finally, substituting equations 6.4 and 6.6 into Equation 6.3 yields a system of linear equations, whose unknowns are the components of the prominent field $\mathbf{s_p}$. This is solved using a standard sparse linear equation solver. Since the prominent field is a direction field, it is then normalized. Note that after normalization, the Laplacian is not guaranteed to remain small. In practice, however, the change is negligible.

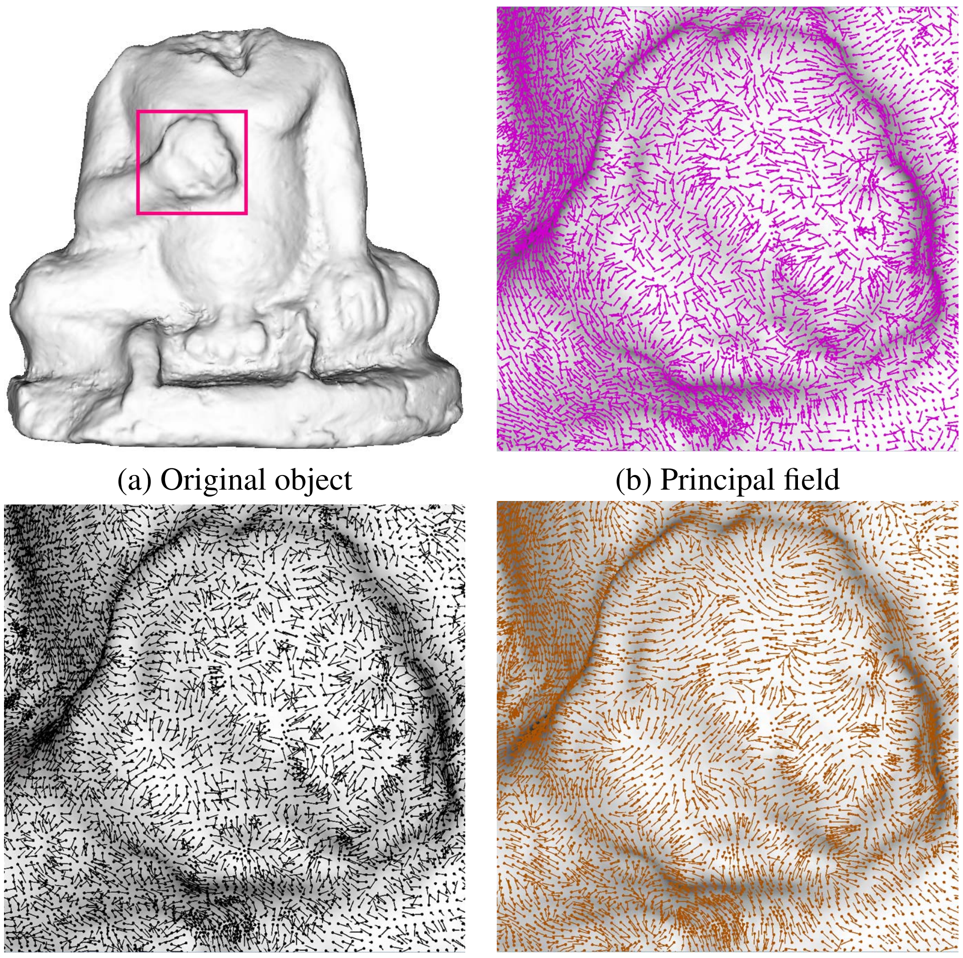
Figure 6.7 compares the prominent field to the other direction fields. It can be seen that the principal direction field and the relief direction field are inappropriate in most surface locations, whereas our prominent field is both in the desired direction near the feature curves and smooth everywhere.

6.4 Application: Surface enhancement and smoothing

Scanned archaeological objects are often unsuitable for further processing and visual analysis, either due to erosion that they underwent during the ages or due to scanning noise. This section describes how our prominent field can be utilized to enhance and smooth these objects, to enable effective processing and analysis.

One way to address these problems is by using adaptive filtering algorithms, which smooth (or denoise) the surface, while keeping the features intact or enhancing them. Existing approaches for adaptive filtering on meshes operate either on the mesh vertices [28, 36, 146], the mesh normals [97, 126], or the curvatures [31]. The techniques differ in the energy functional they attempt to minimize.

While these approaches perform well preserving and enhancing ridges and valleys, they


(a) Original object (b) Principal field

(c) Relief field (d) Prominent field

Figure 6.7: **Direction fields.** The principal directions (b) and the relief directions (c) lack meaning far from their respective feature curves, in contrast to the prominent field, which is in the desired direction near the feature curves and smooth everywhere (d).

are not designed for relief objects. In particular, there are a couple of cases in which they may produce inferior results. The first case occurs when no distinct ridges or valleys can be detected on the surface. These approaches will simply smooth the objects, diminishing the 3D features. The second case occurs when there exist distinct valleys and ridges, but the slope of their step edge is shallow, as illustrated in Figure 6.8. In this case, these approaches aim at enhancing each of these features separately, but do not enhance the step edge model between them. Our goal is to preserve and enhance this step edge by steepening the slope of the step edge.

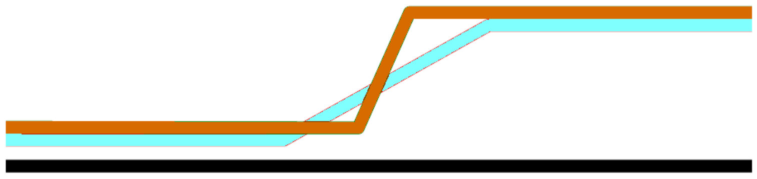


Figure 6.8: **The cyan curve is the local image defined on the black base.** Since this surface has sharp ridges and valleys, it will not be enhanced by standard adaptive filtering. The desired result, illustrated in orange, enhances the 3D feature.

We propose a novel approach that solves these problems. It consists of two steps – bilateral filtering and inverse curvature flow – each makes use of our prominent field to guide the smoothing and enhancement directions. Though we describe a specific bilateral filtering, our prominent field can be combined with many other adaptive filtering techniques.

Step 1 – Bilateral filtering: A bilateral filter sets the position of a vertex to a weighted average of its neighbors. The weights depend both on the distance between the vertices and on their similarity. We propose to base the similarity component on the distance between the vertices along the prominent direction.

Let $\mathbf{p}$ be a vertex on the mesh, $N(\mathbf{p})$ be the set of its neighbors, $d_j = \|\mathbf{p} - \mathbf{p}_j\|$ be the Euclidean distance between $\mathbf{p}$ and one of its neighbors $\mathbf{p}_j$, and $\mathbf{n}_\mathbf{p}$ be the normal at $\mathbf{p}$. In [36] it is proposed to define the similarity as the distance between $\mathbf{p}_j$ and $\mathbf{p}$'s tangent plane: $h_j = |\langle \mathbf{n}_\mathbf{p}, \mathbf{p} - \mathbf{p}_j \rangle|$, so that smoothing is performed when $\mathbf{p}_j$ is close to the tangent plane of $\mathbf{p}$. We propose to add to this definition a term that depends on r_j, the projection of $\mathbf{p} - \mathbf{p}_j$ along the prominent direction. Thus, smoothing will not be performed in the prominent direction, which prevents 3D feature blurring. This is done by multiplying the weights suggested in [36] by the term $e^{-r_j^2/2\sigma_p^2}$.

Hence, our similarity-based change of $\mathbf{p}$ in its normal direction is

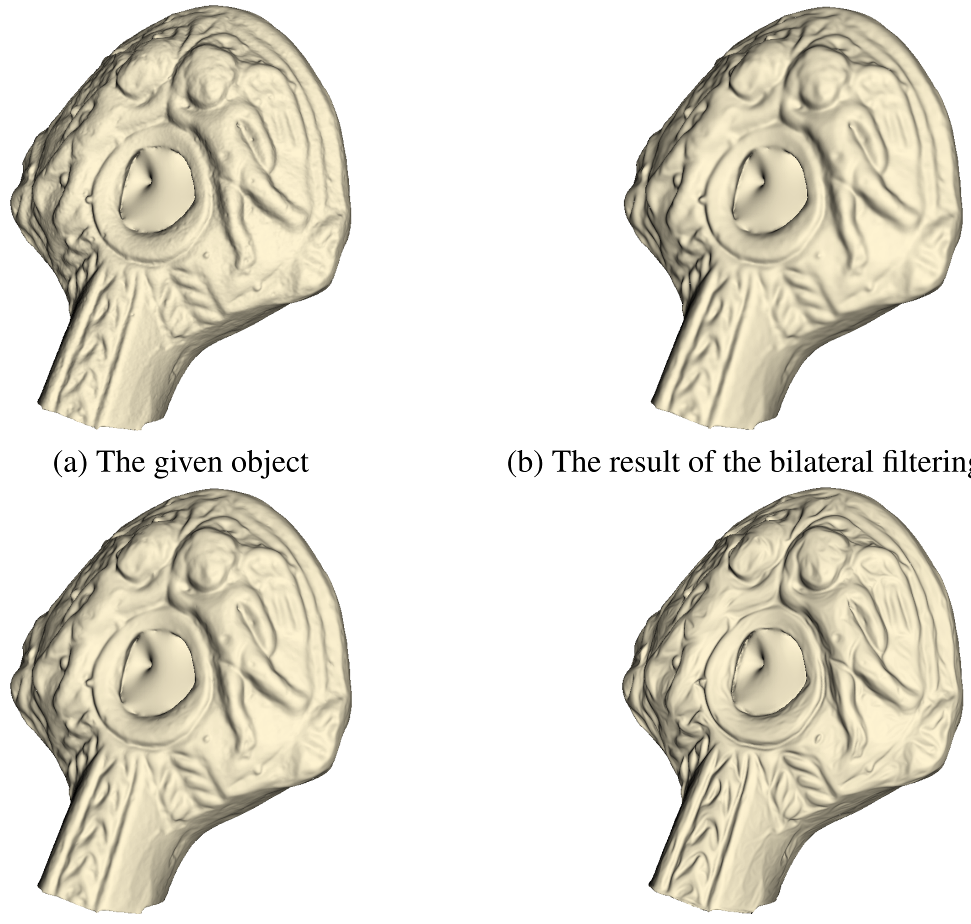
$$\delta_\mathbf{p} = C \sum_{j \in N(\mathbf{p})} e^{-d_j^2/2\sigma_c^2} \cdot e^{-h_j^2/2\sigma_s^2} \cdot e^{-r_j^2/2\sigma_p^2} \cdot h_j, \tag{6.7}$$

yielding a new position for $\mathbf{p}$:

$$\mathbf{p}' = \mathbf{p} + \delta_{\mathbf{p}}\mathbf{n}_{\mathbf{p}}, \tag{6.8}$$

where, C is the normalization coefficient. In the implementation, $\sigma_s = 0.5\sigma_c$, $\sigma_s = 0.4\sigma_c$, and σ_c is a user-supplied parameter that determines the amount of smoothing. It is common to slightly smooth the object prior to computing the distances.

Figures 6.9(c) & 6.10(c) show the results obtained by applying our bilateral filtering to scans of real archaeological artifacts. In comparison to [36] (Figures 6.9(b) & 6.10(b)) it can be seen that the features are more pronounced.



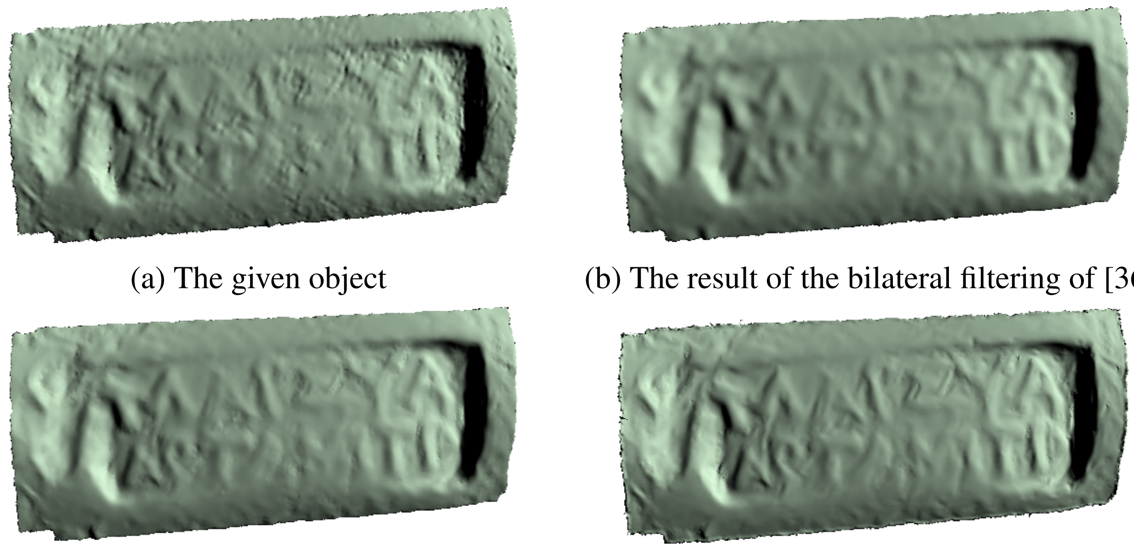
(a) The given object

(b) The result of the bilateral filtering of [36]

(c) The result of our bilateral filtering (Step 1)

(d) Our final result

Figure 6.9: **Enhancement of a late Hellenistic oil lamp from the first century BCE.** In our result, the limbs of the cupid, as well as the ornaments, are more pronounced.



(a) The given object

(b) The result of the bilateral filtering of [36]

(c) The result of our bilateral filtering (Step 1)

(d) Our final result

Figure 6.10: **Enhancement of a Hellenistic handle stamped by a Greek official.** In our result, the letters are crisper, whereas the bumpy background is smoothed.

Step 2 – Inverse-curvature flow: The inverse-curvature flow is a high frequency filter [2, 128, 129], which updates the position of a vertex so as to increase the absolute value of its curvature. It can be based on the mean, maximum, minimum, or any other type of curvature.

While the inverse-curvature flow manages to enhance features, it suffers from two drawbacks. First, it often creates spurious features on the surface, in addition to the enhanced features. This is so since in near-flat region points with locally higher-curvature values are enhanced. Second, it is an iterative process that does not have a well-defined stopping criterion, causing unnaturally exaggerated features.

We propose a new inverse-curvature flow, which is based on two modifications to the standard flow. To solve the first problem, the curvature is computed in the prominent direction, enhancing only the real features. To solve the second problem, a new stopping criterion is suggested, which is based on the intuition in which a point should not exceed the extremum of the height function in the neighborhood of the point. Figure 6.11 illustrates the problem and our solution. It can be seen that the standard inverse curvature flow results in an exaggerated edge (magenta) that exceeds the original height in the neighborhood of the point (cyan). Our edge (orange) stops when it reaches the maximal height, resulting in a more appealing surface.

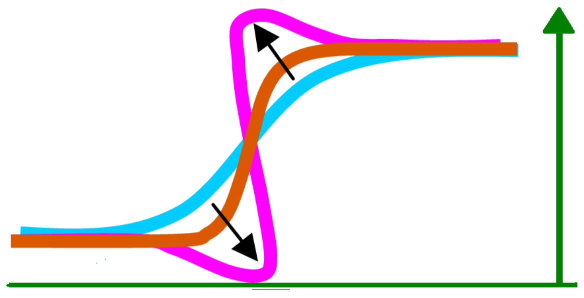


Figure 6.11: **Inverse-curvature flow.** The initial surface is in cyan; the standard inverse-curvature flow is in magenta, and our inverse-curvature flow is in orange. The base is the green line and the normal to the base is the green arrow. Our inverse-curvature flow does not exceed the maximum (minimum) local height.

To perform this computation, we need to estimate the relative height function over the base. However, the decoupling of the surface into a base and a height function is unknown (Figure 6.5). To approximate this decoupling, it suffices to estimate the normal to the base at each point [72]. To estimate the base normals, the surface (S) normals are smoothed at the neighborhood of the point. Our approach utilizes an adaptive Gaussian filter, similarly to [97, 72].

Results: Figures 6.9-6.10 illustrate some results and comparisons. Figure 6.9 shows a Hellenistic oil lamp. The original object is slightly eroded and has small ripples on the surface. Standard bilateral filtering succeeds to remove the ripples but the details become blurred. Our bilateral filtering (Step 1) causes the blurring (e.g., on the legs), but the result can still be enhanced. After Step 2 of our algorithm, the surface becomes smooth while the details become crisper. For instance, the torso of the cupid on the right is smooth while his arms, legs, and wings are more clearly visible. Figure 6.10 depicts a Hellenistic handle stamped by a Greek official. The defects on the original scan are removed by both bilateral filters, but the letters in our final result are more recognizable and protruding.

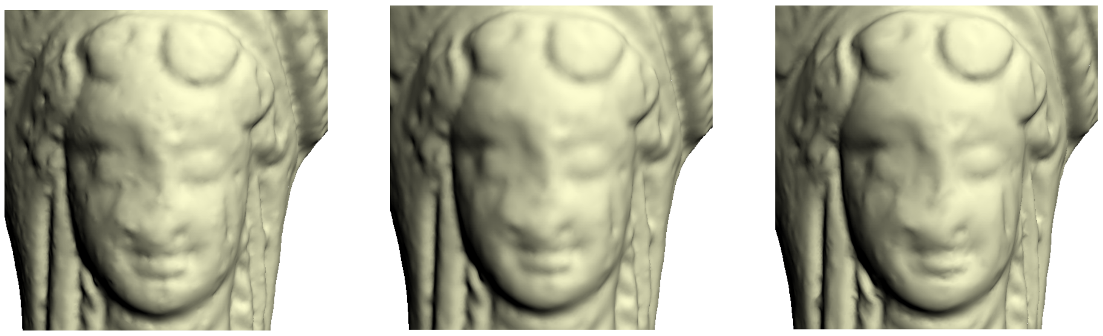

(a) The given object (b) After our bilateral filtering (c) Our final result

Figure 6.12: **A late Hellenistic oil lamp from the first century BCE.** Note especially the eye on the left from which the noise has been removed and its shape is more pronounced.

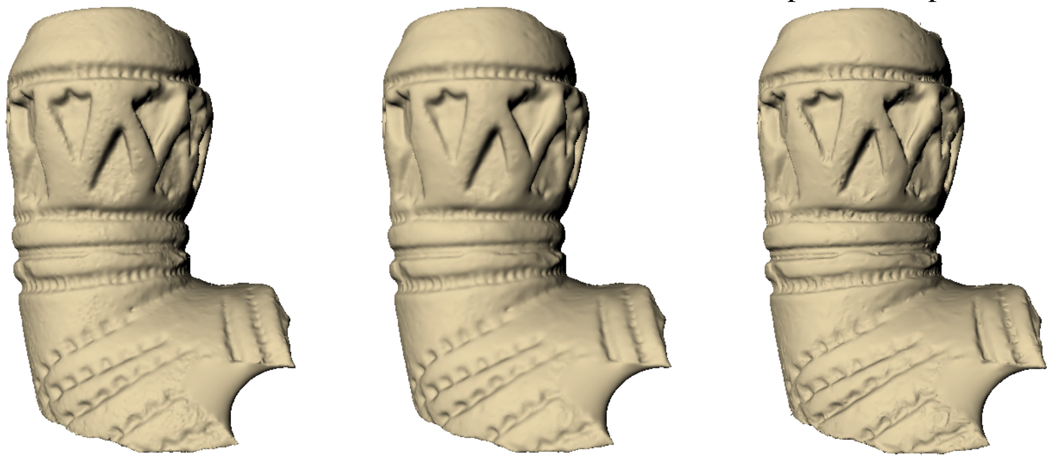

(a) The given object (b) After our bilateral filtering (c) Our final result

Figure 6.13: **An Ottoman pipe.** Note the quality of small carvings which were enhanced by the algorithm.

Additional results are shown in Figures 6.12-6.14. After applying our algorithm, the face in Figure 6.12 has much clearer facial features. Note especially the quality of the nose, the eyes, and the crown. Figure 6.13 displays an Ottoman pipe. Our algorithm keeps intact even the smallest carvings on the pipe. Figure 6.14 displays a shard of a Hellenistic vase. In this example the user decided to stop the iterations of the algorithm before the automatic stopping criterion was reached. The resulting object (Figure 6.14(a)) looks more appealing and has a crisper ornament. Zooming in to the details of the shard (Figure 6.14(b)) it can be seen that the quality of our result is even more visible. For instance, the S-shaped ornament

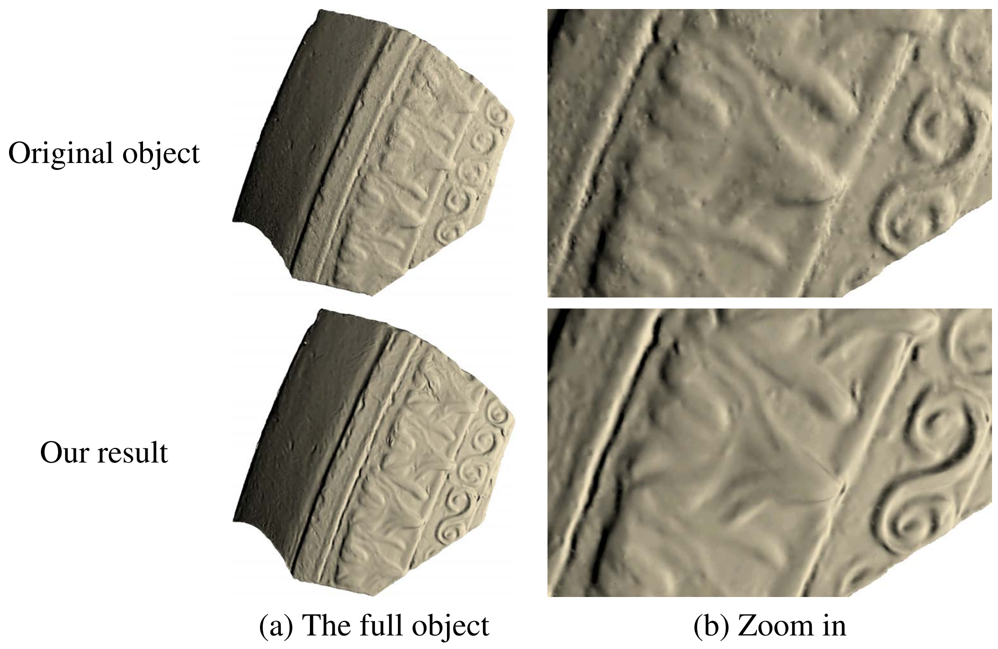


Figure 6.14: **A shard of a Hellenistic vase.** The algorithm removes the noise visible on the left while enhancing the S shaped decoration on the right.

is smoother and has clearer boundaries.

6.5 Application: Prominent coloring

Traditionally, archaeological artifacts are drawn by hand and printed in the reports of archaeological excavations, as illustrated in Figure 6.15. The artists utilize artificial coloring in order to enhance the three-dimensional features. Several kinds of computerized artificial coloring methods have been proposed in the literature [18, 40, 66, 112, 132], in which the object is colored according to its geometric properties. For instance, it is proposed in [66] to color each point on the surface according to its mean or maximal curvature.

We propose a new method for artificial coloring, termed *prominent coloring*. The color of a vertex is set according to its curvature in the prominent direction. The lower the curvature, the darker its color. Formally, given a vertex with prominent curvature κ_p, its color is

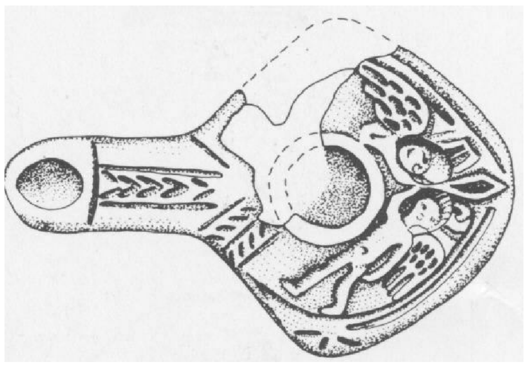
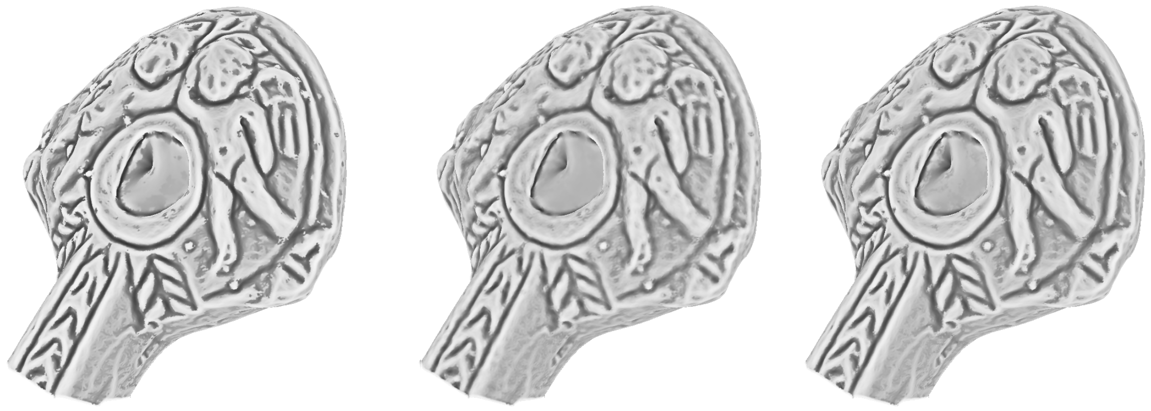


Figure 6.15: **Manual illustration of an archaeological artifact** [122]



(a) Maximal-curvature coloring (b) Mean-curvature coloring (c) Prominent coloring

Figure 6.16: **Comparison of various coloring methods.** The prominent coloring combines the advantages of the mean and maximal coloring – a smooth image with clear boundaries.

defined as

$$\text{color} = \arctan(\lambda \kappa_p), \tag{6.9}$$

where λ is a user supplied parameter controlling the overall image contrast.

Results: Figures 6.16-6.18 illustrate our coloring method and compares it to mean and maximal curvature colorings. Prominent coloring can emphasize poorly visible features. For instance, the scar on the cheek and the cavity on the crown of the person in Figure 6.17 are not clearly seen on the original scan. With our prominent coloring, they are easily

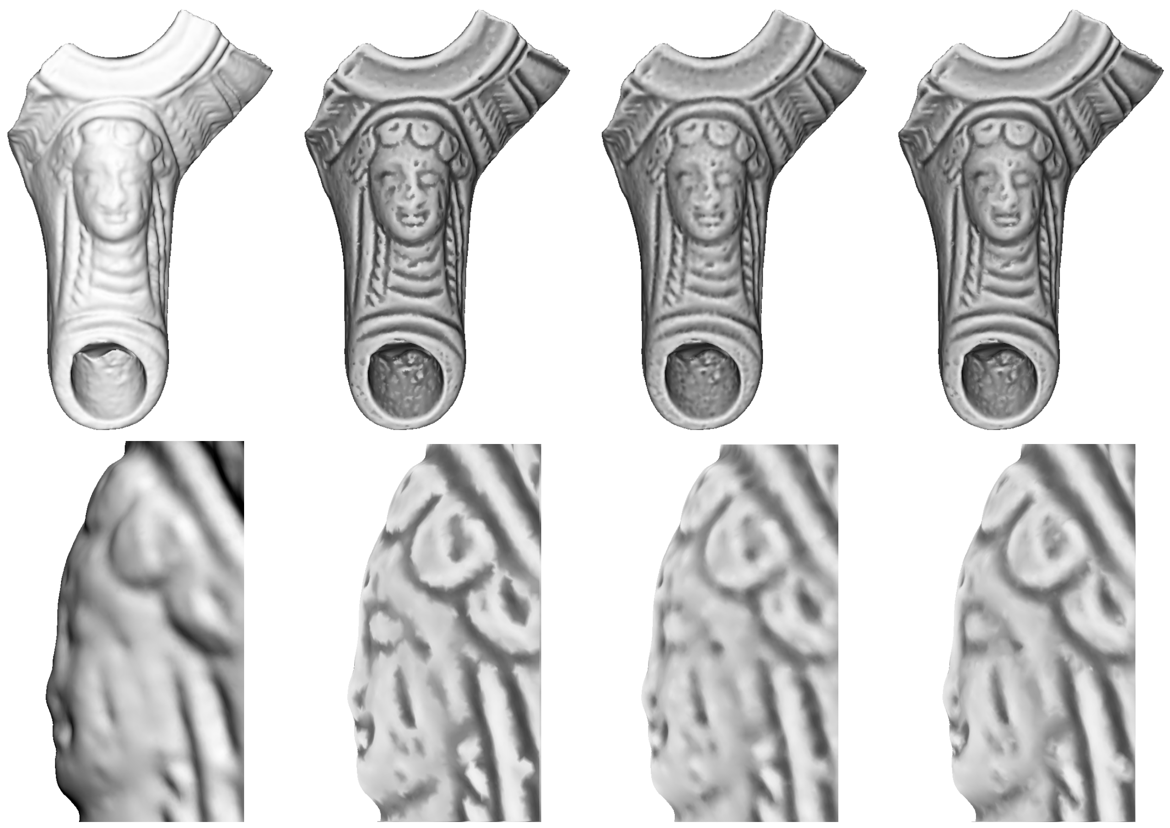
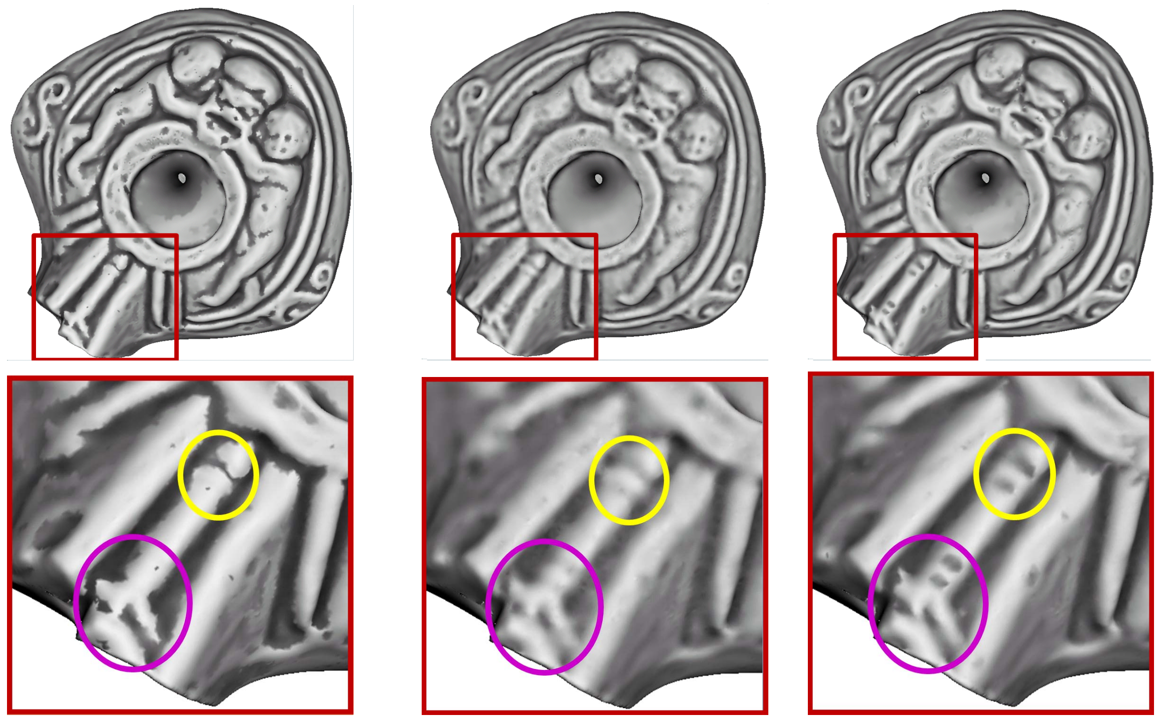


Figure 6.17: **Comparison of various coloring methods.** Top: complete artifact; bottom: partial profile. Note that the maximal-curvature coloring is noisy; the mean-curvature coloring is blurred; our coloring is crisper and less noisy. This is visible, for instance, on the eye, crown, and hair.

detectable.

In general, the maximal curvature coloring is noisy and jagged since the maximal curvature is sensitive to noise and to small surface variations. The mean curvature coloring is blurred since it depends also on the base surface and not only on the details. On the contrary, the prominent coloring manages to produce clear and smooth boundaries. It combines the advantages of the mean and maximal coloring – a smooth image with clear boundaries. For example, these differences can be observed on the eye and the crown in Figure6.17. The prominent coloring better emphasizes the relevant details. This can also be seen in Figure 6.16. The prominent coloring is smoother than the maximal coloring and crisper than the mean coloring.

Moreover, the maximal and the mean coloring can generate spurious details or remove the true ones. For instance, observe the small cavities denoted by the yellow and magenta circles in Figure 6.18. The mean coloring completely removes them. The maximal coloring removes one cavity and changes the shape of the other.



(a) Maximal-curvature coloring (b) Mean-curvature coloring (c) Prominent coloring

Figure 6.18: **Comparison of various coloring methods**. With maximal curvature (a), the area inside the yellow circle appears as if it is divided into two parts and inside the magenta circle appears broken. With mean curvature (b) both area are blurred. The prominent coloring (c) better depicts these areas.

6.6 Conclusions

This paper addressed the problem of automatic processing of scanned artifacts. The processing is based on a definition of a new field – the prominent field, which is a smooth direction field perpendicular to the feature curves. The prominent field is computed in interactive time (a couple of seconds for a 100,000-vertex model). We demonstrated how to employ the prominent field for two applications: surface enhancement and artificial surface coloring, which emphasizes the object features. In both cases, the methods were applied to archaeological artifacts, which are typically noisy and suffered erosion over time. We showed that our results outperformed the results obtained by previous methods.

The applications received positive feedback from archaeologists from the Computerized Archaeological Laboratory at the Hebrew university of Jerusalem. Their impression is that “it is not just a nice way to visualize and publish archaeological objects, but also an important research tool that improves the interpretation of the items especially with written material.” Moreover, they intend to use this enhancement method in their future publications.

In the future, we intend to apply our prominent field to other applications, such as shape matching and reconstruction.

Acknowledgements: This research was supported in part by the Israel Science Foundation (ISF) 628/08, Olendorff foundation, the joint Technion University of Haifa research foundation, and the Goldbers fund for electronics research. We thank Dr. A. Gilboa and Zinman Institute of Archaeology for supplying us the models which were excavated at Tel Dor and for fruitful discussions on the topic.

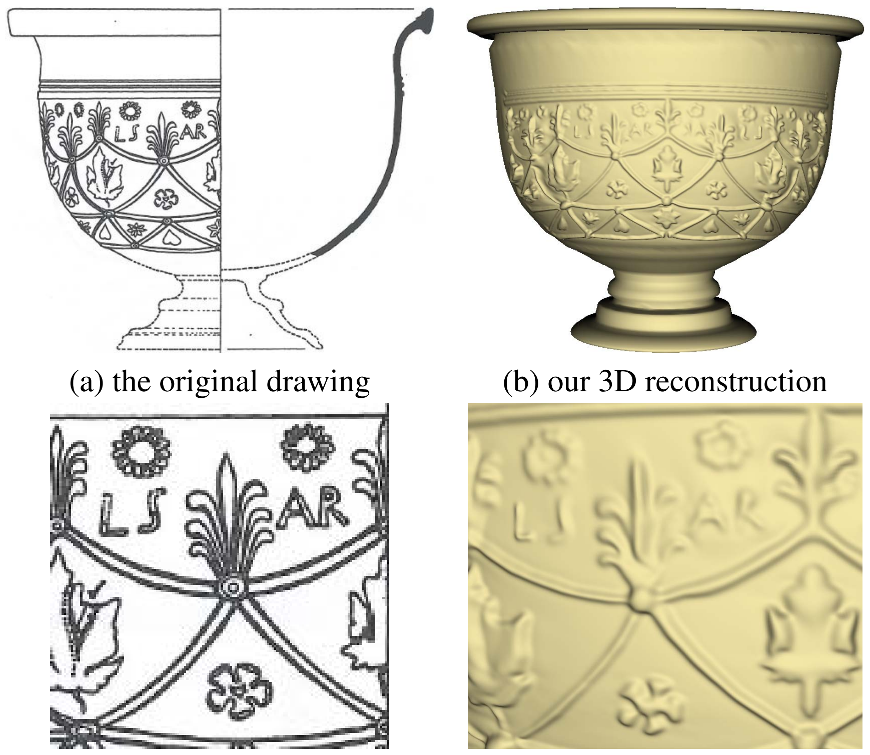
Chapter 7

Reconstruction of relief objects

Abstract

This paper addresses the problem of automatic reconstruction of a 3D relief object from a line drawing. The problem is challenging due to five reasons: the small number of orthogonal views of the object, the sparsity of the strokes, their ambiguity, their large number and their inter-relations. We partition the reconstruction problem into two sub-problems. First, we reconstruct the underlying smooth base of the object from the silhouette. Assuming that the variation of bases belonging to the same class of objects is relatively small, we create the base by modifying a similar base retrieved from a database. Second, we reconstruct the relief on top of the base. Our approach is able to reconstruct the relief from a complex drawing that consists of many inter-related strokes. Rather than viewing the inter-dependencies as a problem, we show how they can be exploited to automatically generate a good initial interpretation of the line drawing. These drawings are highly challenging, since artists created very complex and detailed descriptions of artifacts regardless of any considerations concerning their future use for shape reconstruction.

The conference version of this paper appeared in CVPR 2011 and the journal version was published in JOCCH, 2013. The full name is *Reconstruction of relief objects from archaeological line drawings*.



(a) the original drawing (b) our 3D reconstruction

(c) a zoomed version of (a) (d) a zoomed version of (b)

Figure 7.1: **Reconstruction from a manual drawing consisting of 571 curves.** North Italian Sigillata. A cup by L. Sarius (name appears on the cup) 10 B.C - 30 A.D [13] catalog number 273.

7.1 Introduction

Line drawings have been the standard method of archeological artifact documentation for many years. While many findings might have been lost or destroyed, their illustrations remain. Therefore, the only existing input for reconstruction are the line drawings appearing in the archaeological report (Figure 7.1(a)). Usually, production of such drawings is derived from the analysis of an expert archaeologist. The archaeologist translates into a drawing his interpretation of the object. So our goal is not to recreate a digital replica of the object itself, but to recreate a missing object as it was perceived by the scientist who drew the archeological report. These inputs are usually highly complex. State-of-the-art algorithms for automatic reconstruction were not designed with such drawings in mind.

Line drawings are used to represent different kinds of findings, such as buildings, statues, and others. In this work, we focus on artifacts containing reliefs. They are important sources of information, since they are fingerprints of periods and cultures. In this paper we applied our algorithm to real examples from archeological reports of two types: Hellenistic and Roman relief pottery [13] and Cosa lamps [34].

Automatic reconstruction of 3D relief object from a line drawing can be divided into two sub-problems – reconstruction of the base and reconstruction of the details (i.e. the relief) on top of the base. The main challenge of automatic base reconstruction is the fact that we are given only one or two views of the silhouette, which is insufficient. Given the base, automatic relief reconstruction from a line drawing is a challenging task due to several reasons. First, the lines are usually sparse and thus, the object is not fully constrained by the input. Second, the line drawings are often ambiguous, since the lines may have different geometric meanings – they can indicate 3D discontinuities, surface creases, or 3D step edges. Third, the input may consist of a large number of strokes that need to be efficiently handled by the algorithm. Fourth, these strokes are inter-related. For instance, in the relief of Figure 7.1, the decorations are either protruded or indented as a whole, and a solution in which some of the lines indicate protrusions and others indentations is less likely.

Related work: Reconstruction of a 3D object from a line drawing is a fundamental problem in computer vision; see [21] for a survey. Approaches to reconstruction from a line drawing typically consist of two steps: line labeling and the reconstruction itself. Line labeling focuses on finding a set of consistent labels given a set of lines [17, 50, 81, 116, 139]. A line is assumed to indicate depth or orientation discontinuity of an object. A label states whether the line represents a concave or convex edge or an occlusion.

Reconstruction algorithms build a 3D object from the labeled lines. Usually, the algorithms assume that the object consists of planar faces [85, 125]. They find a set of consistently oriented faces that generate a feasible object. Recent algorithms are also able to handle drawings composed of arcs and not only straight lines [16, 140].

The above algorithms are less suitable for relief objects because of two reasons. First, the algorithms handle only specific types of lines – lines representing depth or orientation discontinuities. Relief objects, however, usually do not have such discontinuities. Instead, they are described by general lines representing 3D step edges. Second, the algorithms are designed to reconstruct CAD-like objects and do not handle effectively highly curved objects.

A related problem was addressed in computer graphics, denoted by *sketch-based modeling*. Most of the work modeled general objects [51, 59, 95], and [38]. The goal is to generate the smoothest-possible object constrained by the given line. The techniques provide intuitive interfaces and generate visually-pleasing results. However, the underlying smoothness assumption results in smooth, blob-like objects, which cannot accurately convey the details of reliefs.

Techniques that focus on relief editing can produce accurate surfaces of various types [39, 62, 142, 55]. However, it might be difficult to employ them on drawings consisting of a large number of curves. First, they require the user to enter manually the parameters of each curve. This is a tedious and time consuming process. Second, they assume that the user input is consistent. This assumption makes the editing of multiple inter-related curves very challenging.

Our approach: We propose an approach that is able to reconstruct a relief from a complex drawing that consists of many inter-related strokes, such as the one in Figure 7.1. The algorithm manages to compute good results automatically, by setting for each curve its interpretation, i.e., its shape parameters. Yet, the user is allowed to fine-tune the interpretation of the drawing, if required.

The approach is derived from the observation that a relief object can be represented as a composition of a smooth base surface and a height function (relief) defined over that base [72, 82]. Therefore, our reconstruction task consists of two sub-tasks: base estimation and relief reconstruction.

For the base estimation, we assume that the variation of bases that belong to the same class of objects is relatively small and the base can be constructed as a modification of one of the bases in a database. Given the drawing, a database of 3D models is searched for models having similar silhouettes. The most similar model is deformed so that its silhouettes best match the drawings. The deformation is performed by solving a linear optimization problem in which the silhouettes provide the boundary conditions.

As for the second sub-task – relief reconstruction – we assume that the base is given. The algorithm is based on two key ideas. First, the inter-dependencies between the strokes of the line drawing can be exploited to automatically generate a good initial interpretation of the line drawing. Second, given an interpretation, it is possible to reconstruct a consistent surface.

For each idea, we provide a novel algorithm that solves the corresponding problem. To interpret the detailed line drawing, we show that our problem can be represented as the problem of topological ordering of a graph and solved efficiently. Given the base and the line-drawing interpretation, the surface is reconstructed by posing it as a pair of linear optimization problems, which can be readily solved.

The contribution of this paper is three-fold. First, the major contribution is an efficient algorithm for reconstruction of reliefs from line drawings. The method is able to handle highly complex real drawings (Sections 7.2-7.4). Second, we introduce an algorithm for automatic estimation of a base from the outline of the given line drawing (Section 7.5). Both algorithms are realized in an interactive system (Section 7.6). Last but not least, the method makes a significant step towards solving an important problem in archaeology (Section 7.7) – a domain that recently attracted a lot of attention in computer vision and computer graphics [12, 69, 72, 110].

A preliminary version of this paper was presented in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [70].

7.2 Relief reconstruction – general approach

Given a line drawing of a relief object our goal is to reconstruct the surface as shown in Figure 7.1. The base is estimated from the line drawing as will be explained in Section 7.5. The algorithm then reconstructs the relief on the base regardless of its shape. This is the focus of the current section. Below we present the problem definition and outline our algorithm.

Problem definition: For most relief objects, the details can be described as a *height function* defined on a surface termed *the base*.

The lines of drawings of relief objects typically indicate changes of the height function. This height function is usually smooth far from the lines and constant along them. Its gradient is strongest near the lines in the direction perpendicular to the lines. Hence, the value of the height function in the line's neighborhood depends only on the distance from the line.

A drawing consists of *drawing curves*, *junctions*, and *margins* (Figure 7.2(a)). We define the *drawing curves*, or simply the *curves*, as the lines of the given drawing. They indicate visually-meaningful locations on the relief. The curves may be connected by *junctions*, but cannot cross them. *Margins* are the borders of a curve's neighborhood.

A drawing defines *the relief* (the height function). Outside the margins, the relief is assumed to be smooth. Inside the margins the relief is approximated by a step edge of height h and width w (Figure 7.3). Height h can be both positive or negative, representing the direction of the edge. The shape of the relief defined by step edge $s(x)$ is $p(u)$:

$$p(u) = h \cdot s(u/w), \tag{7.1}$$

where $u \in [-w, w]$ is the width parameter of the step edge.

The sought-after surface is a combination of the base B and the relief. If we look at a cross section perpendicular to the curve, $B(u)$ is defined at a point on B at a (signed) distance u

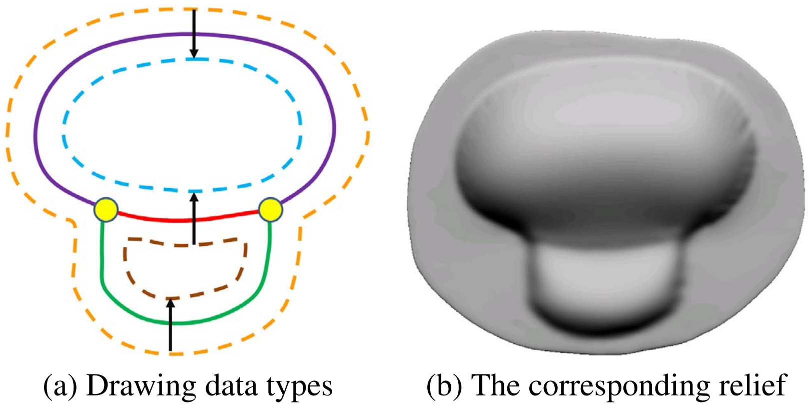

(a) Drawing data types (b) The corresponding relief

Figure 7.2: **Notations.** Curves are solid lines, junctions are yellow circles, margins are dashed lines, and step edge directions are arrows. Each curve and margin is drawn in a different color.

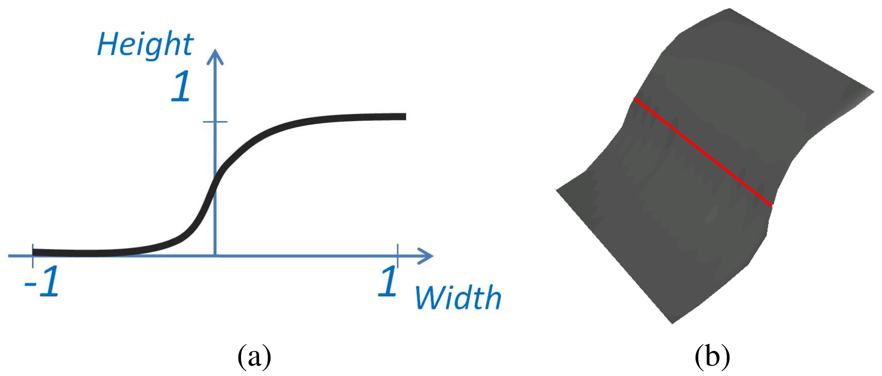


Figure 7.3: **A step edge approximation.** (a) A cross section of a normalized step edge. (b) A 3D view of a normalized step edge.

from the curve. Thus, the surface S within the curve's margins on the cross section can be written as:

$$S(u) = p(u) + B(u) = h \cdot s(u/w) + B(u). \tag{7.2}$$

See Figure 7.4 for an illustration.

We can now rephrase our goal. Given a line drawing, we want to compute the relief object, such that near the curves the shape of the cross section will match the 3D step edge p, whereas elsewhere its shape will be smooth and similar to the base B. Specifically, we need to compute the height h for the step edge of every curve in a consistent manner.

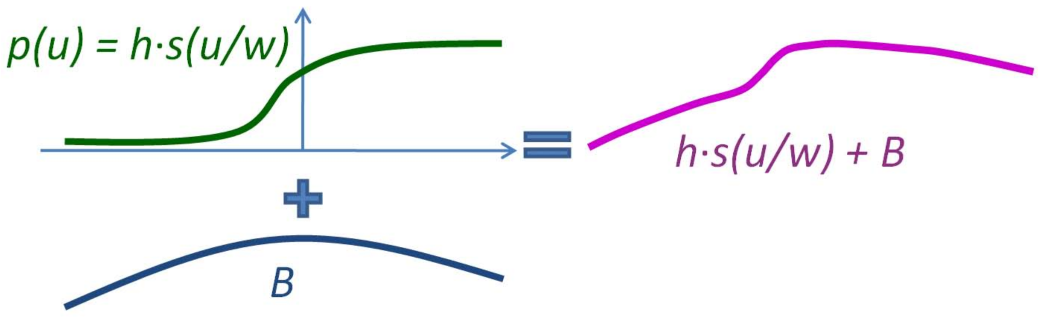


Figure 7.4: **The relief in a curve's neighborhood is a combination of the step edge $p(u)$ and the base B.**

We assume that the margin width w and the step edge's shape $s(x)$ are constant for all the curves. In our system the margin width is set to 0.05% of the diagonal of the object's bounding box. The step edge shape is the shape of an ideal step edge smoothed with a Gaussian of standard deviation equal to the average edge length. Both w and the shape of the edge $s(x)$ can be later modified by the user.

Algorithm overview: In a pre-processing step, the curves are extracted from the image of the drawing, using the ridge detection algorithm of [121]. The algorithm first smoothes the image by a Gaussian filter of standard deviation equal to 2.0 pixels. Then it marks as potential ridges all pixels that satisfy the following criteria:

- Low gradient. The derivative of the image should be close to zero on the ridge.
- The first eigenvalue of the Hessian matrix is high and the second eigenvalue of the Hessian matrix is low. The second derivative of the image should be high in the direction perpendicular to the ridge and low along the ridge.
- The pixel intensity is low. We are searching for dark curves only.

After all the ridges have been marked, the algorithm applies non-minimal suppression to make the ridges thinner. Finally it links all ridge pixels and detects junctions. Small non-connected ridges are removed. Then, junctions are detected and margins are generated. These elements serve as the input to our algorithm. Since the input was drawn manually, we assume it does not contain noise and that all the curves represent real features. We only remove short lines that are often used for shading effects.

Initially, an automatic interpretation of the line drawing is computed, by calculating consistent height values for the step edges (Section 7.3). Given the curves and the step edges, the surface is reconstructed (Section 7.4).

7.3 Line drawing interpretation

Interpreting the line drawing requires setting the height of the step edge of each curve. Doing this manually for a complex drawing composed of dozens, or even hundreds of curves is a cumbersome process. Moreover, the set of assigned heights has to be consistent. Consistency refers to the requirement that two different paths between points should result in the same height difference (Figure 7.5). The solution to this problem should address the challenges of interpretation consistency combined with the large number of strokes.

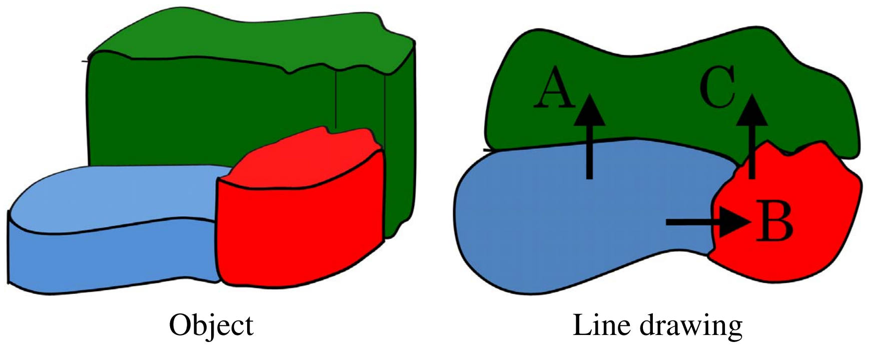
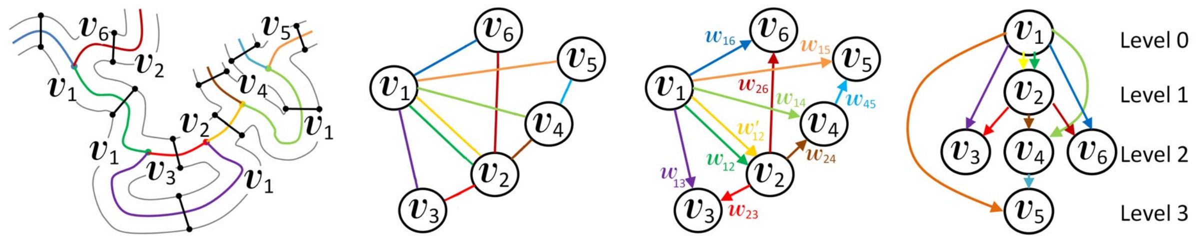


Figure 7.5: **Height consistency.** The height of step edge **A** should be equal to the sum of the heights of step edges **B** and **C**.

The key idea of our method for computing the heights of the step edges is to reduce this problem to the problem of a constrained topological ordering of a graph. A similar reduction is proposed in [127] for adding depth to cartoons. Below we first describe the reduction and then the algorithm for solving the corresponding graph problem.



(a) Original drawing (b) Graph reduction (c) Directed graph (d) Topological ordering

Figure 7.6: **Graph representation.** (a) A drawing in which each curve is colored differently. The margins are denoted by v_i. The black segments represent the step edges. (b) The corresponding graph. Each step edge is colored consistently with the curve it crosses. (c) Given the undirected graph, our algorithm directs it and finds its weights w. (d) The topological levels of the graph. The weight of an edge is equal to the difference between the levels of its nodes.

Reduction to a graph problem: Given a drawing (Figure 7.6(a)), we initially represent it by an undirected graph $G = \{V, E\}$ as follows (Figure 7.6(b)). The margins are the nodes of the graph. There exists an edge between two nodes whenever a curve lies between the corresponding margins.

Our goal is to direct the graph and to find for each directed edge a positive integer weight. The weight corresponds to the height difference between the source and the target nodes of the edge. A directed edge indicates that the margin representing the source node is lower than the margin representing the target node. The weights should be consistent, i.e., all the paths between two nodes should have the same weight. Let us denote the height of node $v \in V$ by h_v and the weight of the edge e_{km} from node k to m by w_{km}. We aim at generating a weighted directed graph that satisfies the following constraints:

1. Positive integer weight: $w_{km} > 0, w_{km} \in \mathbf{Z}$.

2. Height consistency: $h_k + w_{km} = h_m$.

Note that due to transitivity, Constraint 2 guarantees that all the paths between two nodes have equal weights.

If the user provides additional information, such as specific directions or weights, they should also be satisfied. The additional constraints are indicated as follows:

3. Respect the given directions of some edges e_{km}.

4. Respect the given weights w_{km} of some edges e_{km}.

Constraint 3 explicitly specifies the direction of some edges of the graph, i.e. which of the two nodes has a lower height. This constraint is used to make the reliefs consistently higher or lower than the base.

Our algorithm employs the following heuristic to determine this constraint automatically. Nodes representing the base margins are identified as nodes that are adjacent to many edges. Hence, we set the directions such that they point out of these nodes. As a result, the reliefs are consistently higher than the base surface.

Constraint 4 sets the weight of some edges of the graph. It can be employed to make certain parts higher or lower than those computed by the automatic algorithm. This is not mandatory and is omitted by default.

We first, explain how the graph is directed and then how the edge weights are computed.

1. Directing the graph: Given an undirected graph (Figure 7.6(b)), we aim at generating a directed graph that will allow us later to assign consistent weights according to the second constraint (Figure 7.6(c)). This will specify the direction of every step edge, i.e. specify its lower and its higher ends.

We observe that a basic requirement of the sought-after solution is that the resulting directed graph should be acyclic (DAG). This is so since as all the weights are positive, a cycle in the graph would be of a positive weight w_{cycle}, contradicting Constraint 2 that for a node v on the cycle $h_v + w_{cycle} = h_v$.

Any undirected graph may be made into a DAG by choosing a total (linear) order for its vertices and orienting every edge from the earlier endpoint in the order to the later endpoint. Our algorithm chooses an order which is consistent with Constraint 3. The direction of unconstrained edges is chosen so as to produce a DAG.

2. Assigning weights to the edges: Given a DAG, the goal is to compute positive weights of the edges, such that Constraints 1-3 hold. The weights correspond to the heights of the step edges.

Initially, the graph is partitioned into topological levels, similarly to the way done in topological sorting (Figure 7.6 (d)). In this partition, level 0 includes only nodes for which there are no incoming edges and recursively, level i includes only nodes for which the incoming edges come from nodes at level $i-1$ or smaller, and at least one of them comes from level $i-1$.

We assign the weights of the nodes at level i to i and assign the edge weights w_{km} to the difference between the weight of node k to that of node m. This setting satisfies the constraints.

7.4 From relief interpretation to reconstruction

Given a base, a set of curves, and the heights of their step edges that were computed in Section 7.3, the goal is to reconstruct the relief surface. The output of the algorithm is a height function defined for every point on the base, yielding the resulting surface.

The algorithm should address two of the challenges specified in the introduction, namely the sparsity of the input curves and the interactions between close curves. The sparsity challenge is approached by producing a smooth interpolation in regions in which curves do not exist. The interaction challenge requires that the reconstructed surface will not contain undesired artifacts due to interactions between close curves.

To achieve these goals, we require that the reconstructed height function satisfy two constraints. Locally, the relief in the curve's neighborhood is defined up to a constant using only the step edges of the margins. The constant is important since a relief can be defined either on the base or on another relief, yet the height function is measured relative to the base. Globally, the relief should be the smoothest function that coincides with the relief obtained locally for every curve. During the reconstruction, the algorithm should compute for each curve its constant. Hereafter, we describe our algorithm for realizing these two requirements.

1. Computing the relief locally: Given a curve and its step edge, the goal is to compute its relief locally, independently of other curves. Let $\mathbf{p} = (\nu, \upsilon)$ be a point in the curve's neighborhood (defined by the step edge's width), where υ is the arc-length parameter along the curve of its closest point on the curve and ν is the signed distance from it to $\mathbf{p}$ (0 for a point on the curve). The relief $r(\mathbf{p}) = r(\nu, \upsilon)$ at point $\mathbf{p}$ is set to:

$$r(\mathbf{p}) = p(\nu, \upsilon) + C_{\mathbf{p}}, \tag{7.3}$$

where $p(\mathbf{p})$ is the corresponding step edge and $C_{\mathbf{p}}$ is the height of the step edge with respect to the base.

2. Computing the relief model: Given a base and the height function $r(\mathbf{p})$ computed locally for each curve, our goal is to compute the global height function (relief) R on the whole surface. This function should coincide with r in the neighborhoods of the curves (boundary conditions) and be smooth elsewhere.

The key idea is to reconstruct the surface in two linear steps. First, the smoothest Laplacian of the desired function R is estimated, such that it satisfies the boundary conditions. Then, it is used to calculate R. The linearity of these stages allows us to deal with complex line drawings efficiently.

COMPUTING THE LAPLACIAN OF THE RELIEF: To formulate the smoothness requirement, we demand that the Laplacian of the Laplacian of the relief is zero. Intuitively, it is roughly equivalent to the requirement that the mean curvature of the surface changes linearly. Note that if we required only a zero Laplacian, the reconstructed surface would be planar in between the curves, which is undesirable. Consider, for example, Figure 7.2. The non-planar reconstruction of the surface in between the curves in Figure 7.2(b) is the result of this requirement.

Let us denote by Δ the Laplacian operator on a scalar function. We are seeking the Laplacian L of R, which is a scalar function defined on the base surface. L should be equal to the Laplacian of r in the neighborhoods of the curves and its Laplacian $\Delta(L)$ should be zero elsewhere. Formally, L is the solution of the following system of linear equations.

We assume that the surface is represented by a triangulated mesh. Let $\mathbf{p}$ be a vertex of the mesh. We search for L that satisfies:

$$\begin{aligned} L(\mathbf{p}) &= \Delta r(\mathbf{p}), \quad \mathbf{p} \in \text{curve's neighborhood}, \\ \Delta L(\mathbf{p}) &= 0, \quad \text{elsewhere.} \end{aligned} \tag{7.4}$$

Assume that $N(\mathbf{p})$ is the set of the neighbors of $\mathbf{p}$, A is the area of the Voronoi cell of $\mathbf{p}$, and γ_j and δ_j are the angles opposite the edge $[\mathbf{p}, \mathbf{p}_j]$ of the triangles adjacent to this edge.

The Laplacian Δ of a scalar function f (either r or L) at point $\mathbf{p}$ on a mesh is calculated as [89]:

$$\Delta f(\mathbf{p}) = \frac{1}{2A} \sum_{j \in N(\mathbf{p})} (\cot(\gamma_j) + \cot(\delta_j))(f(\mathbf{p}) - f(\mathbf{p}_j)), \tag{7.5}$$

where $N(\mathbf{p})$ is the set of neighbors of $\mathbf{p}$ on the mesh, A is the Voronoi area of $\mathbf{p}$, and γ_j and δ_j are angles opposite to $\mathbf{p}\mathbf{p}_j$. See Figure 7.7 for clarification of notations. Obviously, Equation (7.5) is linear in the values of f. Hence, any linear-equation solver can be utilized to calculate f.

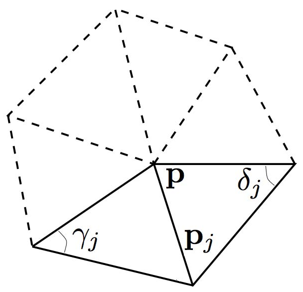


Figure 7.7: **Notations of Eq.5.** The Laplacian Δ of a scalar function f at point $\mathbf{p}$ on a mesh is a linear combination of the values of f on the neighbors $\mathbf{p}_j$ of $\mathbf{p}$.

To solve the system of Equations (7.4), we need to know the values of r. Recall however that by Equation (7.3), r is known only up to constants $C_{\mathbf{p}}$. The trick used here is that we can compute the Laplacian of r without knowing them, since these constants add a linear component to the function and the Laplacian is independent of the linear components.

COMPUTING THE RELIEF: Given the Laplacian of the relief, we calculate the relief (height function) R on the whole base. In the neighborhoods of the curves R should coincide with r, whereas elsewhere the Laplacian ΔR should be equal to the computed Laplacian L. Hence, R is the solution of the following system of linear equations:

$$\begin{aligned} R(\mathbf{p}) &= r(\mathbf{p}), \quad \mathbf{p} \in \text{A curve's neighborhood}, \\ \Delta R(\mathbf{p}) &= L(\mathbf{p}), \quad \text{elsewhere}. \end{aligned} \tag{7.6}$$

In this system of equations, the unknowns are the values of R at every vertex $\mathbf{p}$ and the constants $C_{\mathbf{p}}$. This system of equations is similar to that of Equation (7.4) and thus it is solved in a similar fashion. A similar approach is used for mesh fairing in [114] and [95].

Dealing with curve interactions: The problem of curve interaction manifests itself when a point belongs to several curve neighborhoods. Such a point will therefore appear in several equations in the systems of Equations (7.4) and (7.6). Each such equation is weighted according to the relative distance of the point from the corresponding curve. There are several ways to express the requirement that the surface near the curve coincides with the profile. It was determined empirically that the method we use avoids creating spurious artifacts on the object in cases of curve interactions.

7.5 Base estimation

This section addresses the generation of the base model which will then be given as an input to the reconstruction algorithm. One way to generate the base is to use a professional modeling tool or a sketch-based tool, such as Teddy [51] or FiberMesh [95]. While these systems produce pretty results, they require some expertise, they are interactive, and they can produce relatively simple models.

We, however, would like to estimate the base from the drawing itself. We propose to find a similar base in a database of models, which enables us to reconstruct objects having highly complex bases, when the database contains similar objects. However, since the database is not guaranteed to contain a model that accurately fits the line drawing's outline, the retrieved base has to be modified accordingly. Therefore, our algorithm consists of two stages. First, given the drawings, it finds the most similar model in the database. Then, the model is deformed so as to obtain a base whose orthographic projections are very close to the drawings, while preserving the shape of the matched model. Figure 7.8 summarizes the steps of our base estimation algorithm. Given a drawing (Figure 7.8(a)) and a database (Figure 7.8(b)), the most similar model is retrieved (Figure 7.8(c)). The result of the deformation is shown in Figure 7.8(d).

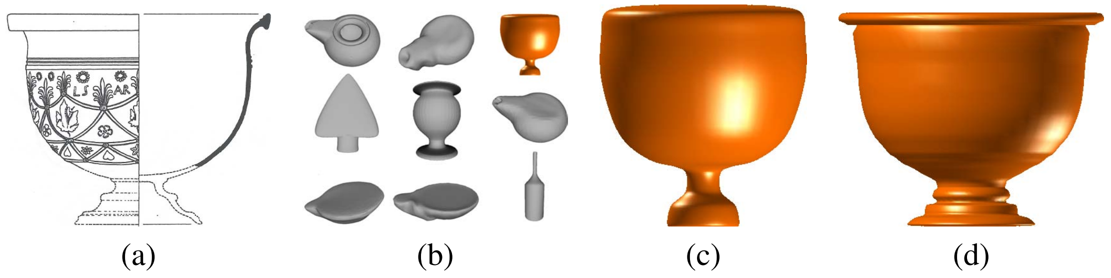


Figure 7.8: **Base estimation algorithm.** Given a line drawing (a) and a database (b), our algorithm first retrieves the most similar model from the database (c) and then deforms it to better suit the line drawing (d).

Reconstruction of a 3D object based on two or three orthographic views, as given in the archeological report, is an ambiguous task. For most artifacts this input is insufficient to produce an accurate base for a 3D model. We believe that our approach of deforming the most similar model from the database resolves the ambiguity in most of the cases when the variability of the base models is not very large.

1. Database search: We use a database of nearly 2000 objects, which consists of three parts:

- 1850 models from different classes, for more details see [79].
- 11 scanned models of archaeological artifacts from the Technion's CG&M Lab repository.
- 35 models of lamps and vases created using various modeling tools.

In the future we plan to enlarge the database of scanned models of archaeological artifacts.

During pre-processing, the models in the database are normalized to align along the principal directions [32]. For each orthographic view (three views) of each object in the database, the silhouette is extracted and sampled. Then a feature vector is calculated as follows. The silhouette is enclosed within a predefined shape (e.g., a circle, a rectangle, etc.) and the feature vector includes the distances from the points on the silhouette to the closest points on the pre-defined shape. Essentially, this vector defines a deformation between 2D shapes – the silhouette and the pre-defined shape. In our implementation we always use a circle as a pre-defined shape.

Given a drawing, the boundary of the drawing (the silhouette of the drawn object) is extracted. The drawing is manually divided into separate images, where each image contains a single view. Silhouette extraction of each image is performed using the active contour model [60]. A feature vector of the boundary is computed as described above. It is compared to those stored in the database using the $L2$ distance. If the drawing contains multiple views, the similarity scores of the views are averaged. The most similar model is returned.

Other methods were proposed for solving this and used for modeling [77, 117, 144]. We found that our matching criterion minimizes the required deformation that has to be applied in the subsequent stage.

2. Model deformation: It is seldom the case that the database contains a base model that perfectly fits the drawing. Therefore, we deform the retrieved model to better match the drawing. The key idea of our algorithm is to move the points on the orthographic projections to their matched points on the drawing and move the other points of the model so as to preserve its shape.

The input to our deformation algorithm is a 3D model and 2D drawings and the output is a deformed 3D model. Let $V = \{\mathbf{v}_1, \cdots, \mathbf{v}_n, \mathbf{v}_i \in \mathbb{R}^3\}$ be the set of the vertices of the input model.

The algorithm, which is illustrated in Figure 7.9, consists of three steps. First, we use orthographic projection to find the model's silhouettes corresponding to the directions of the silhouettes in the drawing. Each silhouette of the model consists of edges and vertices. For a given silhouette, we denote the set of the silhouette vertices by $V_s \subset V$.

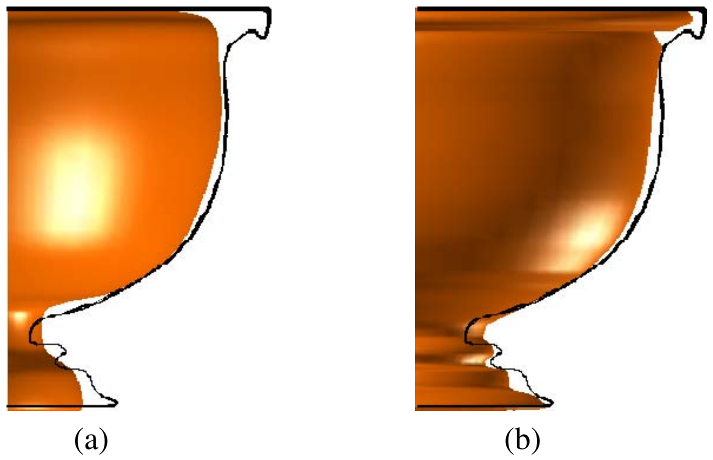


Figure 7.9: **Model deformation.** (a) The desired locations V_s' (the black contour) are computed for all the vertices in V_s (the orange outline). (b) The vertices of V_s move closer to their corresponding vertices in V_s' (the black contour), while preserving the shape, resulting in $\overline{V}$ (the orange outline).

Second, we compute for each vertex $v_i \in V_s$ its desired location in 3D, denoted by V_s'. We find for each vertex in V_s the closest point on the drawing's boundary. Then, the coordinates of each vertex in V_s' are defined such that their projection equals the matched vertex in the drawing, while maintaining the original depth of the vertex. This step is performed for the points of all the silhouettes.

Third, we formulate our problem as an optimization problem whose solution yields the deformed model $\overline{V}$. We require that in this model, every vertex in V_s moves close to its corresponding vertex in V_s', while preserving the shape as much as possible. This is done by minimizing the difference between the Laplacian coordinates of the given model and those of the deformed model.

Specifically, we define the Laplacian coordinates $L(v_i)$ as:

$$L(v_i) = v_i - \frac{1}{|N_i|} \sum_{j \in N_i} v_j, \tag{7.7}$$

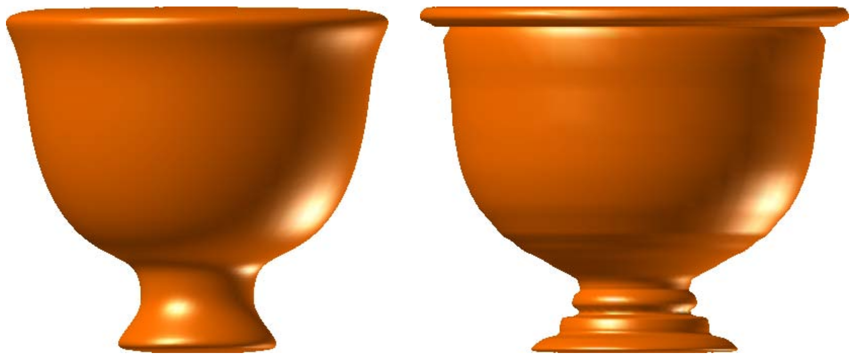
where N_i is the set of v_i's neighbors. As indicated in [119], the Laplacian coordinates are an intrinsic representation of a surface, and thus their preservation leads to preservation of the local surface structure. Moreover, they are a linear function of the object coordinates, which allows efficient computation within the optimization process.

Combining our two requirements – similarity to the line drawing and shape preservation – into a single optimization framework yields:

$$\overline{V} = \arg\min \big(\sum_{v'_i \in V'_s} (\overline{v}_i - v'_i)^2 + \alpha \sum_{v_j \in V} (L(\overline{v}_j) - L(v_j))^2 \big). \tag{7.8}$$

The parameter α controls the importance of preserving the shape with respect to the importance of matching the drawing boundaries. In our experiments, $\alpha = 1$ yielded the best results. This optimization problem is solved using a sparse least-squares solver (we use sparse Cholesky factorization). Note that due to the shape preservation requirement, not all the vertices of V_s move to their corresponding locations, V'_s, as illustrated in Figure 7.9(b).

Our base estimation algorithm is very robust and produces reasonable models even when the object found in the database is highly dissimilar to the required result. Figure 7.10(a) demonstrates this fact, by starting the deformation from a cylinder and yielding a goblet for the drawing in Figure 7.8(a). Naturally, the more similar the retrieved object is, the better the quality of the produced base is, as shown in Figure 7.10(b).

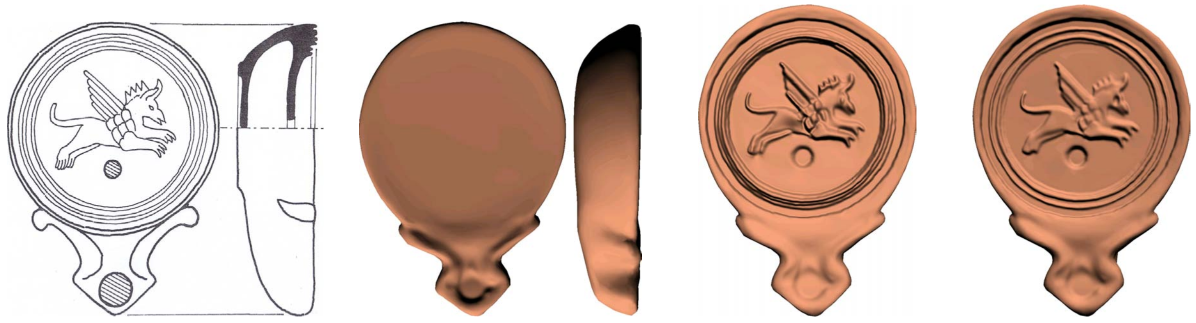


(a) from a cylinder (b) from Figure 7.8(c)

Figure 7.10: **Comparison of the resulting deformed object** The retrieved database model is a cylinder (a) or a more suitable one (b), which is shown in Figure 7.8(c).

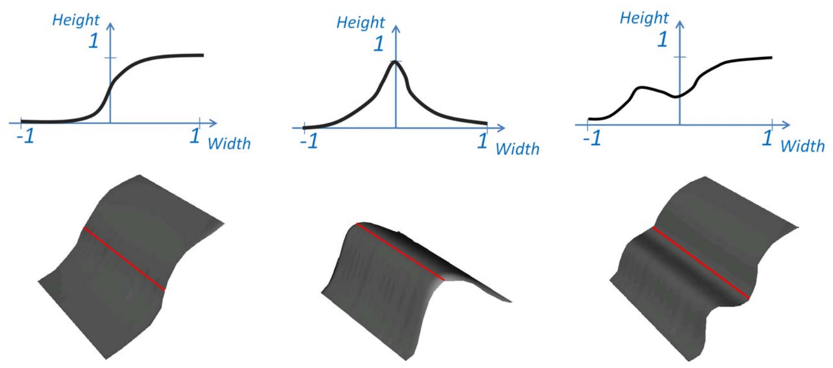
7.6 System

We built an interactive system that realizes our algorithm. Figure 7.11 portrays the pipeline of the system. Initially, the algorithm estimates the underlying base from the drawing, as described in Section 7.5. Next, given the drawing and the base, the relief is reconstructed (Sections 7.2-7.4). Finally, the user may fine-tune the reconstructed surface.



(a) Input – a line drawing (b) Estimated base (c) Automatic output (d) Final result

Figure 7.11: **Algorithm stages.** Given a line drawing (a), our algorithm first estimates the base (b) and then reconstructs the relief (c). The user can then modify the profiles or add new ones, in order to fine-tune the result (d). Here the user changed the relative heights of the wing's feathers and the shape of the armor on the back of the Pegasus. He also switched the direction of the curve of the eye, transforming it from a protrusion to an indentation. (Roman triangular nozzle lamp of a griffin, 25/20 B.C. [34] catalog number 378)



(a) Step (b) Ridge (c) Special

Figure 7.12: **Examples of profile shapes.** Top: profiles; bottom: corresponding 3D reliefs with the line-drawing curves in red.

User interaction: The automatically reconstructed surface is usually satisfactory. However, in some cases, it is desirable to give the user the ability to fine-tune the resulting surface. It may happen when the user has information regarding the geometry of the surface that our algorithm cannot deduce from the drawing, or when our assumption that the surface can be accurately portrayed by a step edge in the line's neighborhood, is not suitable. Therefore, we provide the user with three interaction options, which are easily executed with a single mouse click:

1. A curve's cross section can be depicted with any free-form profile instead of a step edge. Figure 7.12 portrays several possible profiles.
2. The weight and the direction of the step edge (or any other profile) can be set by the user (Constraints 3 and 4 of the interpretation algorithm in Section 7.3).
3. The user can set an arbitrary number of profiles per curve, enabling the shape of the surface to change along the curve.

Implementation and running times: Our algorithm consists of three parts: the base estimation, the line drawing interpretation and the relief reconstruction. The heaviest one, implementation vise, is the relief reconstruction, which is therefore was implemented on GPU. Hereafter, we discuss the implementation and the running times for each part on a

2.4GHz Intel Core 2 Duo processor with 2GB of memory.

The first two parts of the algorithm were implemented in C++. During the generation of the base surface, searching a database that contains almost 2000 models takes 0.3 seconds. The deformation takes less than a second for a 10K-face model and 9 seconds for a 100K model. The running time of the line drawing interpretation is negligible, since it has a linear asymptotic complexity.

Relief reconstruction, however, may provide a challenge for large models, since the large and sparse linear systems in Equations 7.4 and 7.6 need to be solved by iterative optimization methods. To overcome this problem, we took two measures. First, we employ the results of the line drawing interpretation to initialize the optimization procedure. Second, we implemented the Biconjugate Gradient optimization algorithm on the GPU using the cusp library [22]. The running times of the algorithm are in the range of 15 seconds for an 160K-face model (Figure 7.18) to 200 seconds for a model with almost 2M faces (Figure 7.13).

Table 7.1 summarizes the running times.

Fig.	**Num. of curves**	**Preprocessing**	**Automatic**	**Manual**
1	571	5 min	2 min	
11	56	5 min	1 min	1.5 min
13	1300	15 min	3.5 min	
14	39	5 min	30 sec	
15	41	5 min	32 sec	
16	69	5 min	1 min	
17	46	5 min	40 sec	2 min
18	65	5 min	45 sec	2 min

Table 7.1: Time required to pre-process and edit each model. The time for manual editing is shown only where manual editing was applied.

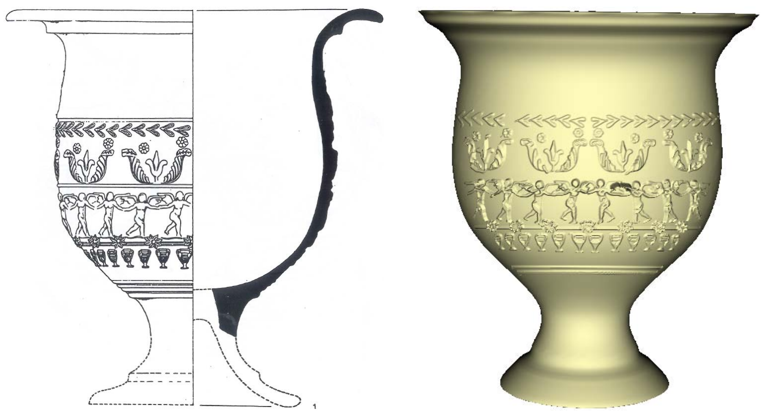


Figure 7.13: **Reconstruction of a Hellenistic relief large light-brown krater** The bowl is found in V. Mrdakovica-Croatia, [13] catalogue number 1. The drawing consists of 1300 tightly interconnected lines.

7.7 Results

We will now present several additional examples of reconstruction of reliefs of archaeological artifacts, demonstrating the ability of our algorithm to deal with complex line drawings. All the archaeological drawings appearing in this section were taken from [13, 34].

Figures 7.1 and 7.13 show the automatic reconstruction of intricate reliefs of a cup and a krater. Though these drawings consist of 571 & 1300 tightly interconnected lines, the reconstruction achieves visually-pleasing results. The speed-up procedure described above enables the reconstruction of these objects in a reasonable time.

Figure 7.1(c) zooms in on the fine details of the reconstructed relief, showing for example, that the reliefs are indeed of different heights. Note that manual reconstruction techniques, such as [142], would require the user to provide parameters for each of the 571 or 1300 lines one after the other, which would be extremely labor intensive.

Figure 7.14 demonstrates the reconstruction of Roman oil lamp. The drawing contain 39 lines. It can be seen that our automatic reconstruction is quite good. Figure 7.15 demonstrates the automatic reconstruction of the base surface and the relief from the drawing of a triangular heat shield of the oil lamp.

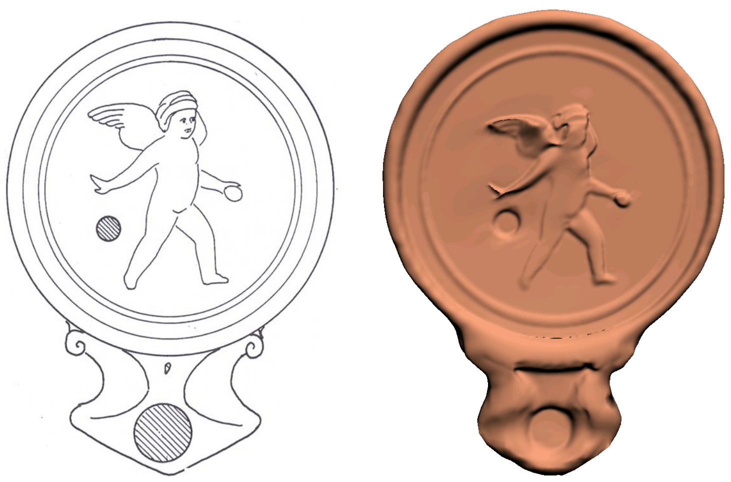


Figure 7.14: **Reconstruction of a Roman triangular nozzle lamp of Eros, A.D. 50-ca 100.** [34] catalogue number 504.

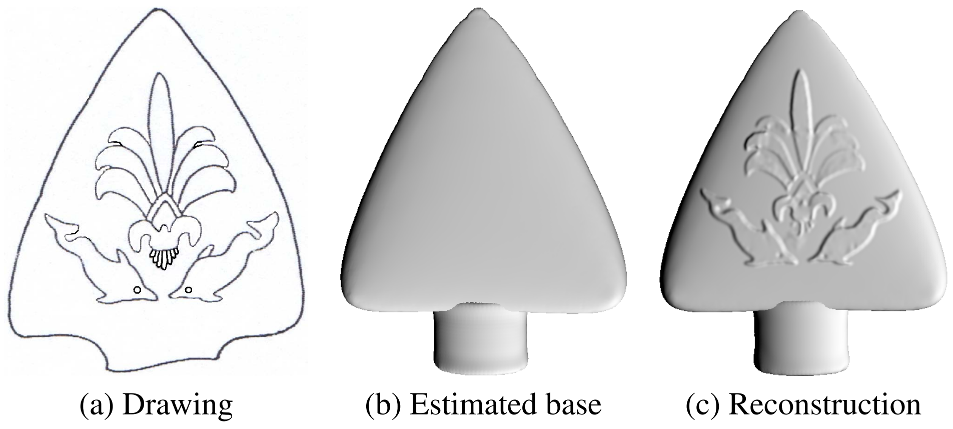


Figure 7.15: **Reconstruction of a Roman Triangular heat shield of a palmette flanked at the base by a pair of dolphins A.D. 50-ca 100.** [34] catalogue number 723.

Figure 7.16 demonstrates the automatic reconstruction of the base surface and the relief from the drawing of Hellenistic relief large bowl. Figure 7.17 shows the reconstruction of a Roman triangular nozzle lamp of a horse. In this case prominent shading was used to enhance the 3D features of the reconstructed object.

Figure 7.18 demonstrates the user's fine-tuning. Here, the user added to two of the curves extra step edges and modified the height of other two step edges. These changes enable height changes along a curve and improve the quality of the bird's tail and wing. Also, the

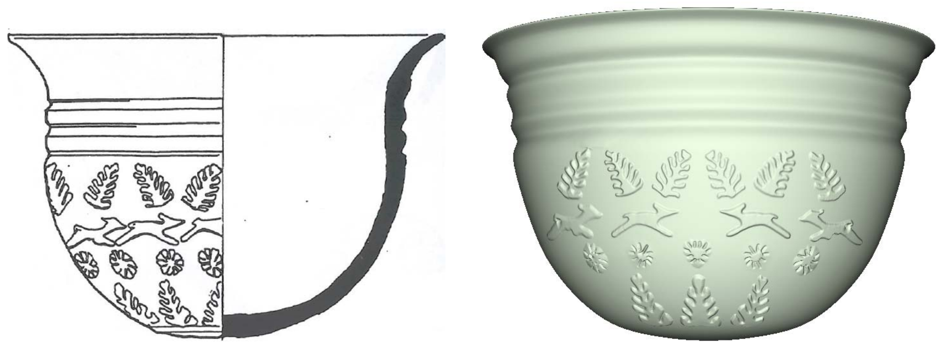


Figure 7.16: **Reconstruction of a Hellenistic relief large bowl** The bowl is found in Resnik, Siculi [13] catalogue number A197.

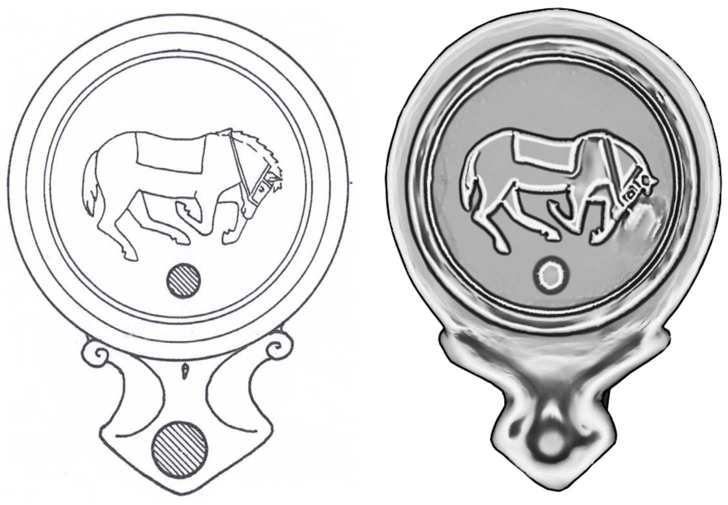


Figure 7.17: **Roman triangular nozzle lamp of a horse, A.D 50-ca. 100.** [34] catalogue number 483.

drawing includes ridges (the bird's legs) and valleys (the bird's eye and the centers of the leaves), which are not supported by the automatic algorithm and were specified by the user.

As opposed to step edges, ridges and valleys have only local influence over the height of the relief. Adding a ridge (or a valley) changes the height only within the margins of the ridge. Thus, these curves are easier to deal with and are omitted from our line interpretation algorithm. They are however dealt with in the relief reconstruction stage.

Limitations: Our approach of base estimation cannot change the topology of the models

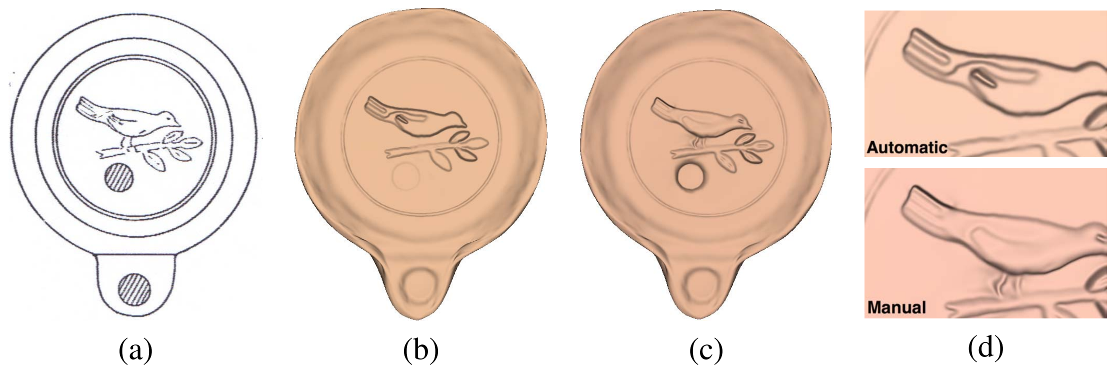


Figure 7.18: **An example of manual fine-tuning of the results.** The original drawing (a) includes ridges (the bird's legs) and valleys (the bird's eye and leaves). Automatic reconstruction (b) is enhanced by several simple manual operations to produce (c). Zoom-in (d) on the automatic and the manual reconstructions reveals that the automatically-obtained surface is less accurate. (Roman fat lamp of a bird on a spray of leaves, 25/20 B.C. - A.D. 40/45. [34] catalog number 775)

found in the database. For example, it fails to produce an accurate reconstruction of bases for oil lamps with a handle, when the most similar lamp from the database does not have a handle. Fortunately, in our experiments in most of the cases the database search returned a model with a handle.

A second limitation concerns our relief construction method. As seen in Figure 7.16, reliefs that are only partially visible (those that intersect the silhouettes) cannot be reconstructed. This is so, since obviously, the invisible part of the relief cannot be utilized.

7.8 Conclusions

In this paper we addressed the problem of automatic reconstruction of a relief object from a line drawing. Based on the observation that relief objects are composed of a smooth base and relief details, we focused on two sub-problems: estimation of a smooth base from the silhouette and reconstruction of the relief on top of the base.

We propose a data-driven algorithm for generating the base from the outline of the given line drawing. The base is constructed irrespective of the details, by utilizing database search and deforming the retrieved most-similar object. This allows us to deal with objects that have complex bases.

While reconstructing reliefs from complex line drawings, we identified four challenges that have to be tackled: the sparsity of the lines, the ambiguity of the line drawing, the large number of strokes comprising the line drawing, and the interactions between close curves. A novel algorithm was proposed that addresses these challenges. It consists of two parts. First, the line drawing is interpreted, solving the challenges of ambiguity and input size, without requiring the user to manually specify the complex line drawing interpretation. Second, given the base and the interpretation, the relief object is reconstructed, addressing the challenges of sparsity and close-curve interaction.

The algorithm was implemented and tested on real complex archaeological illustrations. This lets the archaeologists reconstruct the shapes of artifacts for which, in many cases, the line drawings are the only remaining evidence.

Acknowledgements: This research was supported in part by the Israel Science Foundation (ISF) 1420/12, the Ollendorff Foundation, the P. and E. Nathan Research Fund, and the Argentinian Research Fund. We thank the Laboratory of Computer Graphics & Multimedia at the Technion and the Zinman Institute of Archaeology, University of Haifa, for the scanned models of the archaeological artifacts.

Chapter 8

Conclusions

This thesis was concerned with the definition, detection, and analysis of curves on surfaces. Specifically, the contribution of the thesis was fivefold.

First, we defined two novel types of curves, *demarcating curves* and *relief edges*. They were designed to portray features that resemble 3D step edges. Our experiments showed that these curves manage to capture the features of objects with reliefs more accurately than existing curves.

Second, we presented a framework for automatic estimation of the optimal scale for curve detection on surfaces. This framework can be applied to any type of curve. We demonstrated that on objects that contain features of various scales, The curves obtained by our method outperformed those computed by single-scale algorithms. For objects that contain primarily features of a single scale, the benefit of our algorithm is that it does not require manual tuning of the scale parameter.

Third, we defined a novel vector field on surfaces, termed the *prominent field*, which is a smooth direction field perpendicular to the feature curves. We showed the applicability of the prominent field for surface enhancement and for artificial surface coloring, which emphasizes the object's features. In both cases, we demonstrated that our results outperform the results obtained by previous methods.

Fourth, we addressed the problem of automatic reconstruction of a relief object from a line drawing. The inter-dependencies between the strokes of the line drawing were exploited for automatically generating a good interpretation of the drawing. Then, given an interpretation, we showed how to reconstruct a consistent surface. The algorithm was implemented and tested on real complex archaeological illustrations.

Last, but not least, we successfully applied our algorithms to archaeological objects. We designed a special software in collaboration with archaeologists. The software has been thoroughly tested by many users and obtained positive responses.

8.1 Future work

Our work can be extended in two different directions. The first direction is improving of the definitions and the computation methods of relief edges. Though usually relief edges capture surface features accurately, there are certain cases when they might fail. We wish to handle these cases. The second direction is to utilize our curves for additional applications. We elaborate below.

Machine learning for accurate separation of true and false curves: Our experiments show that relief edges might miss a true feature or create a spurious one (see Chapters 3-7). We suggest to learn the characteristics of correct and incorrect edges, in order to enhance our ability to distinguish between them.

Tools for manual enhancement of relief edges: While our edge detection can work automatically, it was designed for human users. Thus, we would like to provide the users with tools that will help them to improve the automatic results. We want to develop methods that will allow the user to change the location of a curve and to fill "holes" in it. To change the location, the user will manually provide the new location and the algorithm will optimize it to fit both the user's desire and the surface geometry. To fill holes, we can adapt existing methods for curve completion [44].

Shape matching with curves: We would like to check the ability of different types of curves to aid in matching and retrieval of 3D objects. where curves are seldom used. Most algorithms employ point-based and whole surface descriptors. However, we believe that curves can be successfully utilized for these tasks. In Chapter 7 we demonstrated that curves are sufficient for shape reconstruction. This means that curves include all the information required to build or describe an object. Since shape matching is based on shape descriptor, we think that curves carry enough information for matching.

Bibliography

[1] Polygon-technology GmbH. http://www.polymetric.com/.

[2] C.L. Bajaj and G. Xu. Anisotropic diffusion of surfaces and functions on surfaces. *ACM Transactions on Graphics (TOG)*, 22(1):4–32, 2003.

[3] S. Baker, S.K. Nayar, and H. Murase. Parametric feature detection. *Int. J. of Comp. Vis.*, 27(1):27–50, 1998.

[4] A. Bartesaghi and G. Sapiro. A system for the generation of curves on 3D brain images. *Human Brain Mapping*, 14(1):1–15, 2001.

[5] A. Bartoli and P. Sturm. The 3D line motion matrix and alignment of line reconstructions. *Int. J. of Comp. Vis.*, 57(3):159–178, 2004.

[6] Mitra Basu. Gaussian-based edge-detection methods-a survey. *IEEE Trans. on Systems, Man, and Cybernetics*, 32(3):252–260, 2002.

[7] H. Bay, V. Ferraris, and L. Van Gool. Wide-baseline stereo matching with line segments. *IEEE Conf. on Comp. Vis. and Patt. Rec.*, 1:329 – 336, 2005.

[8] A. Belyaev and E. Anoshkina. Detection of surface creases in range data. *Mathematics of Surfaces XI*, pages 50–61, 2005.

[9] A.G. Belyaev, A.A. Pasko, and T.L. Kunii. Ridges and ravines on implicit surfaces. *Comp. Graph. Intern*, pages 530–535, 1998.

[10] S.T. Birchfield and S.J. Pundlik. Joint tracking of features and edges. pages 1–8, 2008.

[11] M. Bokeloh, A. Berner, M. Wand, H.P. Seidel, and A. Schilling. Symmetry detection using feature lines. *EuroGraphics*, 28(2):697–706, 2009.

[12] B. Brown, C. Toler-Franklin, D. Nehab, M. Burns, D. Dobkin, A. Vlachopoulos, C. Doumas, S. Rusinkiewicz, and T. Weyrich. A system for high-volume acquisition and matching of fresco fragments: Reassembling theran wall paintings. In *ACM Transactions on Graphics (TOG)*, volume 27, pages 84–93. ACM, 2008.

[13] Z. Brusic. *Hellenistic and Roman relief pottery in Liburnia (North-East Adriatic, Croatia)*. British Archaeological reports (BAR) International Series 817, 1999.

[14] J. Canny. A computational approach to edge detection. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 8(6):679–698, 1986.

[15] F. Cazals, M. Pouget, et al. Topology driven algorithms for ridge extraction on meshes. Technical report, INRIA, 2005.

[16] X. Chen, S.B. Kang, Y. Xu, J. Dorsey, and H. Shum. Sketching reality: Realistic interpretation of architectural designs. *ACM Trans. Graph.*, 27(2):370–381, 2008.

[17] M. B. Clowes. On seeing things. *Artif. Intell.*, 2(1):79–116, 1971.

[18] J. Cohen, D. Dunkan, D. Snyder, J Cooper, J. Kumar, S. Hahn, D. Chen, B Purnomo, and J Graettinger. iclay: Digitizing cuneiform. In *Symposium on Virtual Reality, Archaeology and Cultural Heritage*, pages 135–143, 2004.

[19] F. Cole, A. Golovinskiy, A. Limpaecher, H.S. Barros, A. Finkelstein, T. Funkhouser, and S. Rusinkiewicz. Where do people draw lines? *ACM Trans. on Graph. (TOG)*, 27(3):88, 2008.

[20] F. Cole, K. Sanik, D. DeCarlo, A. Finkelstein, T. Funkhouser, S. Rusinkiewicz, and M. Singh. How well do line drawings depict shape? *ACM Trans. Graph. (TOG)*, 28(3):28–35, 2009.

[21] M. Cooper. *Line drawing interpretation.* Springer-Verlag New York Inc, 2008.

[22] CUDA. Cusp - CUDA based sparse linear algebra library. http://code.google.com/p/cusp-library/.

[23] D. DeCarlo, A. Finkelstein, and S. Rusinkiewicz. Interactive rendering of suggestive contours with temporal coherence. *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 15–145, 2004.

[24] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. *ACM Trans. on Graph.*, 22(3):848–855, 2003.

[25] D. DeCarlo and S. Rusinkiewicz. Highlight lines for conveying shape. *Int. symp. on Non-photorealistic animation and rendering*, pages 63–70, 2007.

[26] K. Demarsin, D. Vanderstraeten, T. Volodine, and D. Roose. Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Computer-Aided Design*, 39(4):276–283, 2007.

[27] R. Deriche and O. Faugeras. Tracking line segments. *Image and Vision Computing*, 8(4):261–270, 1990.

[28] M. Desbrun, M. Meyer, P. Schröder, and A.H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. pages 317–324, 1999.

[29] O. Deussen, S. Hiller, C. van Overveld, and T. Strothotte. Floating points: A method for computing stipple drawings. *Computer Graphics Forum*, 19(3):40–51, 2000.

[30] M. P. Do Carmo. *Differential geometry of curves and surfaces.* Prentice-Hall, 1976.

[31] M. Eigensatz, R.W. Sumner, and M. Pauly. Curvature-domain shape processing. *Comp. Graph. Forum*, 27(2):241–250, 2008.

[32] M. Elad, A. Tal, and S. Ar. Content Based Retrieval of VRML objects - an iterative and interactive approach. *EG Multimedia, September*, pages 97–108, 2001.

[33] M. Fisher, P. Schroder, M. Desbrun, and H. Hoppe. Design of tangent vector fields. *ACM Transactions on Graphics (TOG)*, 26(3):56, 2007.

[34] C. Fitch and N. Goldman. *Cosa, the lamps*. University of Michigan Press, 1994.

[35] B.P. Flannery, W.H. Press, S.A. Teukolsky, and W. Vetterling. *Numerical recipes in C*. Cambridge Univ Press, 1992.

[36] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. *ACM Trans. on Graph.*, 22(3):950–953, 2003.

[37] D. A. Forsyth and J. Ponce. *Computer Vision – A Modern Approach*. Prentice-Hall, 2002.

[38] Y. Gingold, T. Igarashi, and D. Zorin. Structured annotations for 2D-to-3D modeling. In *ACM Transactions on Graphics (TOG)*, volume 28, pages 148–157. ACM, 2009.

[39] Y. Gingold and D. Zorin. Shading-based surface editing. *ACM Trans. Graph. (TOG)*, 27(3):95–101, 2008.

[40] B. Gooch, P. J. Sloan, A. Gooch, P. Shirley, and R. F. Riesenfeld. Interactive technical illustration. In *Symp. on Inter. 3D Graph.*, pages 31–38, 1999.

[41] A. Guéziec and N. Ayache. Smoothing and matching of 3D space curves. *Int. J. of Comp. Vis.*, 12(1):79–104, 1994.

[42] E. Hameiri and I. Shimshoni. Estimating the principal curvatures and the Darboux frame from real 3D range data. *IEEE SMC B*, 33(4):626–637, August 2003.

[43] Robert M Haralick. Digital step edges from zero crossing of second directional derivatives. *Pattern Analysis and Machine Intelligence*, 23(1):58–68, 1984.

[44] G. Harary and A. Tal. 3D Euler spirals for 3D curve completion. *Computational Geometry: Theory and Applications*, 45(3):115–126, 2012.

[45] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 517–526, 2000.

[46] K. Hildebrandt, K. Polthier, and M. Wardetzky. Smooth feature lines on surface meshes. *SGP*, 5:85–90, 2005.

[47] T. Hou and H. Qin. Efficient computation of scale-space features for deformable shape correspondences. In *ECCV*, pages 384–397. Springer, 2010.

[48] J. Hua, Z. Lai, M. Dong, X. Gu, and H. Qin. Geodesic distance-weighted shape vector image diffusion. *IEEE Trans. on Vis. and Comp. Graph.*, 14(6):1643–1650, 2008.

[49] A. Hubeli and M. Gross. Multiresolution feature extraction for unstructured meshes. *Proceedings of the conference on Visualization'01*, pages 287–294, 2001.

[50] DA Huffman. Impossible objects as nonsense sentences. *Computer methods in image analysis*, pages 338–347, 1977.

[51] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3D freeform design. *ACM Transactions on Graphics (TOG)*, 19(3):409 – 416, 1999.

[52] V. Interrante, H. Fuchs, and S. Pizer. Enhancing transparent skin surfaces with ridge and valley lines. In *Proceedings of the 6th conference on Visualization '95*, pages 52–59, Washington, DC, USA, 1995.

[53] L. A. Iverson and S. W. Zucker. Logical/linear operators for image curves. *IEEE Trans. on patt. anal. and mach. intell.*, 17(10):982–996, October 1995.

[54] A.K. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern recognition*, 29(8):1233–1244, 1996.

[55] P. Joshi and N. Carr. Repoussé: Automatic inflation of 2d artwork. In *Proceedings of the Fifth Eurographics conference on Sketch-Based Interfaces and Modeling*, pages 49–55. Eurographics Association, 2008.

[56] T. Judd, F. Durand, and E. Adelson. Apparent ridges for line drawing. *ACM Trans. on Graphics (TOG)*, 26(3):19–es, 2007.

[57] R.D. Kalnins, L. Markosian, B.J. Meier, M.A. Kowalski, J.C. Lee, P.L. Davidson, M. Webb, J.F. Hughes, and A. Finkelstein. WYSIWYG NPR: Drawing Strokes Directly on 3D Models. *ACM Trans. on Graph.*, 21(3):755–762, 2002.

[58] E. Kalogerakis, D. Nowrouzezahrai, P. Simari, and K. Singh. Extracting lines of curvature from noisy point clouds. *Computer-Aided Design*, 41(4):282–292, 2009.

[59] O.A. Karpenko and J.F. Hughes. SmoothSketch: 3D free-form shapes from complex sketches. *Proceedings of ACM SIGGRAPH 2006*, 25(3):589–598, 2006.

[60] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[61] D. Katsoulas and A. Werber. Edge detection in range images of piled box-like objects. *ICPR*, 2:80–84, 2004.

[62] B. Kerautret, X. Granier, and A. Braquelaire. Intuitive shape modeling by shading design. In *International Symposium on Smart Graphics*, pages 163–174, 2007.

[63] H.S. Kim, H.K. Choi, and K.H. Lee. Feature detection of triangular meshes based on tensor voting theory. *Computer-Aided Design*, 41(1):47–58, 2009.

[64] R. Kimmel and A. Bruckstein. Regularized laplacian zero crossings as optimal edge integrators. *International Journal of Computer Vision*, 53(3):225–243, 2003.

[65] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller. Curvature-Based Transfer Functions for Direct Volume Rendering: Methods and Applications. In *IEEE Vis.*, pages 67–76, 2003.

[66] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *IEEE Trans. on Vis. and Comp. Graph.*, pages 67–76, 2003.

[67] J.J. Koenderink. What does the occluding contour tell us about solid shape. *Perception*, 13(3):321–330, 1984.

[68] J.J. Koenderink. *Solid shape*. Cambridge Univ Press, 1990.

[69] D. Koller, J. Trimble, T. Najbjerg, N. Gelfand, and M. Levoy. Fragments of the city: Stanford's digital forma urbis romae project. *J. of Roman Arch.*, 61(1):237–252, 2006.

[70] M. Kolomenkin, G. Leifman, I. Shimshoni, and A. Tal. Reconstruction of relief objects from line drawings. In *(CVPR)*, pages 993–1000, 2011.

[71] M. Kolomenkin, I. Shimshoni, and A. Tal. Demarcating curves for shape illustration. *ACM Trans. on Graph., SIGGRAPH Asia*, 27(4), 2008.

[72] M. Kolomenkin, I. Shimshoni, and A. Tal. On edge detection on surfaces. In *CVPR*, pages 2767–2774, 2009.

[73] M. Kolomenkin, I. Shimshoni, and A. Tal. Prominent field for shape processing of archaeological artifacts. *IEEE Workshop on eHeritage and Digital Art Preservation (ICCV)*, 2009.

[74] M. Kolomenkin, I. Shimshoni, and A. Tal. Prominent field for shape processing of archaeological artifacts. *IJCV*, 94(1):89–100, 2011.

[75] J.F. Lalonde, R. Unnikrishnan, N. Vandapel, and M. Hebert. Scale selection for classification of point-sampled 3D surfaces. In *3DIM*, pages 285–292, 2005.

[76] T. Langer, A. Belyaev, and H.P. Seidel. Exact and interpolatory quadratures for curvature tensor estimation. *Comp. Aided Geometric Design*, 24(8-9):443–463, 2007.

[77] S. Lee and T. Funkhouser. Sketch-based search and composition of 3D models. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*, 2008.

[78] Y. Lee, L. Markosian, S. Lee, and J. F. Hughes. Line drawings via abstracted shading. *ACM Transactions on Graphics*, 22(3):19:8 – 19:15, 2007.

[79] G. Leifman, R. Meir, and A. Tal. Semantic-oriented 3d shape retrieval using relevance feedback. *The Visual Computer*, 21(8-10):865–875, 2005.

[80] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *IJCV*, 30(2):117–154, 1998.

[81] J. Liu, L. Cao, Z. Li, and X. Tang. Plane-based optimization for 3D object reconstruction from single line drawings. *PAMI*, 30(2):315–327, 2007.

[82] S. Liu, R.R Martin, F.C Langbein, and P.L Rosin. Background surface estimation for reverse engineering of reliefs. *Int. J. of CAD/CAM*, 7, 2007.

[83] T. Luo, R. Li, and H. Zha. 3D line drawing for archaeological illustration. *International Journal of Computer Vision*, 94(1):23–35, 2011.

[84] L.J.P. Maaten, P.J. Boon, J.J. Paijmans, A.G. Lange, and E.O. Postma. Computer vision and machine learning for archaeology. In *Computer Applications and Quantitative Methods in Archaeology*, pages 112–130, 2006.

[85] J. Malik. Interpreting line drawings of curved objects. *IJCV*, 1(1):73–103, 1987.

[86] D. Marr and E. C. Hildreth. Theory of edge detection. *Proc. of the Royal Society of London*, B(207):187–217, 1980.

[87] P. Meer and B. Georgescu. Edge detection with embedded confidence. *IEEE PAMI*, 23(12):1351–1365, 2001.

[88] E. Mehlum and C. Tarrou. Invariant smoothness measures for surfaces. *Advances in Comp. Math.*, 8(1-2):49–63, 2006.

[89] M. Meyer, M. Desbrun, P. Schroder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. *VisMath*, 3(7):34–57, 2002.

[90] K. Mikolajczyk, A. Zisserman, and C. Schmid. Shape recognition with edge-based features. *British Mach. Vis. Conf.*, 2:779 – 788, 2003.

[91] O. Monga, R. Deriche, G. Malandain, and J. P. Cocquerez. Recursive filtering and edge tracking: two primary tools for 3D edge detection. *IVC*, 9(4):203–214, 1991.

[92] S. Musuvathy, E. Cohen, J. Damon, and J. Seong. Principal curvature ridges and geometrically salient regions of parametric b-spline surfaces. *Computer-Aided Design*, 43(7):756–770, 2011.

[93] S. Musuvathy, E. Cohen, J. Seong, and J. Damon. Tracing ridges on b-spline surfaces. *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, pages 55–66, 2009.

[94] V. S. Nalwa and T. O. Binford. On detecting edges. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 8(6):699–714, 1986.

[95] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Fibermesh: designing freeform surfaces with 3D curves. *ACM Trans. Graph. (TOG)*, 26(3):41–51, 2007.

[96] J. Novatnack and K. Nishino. Scale-dependent 3D geometric features. In *ICCV*, pages 1–8, 2007.

[97] Y. Ohtake, A. Belyaev, and H.P. Seidel. Mesh smoothing by adaptive and anisotropic gaussian filter applied to mesh normals. *Vis., Model., and Visual.*, pages 203–210, 2002.

[98] Y. Ohtake, A. Belyaev, and H.P. Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. on Graph.*, 23(3):609–612, 2004.

[99] D. Page, A. Koschan, M. Abidi, and J. Overhiolt. Ridge-valley path planning for 3D terrains. *IEEE Int. Conf. on Rob. and Auto.*, pages 119–124, 2006.

[100] X. Pang, Z. Song, and W. Xie. Extraction of valley-ridge lines from the point cloud-based 3D fingerprint model. *IEEE Computer Graphics and Applications*.

[101] S. Pankanti, C. Dorai, A.K. Jain, et al. Robust feature detection for 3D object recognition and matching. *Geometric Methods in Computer Vision*, 2031:366–377, 1993.

[102] M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled surfaces. *Comp. Grap. Forum*, 22(3):281–289, 2003.

[103] M. Pauly, L.P. Kobbelt, and M. Gross. Point-based multiscale surface representation. *ACM Transactions on Graphics*, 25(2):177–193, 2006.

[104] D. E. Pearson and J. A. Robinson. Visual communication at very low data rates. *Proceedings of IEEE*, 73:795–812, 1985.

[105] X. Pennec, N. Ayache, and J. Thirion. Landmark-based registration using features identied through differential geometry. In *Handbook of Medical Imaging*, pages 499–513. Academic Press, 2000.

[106] T. Poggio, H. Voorhees, and A. Yuille. A regularized solution to edge detection. *Journal of Complexity*, 4(2):106–123, 1988.

[107] N. Ray, W.C. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic global parameterization. *ACM Transactions on Graphics (TOG)*, 25(4):148–195, 2006.

[108] M. Reuter, F.E. Wolter, and N. Peinecke. Laplace–beltrami spectra as shape-dnaof surfaces and solids. *Computer-Aided Design*, 38(4):342–366, 2006.

[109] C. Rossl, L. Kobbelt, and H. Seidel. Extraction of feature lines on triangulated surfaces using morphological operators. *Proceedings of the AAAI Symposium on Smart Graphics*, pages 71–75, 2000.

[110] H. Rushmeier. Eternal Egypt: experiences and research directions. In *Modeling and Visualization of Cultural Heritage*, pages 22–27, 2005.

[111] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *3D Data Processing, Visualization and Transmission*, pages 486–493, 2004.

[112] S. Rusinkiewicz, M. Burns, and D. DeCarlo. Exaggerated shading for depicting shape and detail. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), July 2006.

[113] T. Saito and T. Takahashi. Comprehensible Rendering of 3-D Shapes. *Computer Graphics*, 24(4), 1990.

[114] R. Schneider and L. Kobbelt. Geometric fairing of irregular meshes for free-form surface design. *Computer aided geometric design*, 18(4):359–379, 2001.

[115] N Senthilkumaran and R Rajesh. Edge detection techniques for image segmentation-a survey of soft computing approaches. *Intern. Journal of recent trends in engineering*, 1(2):250–254, 2009.

[116] I. Shimshoni and J. Ponce. Recovering the shape of polyhedra using line-drawing analysis and complex reflectance models. *Comp. Vis. and Img. Under.*, 65(2):296–310, 1997.

[117] H. Shin and T Igarashi. Magic canvas: interactive design of a 3-D scene prototype from freehand sketches. In *Graphics Interface*, pages 63–70, 2007.

[118] P. Smith, T. Drummond, and R. Cipolla. Layered motion segmentation and depth ordering by tracking edges. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(4):479–494, 2004.

[119] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.P. Seidel. Laplacian surface editing. *Eurographics symposium on Geometry Processing*, pages 184–194, 2004.

[120] J. Stam. Flows on surfaces of arbitrary topology. *ACM Transactions On Graphics (TOG)*, 22(3):724–731, 2003.

[121] C. Steger. An unbiased detector of curvilinear structures. *PAMI*, 20(2):113–125, 1998.

[122] Ephraim Stern. Excavations at dor. Technical report, Institute of Archaeology of the Hebrew University, 1995.

[123] T. Strothotte and S. Schlechtweg. *Non-photorealistic computer graphics: modeling, rendering, and animation.* Morgan Kaufmann, 2002.

[124] G. Stylianou and G. Farin. Crest lines for surface segmentation and flattening. In *IEEE Transactions on Visualization and Computer Graphics*, pages 536–544, 2004.

[125] K. Sugihara. Mathematical structures of line drawings of polyhedrons-toward man-machine communication by means of line drawings. *PAMI*, 3(5):458–469, 1982.

[126] X. Sun, PL Rosin, RR Martin, and FC Langbein. Fast and effective feature-preserving mesh denoising. *IEEE Trans. on Vis. and Comp. Graph.*, 13(5):925–938, 2007.

[127] D. Sỳkora, D. Sedlacek, S. Jinchao, J. Dingliana, and S. Collins. Adding depth to cartoons using sparse depth (in) equalities. *Comp. Grap. Forum*, 29(2):615–623, 2010.

[128] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface smoothing via anisotropic diffusion of normals. *IEEE Visualization, 2002*, pages 125–132, 2002.

[129] G. Taubin. A signal processing approach to fair surface design. *Comp. grap. and interact. techn.*, pages 351–358, 1995.

[130] G. Taubin. Curve and surface smoothing without shrinkage. In *ICCV*, pages 852–860, 1995.

[131] J.T. Todd. The visual perception of 3D shape. *Trends in Cognitive Sciences*, 8(3):115–121, 2004.

[132] C. Toler-Franklin, A. Finkelstein, and S. Rusinkiewicz. Illustration of complex real-world objects using images with normals. In *NPAR-07*, pages 111–119, 2007.

[133] V. Torre and T. Poggio. Differential operators for edge detection. 1983.

[134] V. Torre and T.A. Poggio. On edge detection. *PAMI*, (2):147–163, 1986.

[135] G. Turk. Texture synthesis on surfaces. *ACM Transactions on Graphics (TOG)*, 19(3):347–354, 2001.

[136] M. Vieira and K. Shimada. Surface mesh segmentation and smooth surface extraction through region growing. *Computer aided geometric design*, 22(8):771–792, 2005.

[137] W. von Funck, H. Theisel, and H.P. Seidel. Vector field based shape deformations. *ACM Transactions on Graphics (TOG)*, 25(3):11–25, 2006.

[138] A. Vrubel, O. Bellon, and L. Silva. A 3D Reconstruction Pipeline for Digital Preservation. *CVPR*, pages 2687 – 2694, 2009.

[139] D. Waltz. Understanding line drawings of scenes with shadows. *The Psy. of Comp. Vis.*, pages 19–91, 1975.

[140] Y. Wang, Y. Chen, L. Liu, and X. Tang. 3D reconstruction of curved objects from single 2D line drawings. *CVPR*, 33(1):85–103, 2009.

[141] L.Y. Wei and M. Levoy. Texture synthesis over arbitrary manifold surfaces. *ACM Transactions on Graphics (TOG)*, 19(3):355–360, 2001.

[142] T.P. Wu, C.K. Tang, M.S. Brown, and H.Y. Shum. ShapePalettes: interactive normal transfer via sketching. *ACM Trans. Graph. (TOG)*, 26(3):44–52, 2007.

[143] X. Xuexiang, H. Ying, T. Feng, and S. Hock-Soon. An effective illustrative visualization framework based on photic extremum lines (PELs). *IEEE Trans. on Vis. and Comp. Graph.*, 13(6):1328–1335, 2007.

[144] C Yang, D. Sharon, and M. Van de Panne. Sketch-based modeling of parameterized objects. In *Proceedings of Eurographics workshop on sketch-based interfaces and modeling (SBIM)*, 2005.

[145] S. Yoshizawa, A. Belyaev, and H.P. Seidel. Fast and robust detection of crest lines on meshes. *ACM symp. on solid and physical modeling*, pages 227–232, 2005.

[146] S. Yoshizawa, A. Belyaev, and H.P. Seidel. Smoothing by example: Mesh denoising by averaging with similarity-based weights. *IEEE Int. Conf. on Shape Model. and App.*, pages 9–19, 2006.

[147] S. Yoshizawa, A. Belyaev, H. Yokota, and H.P. Seidel. Fast and faithful geometric algorithm for detecting crest lines on meshes. *Pacific Conference on Computer Graphics*, pages 231–237, 2007.

[148] E. Zhang, K. Mischaikow, and G. Turk. Vector field design on surfaces. *ACM Transactions on Graphics (TOG)*, 25(4):1294–1326, 2006.

[149] L. Zhang, Y. He, X. Xie, and W. Chen. Laplacian lines for real-time shape illustration. *Interactive 3D graphics and games*, pages 129–136, 2009.

[150] D. Ziou and S. Tabbone. Edge detection techniques-an overview. *PATTERN RECOGNITION AND IMAGE ANALYSIS*, 8:537–559, 1998.

[151] S.W. Zucker and R.A. Hummel. A three-dimensional edge operator. *IEEE PAMI*, 3(3):324–331, 1981.

לשחזר אוביקטים בצורה אוטומטית. לבסוף, אנחנו מפעילים אלגוריתמים שלנו על אוביקטים ארכיאולוגים.

לעקמומיות בכיוון הניצב לכיוון התו הקרוב. צביעה זה מגדילה את נגוד הצבעים על התווים וכך מדגישה אותם.

יתר על כן, אנו מציעים שיטה כללית לחישוב אוטומטי של קנה המידה האופטימלי בכל נקודה על פני המשטח. בשיטה זו קנה המידה תלוי בעקומה. בשלב הראשון, קנה המידה האופטימלי מחושב מקומית לכל נקודה על המשטח. לאחר מכן, קני מידה מקומיים משמשים לבניית משטח אחד המכונה "המשטח האופטימלי". על המשטח האופטימלי ניתן להפעיל אלגוריתם מקורי לחיפוש עקומות. השיטה מתאימה לכל סוגי העקומות שאת עוצמתן ניתן לאפיין ע"י עקמומיות ונגזרת של עקמומיות. היא מבטלת צורך בהתערבות המשתמש ומאפשרת לגלות תווים בקני מידה שונים על אותו אוביקט.

בנוסף אנחנו מציעים שיטה לשיפור אוביקטים המבוססת על שדה וקטורי חדשני המכונה *prominent field*. אינטואיטיבית, השדה מוגדר כשדה חלק שניצב לכיוון התווים על המשטח. החלקת האוביקט בכיוון הניצב לכיוון השדה מורידה רעש אך לא פוגעת בתווים.

אחרי גילוי עקומות על משטחים, אנו פונים לבעיה ההפוכה של שחזור משטחים מציורי עקומות. היכולת לשחזר צורה מתוך ציור היא אחת היכולות הבסיסיות של בני אדם ואפילו ילדים קטנים מסוגלים לזהות צורות ולשחזר אותן בראש. לעומת זאת, שחזור אוטומטי מתוך ציור נשאר בעיה קשה עקב כמה סיבות. קודם כל, דלילות העקומות לא מייצרת מספיק אילוצים להגדרת האוביקט. שנית, לעתים תקופות הציורים דו משמעיים מפני שלעקומה יכולות להיות כמה משמעויות גיאומטריות, למשל היא יכולה לייצג אי רציפות תלת ממדית, קמט במשטח או פונקצית מדרגה. שלישית, ציור עלול לכלול כמות גדולה של עקומות שהמשתמש צריך לאפיין. ולבסוף, העקומות תלויות אחת בשניה.

אנו מציעים אלגוריתם המסוגל לשחזור תבליט מתוך ציור המורכב מכמות גדולה של עקומות. האלגוריתם מבוסס על שני רעינות מרכזיים. ראשית, הוא מנצל את התלות בין עקומות בציור על מנת ליצור באופן אוטומטי פרשנות ראשונית ומקורבת של התמונה המקומית. בהינתן הפרשנות ניתן לשחזר את המשטח בצורה מדויקת.

כאפליקציה בתזה אנו מתרכזים בתחום הארכיאולוגיה. החלטתי להתמקד בתחום זה נובעת מכמה סיבות. קודם כל, מנקודת המבט האלגוריתמית, התחום הוא מאתגר ביותר, מפני שחפצים ארכיאולוגים, אחרי שהיה של כמה אלפי שנים מתחת לאדמה, שבורים ושחוקים בשונה מהאוביקטים התלת ממדיים הממודלים ע"י אדם או אוביקטים מודרניים סרוקים. שנית, ארכיאולוגיה מספקת לנו הגדרה ברורה של הצלחה, מפני שהשימושיות של תוצאותינו מאומתת מיידית על ידי ארכיאולוגים. שלישית, המחקר שלנו יכול להאיץ משמעותית את המחקר הארכיאולוגי. דרכי רישום וגישה אל מידע על ממצאים ארכיאולוגים לא השתנו הרבה במאה שנים אחרונות והן מהוות עדיין את אחד מצווארי הבקבוק הגדולים במחקר הארכיאולוגי. הכנסת שיטות מתקדמות מתחומי ראיה ממוחשבת וגרפיקה ממוחשבת יכולה להאיץ ולשפר את המחקר. רביעית, לתחום הזה יש חשיבות רבה משום שניתוח של ממצאים ארכאולוגיים, כגון חותמות, קרמיקה כלי, מטבעות, וכו', הוא מקור עיקרי של הידע שלנו על העבר.

בין אוביקטים ארכיאולוגים בחרנו להתמקד באוביקטים בעלי תבליטים. אוביקטים אלו מורכבים ממשטח חלק ועיטורים שבולטים ממנו או טבולים בו. העיטורים יכולים לייצג כל דבר, מצורות גיאומטריות פשוטות עד לאוביקטים תלת ממדיים שלמים.

לסיכום, לתזה שלנו חמש תרומות מרכזיות. א', אנו מציעים סוג חדש של עקומות הנקראות עקומות תבליטים. עקומות תבליטים מתארות מקומות דמויי מדרגה תלת ממדית יותר מדויק מעקומות אחרות. ב', אנו מציעים שיטה כללית לגילוי עקומות בסקלה אופטימלית. ניתן להפעיל את השיטה הזאת על כל סוגי העקומות. ג', אנו מגדירים שדה וקטורי חדשני ומוכיחים שהוא שימושי בסינון והדמיה של אוביקטים. ד', אנו מראים איך לשחזר אוביקט מתוך ציור עקומות. האלגוריתם יכול

עקומות על משטחים עם אפליקציות בארכיאולוגיה

תקציר

מראשיתה של אנושות עקומות שימשו אותנו בהעברת מידע ויזואלי. מציורי קיר וקעקועים, דרך שיטות כתב ראשוניות ומפות, ועד לציורים הנדסיים ויצירות אומנות – עקומות ליוו אותנו ועזרו לנו להתקשר. גם היום עקומות ממשיכות לשחק תפקיד חשוב בהנדסה, הדמיה ואומנות.

לעקומות ישנן מספר תכונות ההופכות אותן למאוד אטרקטיביות. ראשית, מאוד קל וזול לייצר אותן. אין צורך בחומרים מיוחדים והכנות מיוחדות. שנית, המשמעות של ציור מורכב מעקומות אינטואיטיבית וניתנת להבנה ללא קשר להשכלה או לרקע תרבותי. שלישית, העקומות דלילות, או במילים אחרות כמות קטנה של עקומות מספיקה לתיאור של צורות מורכבות.

הדלילות של עקומות הופכת אותם לאהודות לא רק על בני אדם אלא גם על מחשבים. מידע דליל דורש פחות זיכרון ועוזר לפתח אלגוריתמים מהירים יותר שיכולים לרוץ על חומרה פשוטה יותר. כמות גדולה של אפליקציות בגרפיקה וראיה ממוחשבת שמשתמשות בעקומות רק מדגישה את העובדה הזאת. ישנן אפליקציות רבות בסגמנטציה, עקיבה, חיפוש, ציור לא ריאליסטי, ניווט ותחומים אחרים. האפליקציות האלה בדרך כלל משתמשות בעקומות על מנת לתאר תווים (features) – תבניות בולטות בתמונות או על גבי האוביקטים.

עקומות בתמונות היו אחד הנושאים החשובים במחקר בראיה ממוחשבת בעשורים אחרונים. פותחו שיטות רבות לגילוי העקומות ושימוש בהן. למרות שעקומות על משטחים חשובות באותה מידה כמו עקומות בתמונות, הן קיבלו פחות תצומת לב. הוצעו מספר הגדרות של עקומות על משטחים, כשכל הגדרה מתאימה לבעיה מסוג אחר. המשפחה הכי ידועה של עקומות על משטחים היא קווי רכס (ridges) וקווי עמק (valleys). הקווים האלה מייצגים נקודות בהם כיוון המשטח משתנה באופן חזק. קווי רכס וקווי עמק מתארים תכונות חשובות של אובייקטים, אבל הן לא מתאימות לאוביקטים מסויימים כמו אוביקטים חלקים ואוביקטים עם תבליטים. משפחות אחרות של עקומות הן עקומות פרבוליות (parabolic curves) שמחלקות משטחים לאזורים היפרבוליים ואליפטיים, ועקומות של אפס עקמומיות ממוצעת (zero mean-curvature curves) שמסווגות משטחים לאזורים קמורים וקעורים. לעקומות האלו יש אלגנטיות מתמטית, אך חסרה משמעות פרקטית. הן מאוד רגישות לרעש ובמקרים נדירים מתארות תווים אמיתיים של אוביקט.

כמו כן הוגדרו עקומות שתלויות בנקודת מבט. העקומות האלה תלויות לא רק במאפיינים הגיאומטריים של האוביקט אלא גם במיקום של הצופה. לעתים קרובות העקומות האלה נעימות לעין ולכן מתאימות לאפליקציות גרפיות כמו ציור לא ריאליסטי. אבל הן פחות מתאימות למקרים בהם נדרש ייצוג יציב של תוי האוביקט מכל הכיוונים, כמו, למשל, בארכיאולוגיה ורפואה.

בתזה הזאת אנו מציעים סוג חדש של עקומות, הנקרא *עקומות תבליטים* (*relief edges*), אשר מטפל במגבלות של קווי רכס ועמק המתוארים לעיל. עקומות תבליטים מתאימות במיוחד לאוביקטים עם תבליטים. אינטואיטיבית, ניתן לחשוב על אוביקט כזה כמשטח חלק לא ידוע שעליו מוגדרת פונקציה הנקראת *תמונה מקומית (local image)*. עקומות תבליטים הן השפות (edges) של התמונה הזאת. אנו מראים איך ניתן לחשב את השפות האלה מהמאפיינים הגיאומטריים המקומיים של האוביקט.

אנו מראים איך עדיף להשתמש בעקומות תבליטים על מנת לתאר אוביקטים תלת ממדים. ניתן להשתמש בעקומות בצורה ישירה ,פשוט על ידי ציור שלהן על גבי האוביקט .עם זאת, ייצוג טוב יותר מתקבל כאשר העקומות משולבות עם צביעה המבוססת על עקמומיות המשטח. לפי שיטה זו , האוביקט נצבע בגווני אפור כאשר עוצמת הצבע בנקודה על האובייקט פרופורציונלית

המחקר נעשה בהנחיה של פרופ' אילן שמשוני מחוג למערכות מידע
באוניברסיטת חיפה ופרופ' איילת טל מפקולטה להנדסת חשמל
בטכניון

תודות

אני מודה להוריי, תמר וויקטור קולומנקין, על תמיכתם

אני מודה לפרופ' אילן שמשוני ופרופ' איילת טל על עזרתם
בכל שלבי המחקר

אני מודה למוסד הטכניון ולקרן אולנדורף על התמיכתם הכספית
הנדיבה בהשתלמותי

ניתוח עקומות עם אפליקציות בארכיאולוגיה

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
דוקטור לפילוסופיה

מיכאל קולומנקין

הוגש לסנט הטכניון - מכון טכנולוגי לישראל
תמוז תשע"ג חיפה יוני 2013