Reconstruction of relief objects from line drawings

Michael Kolomenkin
Technion
michkol@tx.technion.ac.il

George Leifman
Technion
gleifman@tx.technion.ac.il

Ilan Shimshoni
The University of Haifa
ishimshoni@is.haifa.ac.il

Ayellet Tal
Technion
ayellet@ee.technion.ac.il

Abstract

This paper addresses the problem of automatic reconstruction of a 3D relief from a line drawing on top of a given base object. Reconstruction is challenging due to four reasons – the sparsity of the strokes, their ambiguity, their large number, and their inter-relations. Our approach is able to reconstruct a model 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. Then, given a base and an interpretation, we propose an algorithm for reconstructing a consistent surface. The strength of our approach is demonstrated in the reconstruction of archaeological artifacts from drawings. 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.

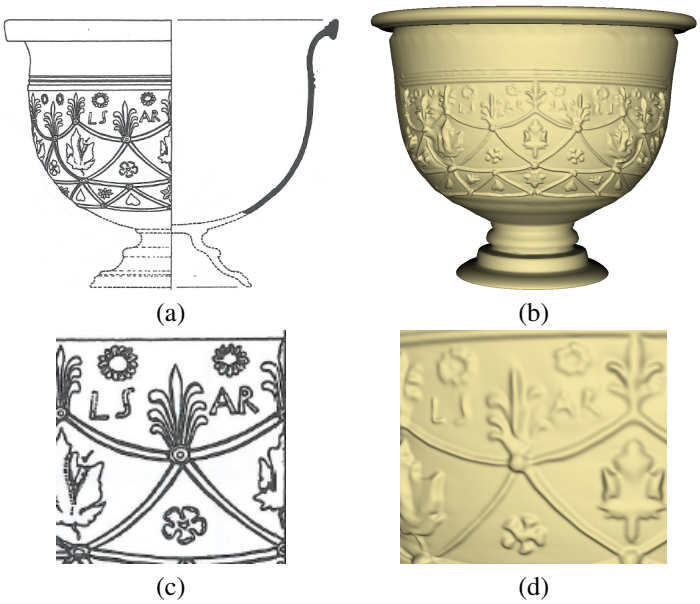


Figure 1. Reconstruction of the relief on a Roman vase from a manual drawing consisting of 571 curves. (a) is the original drawing, (b) is our 3D reconstruction. (c) and (d) are zoomed versions of (a) and (b).

1. Introduction

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 [24]. Therefore, it is only natural that line drawings would be used for automatic reconstruction. In some cases, a line drawing is not only the preferred input, but rather the only available source of information. For instance, in archaeology, if we wish to reconstruct artifacts found in the past, sometimes the only available data is the line drawing appearing in the archaeological report (Figure 1(a)).

Automatic 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 specified by the user and handled by the algorithm efficiently. Fourth, these strokes are inter-related. For instance, in the relief of Figure 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 [6] 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 [5, 8, 14, 20, 25]. 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 [16, 22]. 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 [4, 26].

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 [9, 10, 18]. 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 editing of reliefs are interactive and can handle only one curve at a time [7, 11, 27] . They have difficulties handling multiple strokes, which is one of our goals.

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 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.

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 [13, 15]. We assume that the base is given and focus on the reconstruction of the relief. 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.

To demonstrate the strength of our approach, we apply it to a highly challenging domain – the reconstruction of archaeological artifacts from drawings. Artifacts containing reliefs are important sources of information, since they are fingerprints of periods and cultures. Line drawings have been the standard method of their 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. These inputs are usually highly complex. State-of-the-art algorithms were not designed with such drawings in mind.

The contribution of this paper is hence threefold. First, we propose an efficient approach for reconstruction of reliefs from line drawings (Section 2). The method is able to handle highly complex real drawings. Second, we propose solutions to two problems – line-drawing interpretation and relief reconstruction (Sections 3-4). Last but not least, the method makes a significant step towards solving an important problem in archaeology (Section 5) – a domain that recently attracted a lot of attention in computer vision and computer graphics [2, 12, 13, 19].

2. General approach

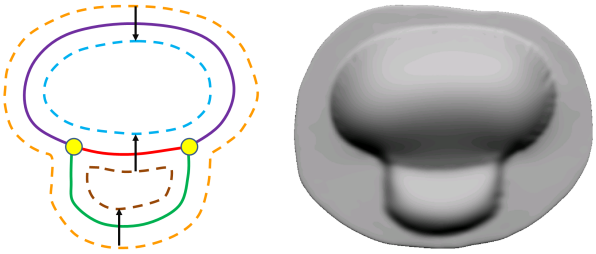
Given a line drawing of a relief object and a base surface, our goal is to reconstruct the surface, as shown in Figure 1. The algorithm reconstructs the relief on the input base regardless of its shape. Below we present the problem definition and outline our algorithm.

Problem definition: In 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.

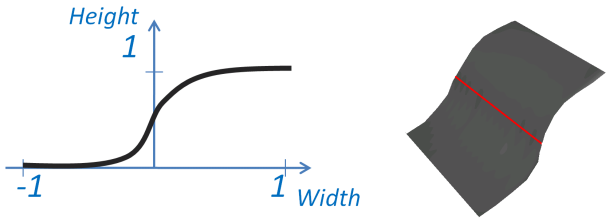
A drawing consists of *drawing curves*, *junctions*, and *margins* (Figure 2(a)). We define the *drawing curves*, or simply the *curves*, as the lines of the given drawing. They indicate visually-meaningful locations on the relief. The curves may be connected by *junctions*, but cannot cross them. *Margins* 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



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

Figure 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.



(a) (b)

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

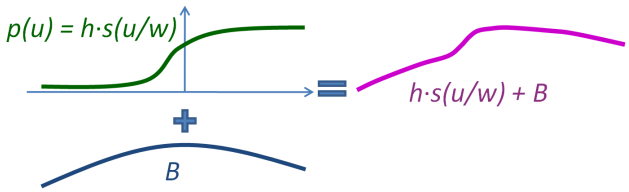


Figure 4. The relief in a curve's neighborhood is a combination of the step edge $p(u)$ and the base B.

of height h and width w (Figure 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 $p(u)$:

$$p(u) = h \cdot s(u/w), \tag{1}$$

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 the point on B at (signed) distance u from the curve. Thus, the surface S within the curve's margins on the cross section can be written as:

$$S(u) = p(u) + B(u) = h \cdot s(u/w) + B(u). \tag{2}$$

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.

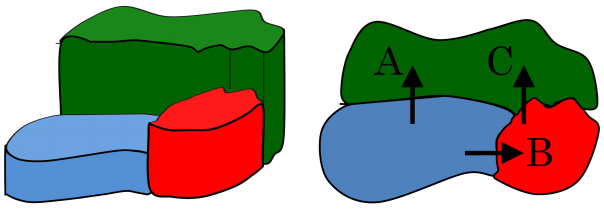
Specifically, we need to compute the height h for the step edge of every curve in a consistent manner. We assume that the margin width w and the step edge's shape $s(x)$ are constant for all the curves. The margin width is set to 0.05% of the diagonal of the object's bounding box. The step edge shape is the shape of an ideal step edge smoothed with a Gaussian of standard deviation equal to an average edge length. Both w and $s(x)$ 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 [21]. Then, junctions are detected and margins are generated. These elements serve as the input to our algorithm. Sine the input drawing is manual, we assume it does not contain noise and 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 3). Given the curves and the step edges, the surface is reconstructed (Section 4).

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 5). The solution to this problem should address the challenges of interpretation consistency combined with a large number of strokes.

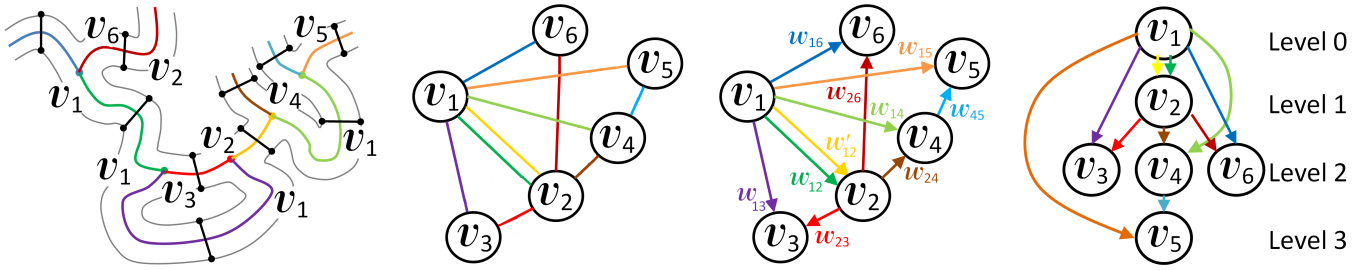


Object Line drawing

Figure 5. Height consistency. The height of step edge **A** should be equal to the sum of the heights of the 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 [23] for adding depth to cartoons. Below we first describe the reduction and then the algorithm for solving the corresponding graph problem.

Reduction to a graph problem: Given a drawing (Figure 6(a)), we initially represent it by an undirected graph



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

Figure 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.

$G = \{V, E\}$ as follows (Figure 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 find for each directed edge a positive 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 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. $w_{km} > 0$, positive height.
2. $h_k + w_{km} = h_m$, height consistency.

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

3. Given directions 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 important for correct reconstruction, since without it the resulting surface is randomly chosen. It is used, for example, to make the reliefs consistently higher or lower than the base.

Our algorithm employs the following heuristic to determine this constraint. 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.

Algorithm: To estimate the weights of the edges satisfying the above constraints we divide the task into two subtasks. First, the graph is directed. Next, the edge weights are computed. We elaborate below.

1. Directing the graph: Given an undirected graph (Figure 6(b)), we aim at generating a directed graph that will allow us later to assign consistent weights according to the second constraint (Figure 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 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 the 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 it is performed in topological sorting (Figure6 (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.

4. Surface reconstruction

Given a base, a set of curves, and the heights of their step edges that were computed in Section 3, the goal is to reconstruct the 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 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 undesirable 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:

$$r(\mathbf{p}) = p(\nu, \upsilon) + C_{\mathbf{p}}, \tag{3}$$

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, whose computation will be discussed below.

2. Computing the relief model: Given a mesh of the 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 2. The non-planar reconstruction of the surface in between the curves in Figure 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. Let $\mathbf{p}$ be a vertex of the mesh, we search for L that satisfies:

$$\begin{aligned} L(\mathbf{p}) &= \Delta r(\mathbf{p}), \quad \mathbf{p} \in \text{curve's neighborhood}, \\ \Delta L(\mathbf{p}) &= 0, \quad \text{elsewhere.} \end{aligned} \tag{4}$$

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 [17]:

$$\Delta f(\mathbf{p}) = \frac{1}{2A} \sum_{j \in N(\mathbf{p})} (\cot(\gamma_j) + \cot(\delta_j))(f(\mathbf{p}) - f(\mathbf{p}_j)). \tag{5}$$

Obviously, Equation (5) is linear in the values of f. Hence, any linear-equation solver can be utilized to calculate f.

To solve the system of Equations (4), we need to know the values of r. Recall however that by Equation (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:

$$\begin{aligned} R(\mathbf{p}) &= r(\mathbf{p}), \quad \mathbf{p} \in \text{A curve's neighborhood}, \\ \Delta R(\mathbf{p}) &= L(\mathbf{p}), \quad \text{elsewhere.} \end{aligned} \tag{6}$$

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 (4) and thus it is solved similarly.

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 (4) and (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.

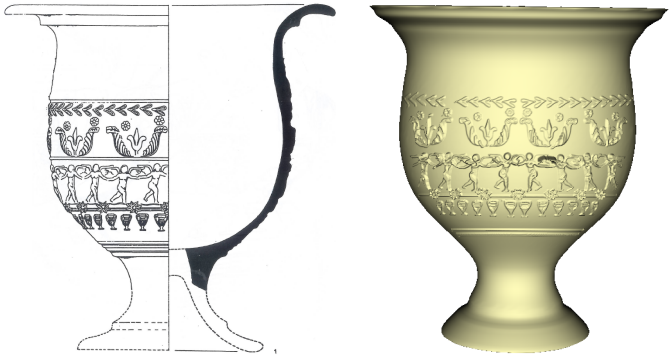
5. Implementation and results

Implementation and running times: We implemented the two components of the algorithm: line drawing interpretation and reconstruction. The former is implemented in C++. Since it has a linear asymptotic complexity, the running time is negligible.

Reconstruction, however, may provide a challenge for large models, since large and sparse linear systems in Equations 4 and 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 on the GPU using cusp library [1]. Under this implementation the running times of the algorithm are in the range of 15 seconds for an 80,000-vertex model (Figure 12) to 200 seconds for a 970,000-vertex model (Figure 7) on a 2.4GHz Intel Core 2 Duo processor with 2GB of memory.

Results: We will now present several examples of automatic reconstruction of reliefs of archaeological artifacts, demonstrating the capabilities of our algorithm to deal with complex line drawings. All the archaeological drawings appearing in this section were taken from [3].

Figures 1 and 7 show the reconstruction of the intricate reliefs of two vases. Though these drawings consist of 571 & 1300 tightly interconnected lines, the reconstruction achieves visually-pleasing results. The speed-up steps described above enable the reconstruction of these objects in a reasonable time.



(a) Given drawing (b) Reconstruction

Figure 7. Reconstruction of the relief of a Roman vase

Figure 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 [27], 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 8 demonstrates the reconstruction of the relief of two Roman oil lamps. The drawings contain 39 and 49 lines, respectively. It can be seen that our automatic reconstructions are quite good.

Figure 9 shows a recently-discovered Rhodian stamp (third millennium AD) indicating a major annual celebratory event.

The same stamp was discovered on a bunny (Figure 10), demonstrating that our method can place reliefs on any base surface irrespective of its complexity.

Figure 8. Reconstruction of the reliefs of two Roman oil lamps. The given drawing is on the left side of a pair and the reconstructed relief is on the right.

Figure 9. A recent major discovery of a Rhodian stamp (circa 21^{st} century AD)

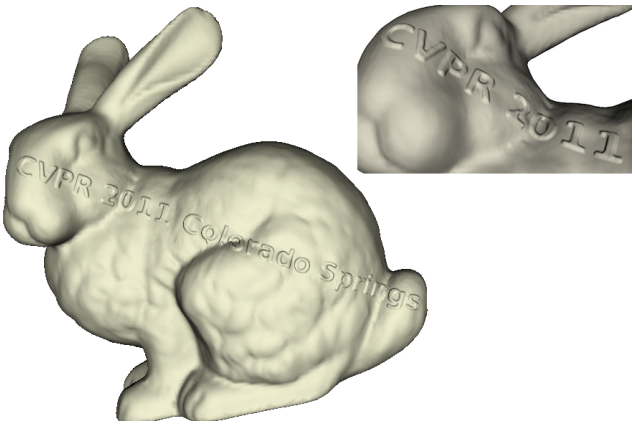
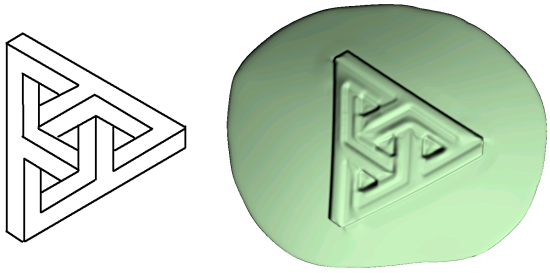


Figure 10. CVPR logo put onto the bunny.

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. For example, ridges and valleys can be employed.
2. The height and the direction of the step edge (or other profile) can be set by the user and added to Constraint 3 of the interpretation algorithm (Section 3).
3. The user can set an arbitrary number of profiles per curve, enabling the shape of the surface to change along the curve.

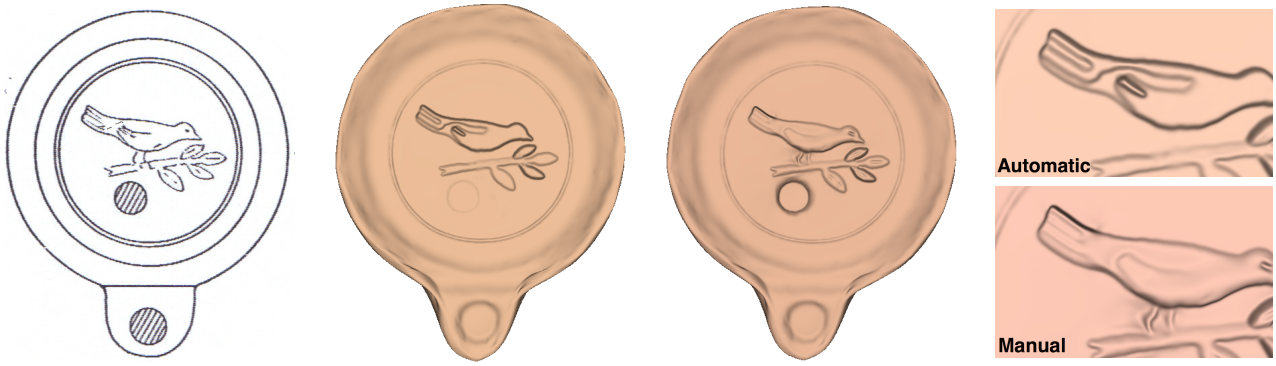
For example, the automatic interpretation fails when the drawing represents impossible objects, such as the ones created by Escher. However, our system can create a 3D relief by different profiles with different heights along the curves. Thus, straight lines in the drawing yield curved lines in 3D. This is illustrated in Figure 11.



(a) Given drawing (b) Manual reconstruction

Figure 11. Limitation – impossible objects cannot be reconstructed automatically. They will, however, be reconstructed if the user provides several profile heights along the curves.

In Figure 12, 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 drawing includes ridges (the birds 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.



(a) Drawing (b) Automatic reconstruction (c) Manual intervention (d) Zoom-in

Figure 12. An example of manual fine-tuning of the results. The original drawing (a) includes ridges (bird's legs) and valleys (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.

6. Conclusions

In this paper we addressed the problem of automatic reconstruction of a relief object from a line drawing. We identified four challenges that have to be tackled in order to reconstruct reliefs from complex line drawings: 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.

Based on the observation that relief objects are composed of a smooth base and relief details, 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. The 3D models can be used for comparison to 3D scans of newly-found artifacts of the same type.

Aknowledgements. 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.

References

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

[2] B. Brown, C. Toler-Franklin, D. Nehab, M. Burns, D. Dobkin, A. Vlachopoulos, C. Doumas, S. Rusinkiewicz, and T. Weyric. A system for high-volume acquisition and matching of fresco fragments: Reassembling Theran wall paintings. *ACM Trans. Graph.*, 27(3):84:1–9, 2008.

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

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

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

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

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

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

[9] 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.

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

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

[12] 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.

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

[14] 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.

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

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

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

[18] 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.

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

[20] 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.

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

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

[23] 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.

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

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

[26] 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.

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