iWIRES: An Analyze-and-Edit Approach to Shape Manipulation

Ran Gal
Tel Aviv University

Olga Sorkine
New York University

Niloy J. Mitra
IIT Delhi

Daniel Cohen-Or
Tel Aviv University

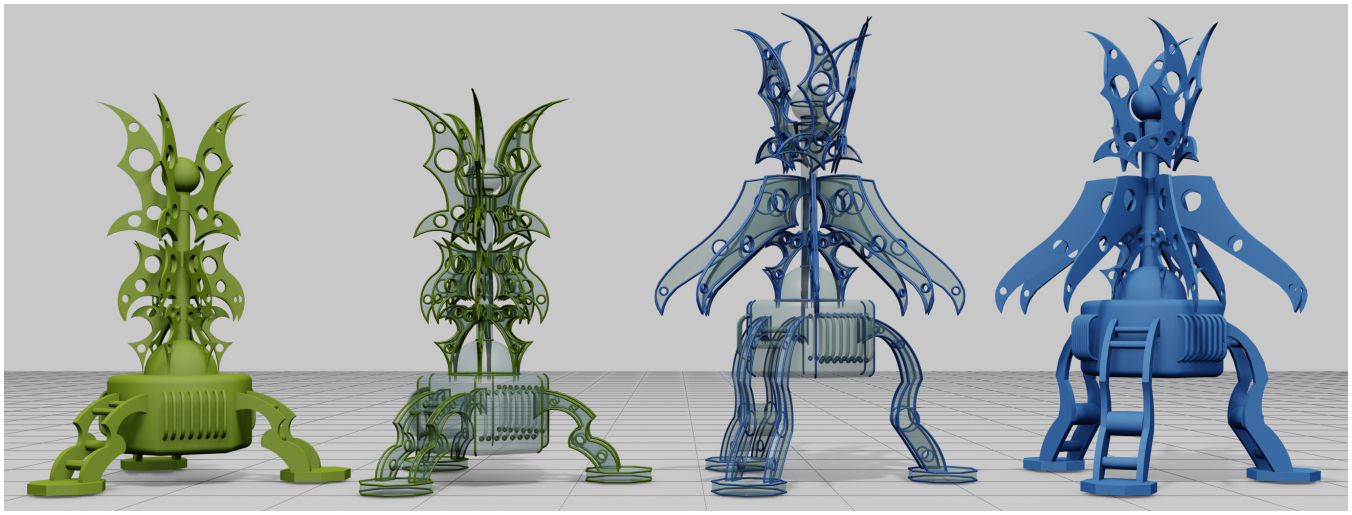


Figure 1: *A complex model (left) consisting of* 108 *components is analyzed and* 250 *intelligent wires (in green) are extracted. Editing a few wires induces a new wire configuration (in blue) and leads to the result on the right.*

Abstract

Man-made objects are largely dominated by a few typical features that carry special characteristics and engineered meanings. State-of-the-art deformation tools fall short at preserving such characteristic features and global structure. We introduce iWIRES, a novel approach based on the argument that man-made models can be distilled using a few special 1D *wires* and their mutual relations. We hypothesize that maintaining the properties of such a small number of wires allows preserving the defining characteristics of the entire object. We introduce an *analyze-and-edit* approach, where prior to editing, we perform a light-weight analysis of the input shape to extract a descriptive set of wires. Analyzing the individual and mutual properties of the wires, and augmenting them with geometric attributes makes them intelligent and ready to be manipulated. Editing the object by modifying the intelligent wires leads to a powerful editing framework that retains the original design intent and object characteristics. We show numerous results of manipulation of man-made shapes using our editing technique.

Keywords: mesh editing, man-made objects, structured deformation, space deformation, constraint propagation

1 Introduction

In recent years, shape editing has been extensively studied by the geometric modeling community. In particular, research efforts have been devoted to allow the user to directly manipulate surfaces while preserving their geometric surface details. Generally speaking, a key challenge in shape editing is to enable *intuitive manipulation* – that is, the performed change is the one expected. Clearly, such a notion is highly domain dependent. It is natural to expect that manipulation applied to the shape preserves the local surface details [Botsch and Sorkine 2008]. Such detail-preserving techniques treat the edited object to be made of a homogeneous, rubber-like material that responds uniformly to user manipulations. These approaches have been highly successful for organic objects, such as faces, body parts, animals, etc. However, they are less suited for man-made shapes, such as furniture, cutlery, mechanical parts or electronic devices. Such engineered models are largely dominated by flat or smooth faces, where the shape is defined by a few typical features which carry special characteristics and geometric meaning.

We hypothesize that conserving the properties of this rather small number of features allows preserving the defining characteristics of the entire object. In contrast, a surface manipulation that assumes a homogeneous surface, oblivious to the special characteristics of the shape, actually damages its high-level structure, thus defeating the purpose of editing. We use the term *editing*, rather than *deformation*, as the former is not supposed to be destructive by definition. Editing is rather a constructive operation that implicitly aims at preserving the essence of the shape (see Figure 1).

Our work is inspired by the research of Singh and Fiume [1998] and Orzan et al. [2008]. These works show that an entire shape or an image can be defined and characterized by a rather small set of curves. We adopt the name *wires* of Singh and Fiume to denote the curves that are key structural features capturing the shape. Our 3D geometry editing framework aims to preserve these key features and characteristics of objects, especially man-made ones.

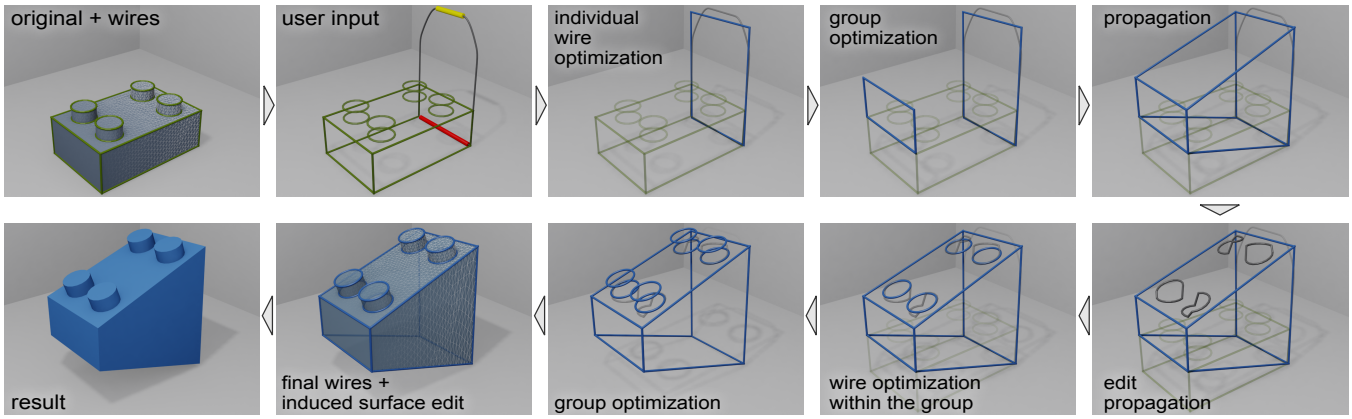


Figure 2: *Deforming a* Lego *model using* iWIRES. *Given the model geometry, we extract wires (shown in green), learn their individual characteristics and mutual relations, and preserve them when the model undergoes deformation. The user-prescribed edit, defined by the yellow and red handles, is mapped to one of the wires (in gray). We optimize the individual wire (as shown in blue), propagate the edit operation to other wires (gray), and perform a multi-stage constrained optimization to enforce the individual properties and mutual wire relations. The resulting wire positions, shown in blue, determine the final edited shape (see Section 2 for details).*

We take a *analyze-and-edit* approach, where prior to editing, we perform a light-weight analysis of the input shape to extract the wires, their individual geometric characteristics, and relationships with other wires. Supplementing the wires with these special geometric attributes makes them *intelligent* and ready to be manipulated. Edits are applied to the wires while preserving the attached characteristics and relations. Editing the intelligent wires results in a *constrained deformation* setup rather than a free-form deformation. For instance, when the object contains a circular feature or boundary, the edited shape tends to maintain this property (see Figure 1). The final shape of the object is determined using the deformed wire scaffolding. The accompanying video exhibits our editing system in action.

2 Overview

Our *analyze-and-edit* approach is based on the observation that man-made shapes can often be characterized by a few special curves or wires. During editing, by maintaining the individual characteristics of the wires and their mutual relationships, we attempt to preserve the essence of the manipulated shape. Figure 2 demonstrates the basic flow of our method. Our framework comprises of two main subparts:

Analysis. Given a shape, we first extract a set of defining curves or wires. We analyze the object to identify the characteristics of the individual wires (Section 4.1), and learn their mutual relationships (Section 4.2), thus making the wires intelligent. Though for general models it is challenging to extract feature curves [Ohtake et al. 2004], for most man-made models, sharp crease lines are good candidates. Even such simple curves capture powerful shape characteristics. In our *Lego* example, we identified eight circles and twelve line segments to form the wire collection. Our framework deals with how to efficiently and intelligently deform/edit such a collection of wires, and is oblivious to their extraction process.

A key step of our algorithm deals with identifying the defining properties of the wires and their global relations. For example, the analysis step for the *Lego* yields the following: the eight planar circular curves are identified as individual wires, as well as the six rectangles. The mutual relations are recorded by grouping the relevant wires: for instance, wires lying on the same plane or on parallel planes form (not necessarily mutually exclusive) groups (see Figure 3).

Edit. Like most deformation tools, we allow the user to indicate modeling constraints by manipulating deformation handles or by sketching (Section 6). We propagate this deformation to the closest wire and optimize it to enforce its individual characteristics (Section 5.1). In our *Lego* example, an individual "seed" wire is first free-form deformed using the user manipulation, and then optimized to become a rectangle. This edit is propagated to other wires which are mutually related to the seed wire (i.e., the wires in the groups the seed wire belongs to), using a dedicated local frame encoding. The groups are optimized in order to restore the group characteristics (for instance, planarity), as described in Section 5.2. The edit then continues to propagate to other wire groups, influencing the shape of their wires (for example, the rectangular wires of the *Lego* influence the circular wires, as depicted in gray on the bottom right of Figure 2). The individual shape of each newly influenced wire is optimized, as well as the mutual (group) properties. Once all the wires have been treated and their final form is set, they serve as modeling constraints for a differential surface deformation (Section 5.3), and the model is reconstructed respecting the edited wires (see Figure 2, bottom left).

The key argument of this work is that man-made or engineered objects can be distilled using a few special 1D wires and their mutual relations, and edited using simple means while preserving their defining characteristics. In this paper we take a first attempt to realize this approach, and present a technique that allows a quick creation of models by reusing and editing existing shapes [Funkhouser et al. 2004] (see Figure 10). A notable property of our method is that it can deal with disconnected surfaces, and in general, models that consist of a multitude of parts (for example, the model in Figure 1 has 108 parts). Surface-based methods cannot deal with disconnected components in a coherent way. Space-deformation methods can, however, they are oblivious to the characteristics and properties of the model features since they treat the entire space and the embedded surfaces uniformly. Even editing a simple model like the *Lego* by state-of-the-art methods, while keeping its characteristic features, is extremely hard (see Figure 6). On the other hand, iWIRES provides an intuitive outcome with easy interaction.

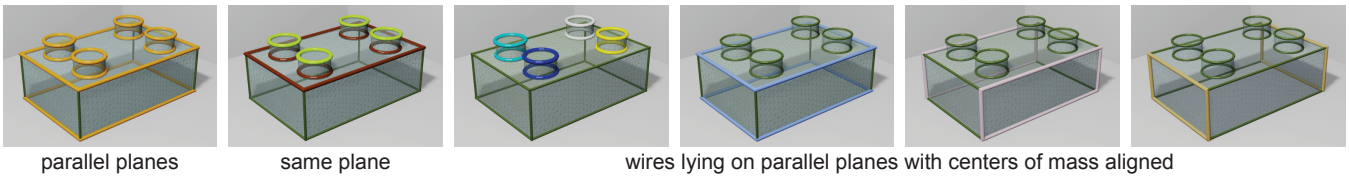


Figure 3: *Wire groups resulting from the analysis of the* Lego *model. Each group is rendered in a distinct color. Note that the groups are not necessarily mutually exclusive.*

3 Related work

Space deformations (also called free-form deformations) are to this day the method of choice for shape deformation. A space deformation is defined via a (usually simple) control object; user-defined deformation of this object is interpolated to the 3D space and evaluated at the input surface points. Space deformations are popular since they can handle various object representations, including free-form surfaces, meshes with multiple connected components, or even unstructured point clouds. In addition, space deformations are simple to implement, and they are highly efficient and robust, because the cost of the deformation is mainly dependent on the complexity of the control object and not on the deformed shape. Early space deformations used lattices as control objects [Sederberg and Parry 1986; Coquillart 1990; Milliron et al. 2002]; these, however, are cumbersome to manipulate manually since the control points do not necessarily correspond to meaningful parts of the shape that the user wishes to modify. The structure of the underlying shape that is being deformed by the space warp can be easily destroyed, unless the space warp is very carefully designed.

Over the last decade it has been recognized that precise control over the properties of the deforming surface is required for more satisfactory results. Since space deformations are oblivious to the actual shape that is being edited, better control can be gained by employing control objects whose shape and structure is closely related to that of the edited shape. The *Wires* framework by Singh and Fiume [1998] employs spatial curves to construct the deformation; the curves are aligned with prominent characteristic features of the edited shape and affect the surface parts in their vicinity. The framework uses a clever way to blend the space deformations induced by multiple curves; however, the deformation of the individual wires and the interaction among multiple wires is completely left to the user. In our work, the wires are supplied with intelligence about their own shape and about other influencing wires, such that important relationships are preserved during deformation.

Later work proposed the use of so-called cages as control objects for shape deformations. Typically, the cage is a very coarse and offsetted version of the input shape. Various coordinate functions have been designed to carry over the deformation of the cage to the entire space, such as mean-value coordinates [Ju et al. 2005], harmonic coordinates [Joshi et al. 2007], Green coordinates [Lipman et al. 2008]. Radial-basis functions [Botsch and Kobbelt 2005], volume-preserving warps [Angelidis et al. 2004; von Funck et al. 2006] and locally rigid transformations [Sumner et al. 2007; Botsch et al. 2007] have also been explored. Still, all these approaches implicitly treat the edited shape as a homogeneous surface and pay no special attention to its high-level structure and features.

In a parallel vein of research, deformation approaches that work directly on the input shape have been developed. Their great advantage is the ability to formulate the deformation process so as to precisely control the low-level differential surface properties of the edited object, with the downside being the direct dependence of the computational costs on the complexity of the object's representation (e.g., the number of mesh vertices). State-of-the-art surface-based deformation techniques have achieved a good understanding of the necessary deformation mechanisms that preserve such low-level shape characteristics as curvature, normals or local rigidity by formulating corresponding variational problems (see the survey of linear methods in [Botsch and Sorkine 2008]; for nonlinear techniques we refer to [Botsch et al. 2006; Sorkine and Alexa 2007] and the references therein). For articulated shapes, local volume preservation, limb rigidity and other physical constraints have been explored [Botsch and Kobbelt 2003; Zhou et al. 2005; Huang et al. 2006; Lipman et al. 2007; Shi et al. 2007; Au et al. 2007].

Several surface-based approaches allow manual specification of varying surface stiffness [Botsch et al. 2006; Popa et al. 2007], thus allowing some surface parts to behave more rigidly than others. However, no higher-level knowledge about the structure of the object is deduced, which makes these techniques rather difficult to use for the editing of man-made objects. It is worth noting that a restricted type of space deformation was designed in [Kraevoy et al. 2008] for axis-aligned non-homogeneous scaling of structured man-made models; the space deformation protects certain parts of space, occupied by sensitive object features, such that they only undergo similarity transformations. This method is related to material-aware deformation [Popa et al. 2007], yet again, mostly low-level differential information is considered when defining the parts to be protected during axis-aligned scaling.

Masuda et al. [2007] propose to manually mark features (such as hole boundaries or sharp curves) and incorporate hard constraints into the deformation to rigidly preserve the features' individual shapes; their deformation framework was later extended to non-manifold meshes and disconnected meshes in [Masuda and Ogawa 2007] using proximity constraints. Complete rigidity of the features may be too restrictive in many scenarios. Further, the relationships *between* the features are not considered in the works above. Cabral and colleagues [2009] focus on reshaping architectural models and propose a more lenient way to handle features (mostly rectilinear in their case) by constraining the angles but allowing changing the length of individual segments. We explore a more general approach to optimize the shape of the features (Section 5.1).

As mentioned above, our work is inspired by the idea that many man made shapes can be described by a sparse collection of characteristic curves. This was the motto of *Wires* [Singh and Fiume 1998] that introduced a novel modeling paradigm for space deformations; the notion of a curve network that induces an interpolating surface has been long used in traditional CAD/CAM to model shapes from scratch using free-form surface patches. The idea has been recently combined with sketch-based mesh modeling [Nealen et al. 2007], relieving many topological and geometric restrictions of a classic curve network. Collections of 2D curves with attached color information are used to define piecewise-smooth color gradients of arbitrary geometry for 2D vector graphics in [Orzan et al. 2008], again capitalizing on the observation that a sparse set of characteristic feature curves often suffices to completely define the entire object (a drawing, in this case). Our work contributes the idea of

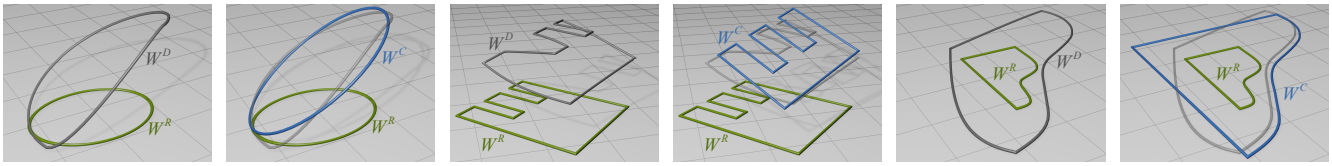


Figure 4: *Individual wire optimization. W^R (green) denotes the reference curve, W^D (gray) is the deformed wire (following user manipulation or edit propagation) and W^C (blue) is the final, optimized shape of the wire.*

equipping the curves with additional information about their shape and relation to other curves, such that high-level structure editing can be performed without destroying important features and shape properties of the object.

It should be noted that our goal is not rigorous reverse engineering of the input shape [Benkö et al. 2001; Attene et al. 2007] or structure detection [Pauly et al. 2008], but rather light-weight analysis and easy subsequent interaction. This allows us to handle a rather broad spectrum of shapes and avoid the typical hurdles and restrictions of CAD boundary representations such as NURBS patches, making the technique attractive to a wider user audience.

4 Wire extraction and analysis

Our shape analysis is a weak form of reverse engineering, as we do not precisely reconstruct the entire shape, but rather require just enough information to enable structure-preserving edits. This requirement is much simpler compared to the classical (and notoriously hard) problem of shape understanding. We select a sparse set of one-dimensional features and mark them as *wires*. These are salient curves on the object that sufficiently describe a shape for the purpose of editing. We do not pose any topological restrictions on the collection of wires, nor do the wires necessarily segment the shape into surface patches. Instead of directly preserving characteristics of the input surface, we encode them using our wire collection. We identify the characteristics of the individual wires and their mutual relations, thus making them intelligent. This knowledge is used during subsequent edits. We identify a set of properties common to typical engineered models, such as linearity and planarity, parallelism, symmetry, and maintain such relations during deformations. Each individual wire is made aware of its characteristic properties, while we store their mutual relationships by groups.

Wire extraction. For many man-made objects, the defining lines of the shape lie on the intersections between smooth surfaces. Therefore, to extract the wires, we identify "sharp" mesh edges – those that have sharp dihedral angles or lie on the boundary. We use a tracing procedure to form the wires: starting from a sharp edge that has not yet served as a wire "seed", a wire is formed by walking along consecutive sharp edges; whenever a crossing is reached, we choose the edge that would keep the wire planar, if possible. The wire ends when either the curve becomes a closed loop, or further walk is not possible. The wire extraction process continues until all the sharp edges have been processed. Note that our extraction procedure relies on relatively clean input due to the use of dihedral angles. While it is very often the case with man-made objects, still a more robust technique can be applied in place of our heuristic, such as geometric snakes [Lee and Lee 2002] or ridges and ravines [Ohtake et al. 2004]. This would be suitable when dealing with scanned input that may contain noise. For all the examples shown in the paper and the supplementary video, the wires were automatically extracted from the models using a threshold angle of 40 degrees.

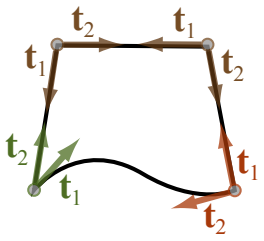
4.1 Individual wire characterization

The following properties are associated with each wire:

- Planar or non-planar.
- "Atomic" type: the entire wire can be well approximated by a straight line, (part of) a circle, (part of) an ellipse or a polynomial curve of a bounded degree. For most man-made objects, we observe that lines and elliptical features capture the defining characteristics of the models. Preserving such characteristics is crucial during the subsequent deformation stage.
- Compound wire: when the fitting error using a single element is large, we segment the wire into sub-wires by dividing it at salient internal angles. Each sub-wire is classified according to an atomic type (if the fitting error is still large, no special attribute is attached). A polyline is a special case of a compound wire, where each sub-wire is a straight line. Polylines appear to be the most common wires in the objects we have tested.

The atomic attribute of a wire (or a sub-wire) is decided by fitting each type and selecting the one with the smallest fitting error. To qualify as a circle or ellipse, the wire should cover a substantial part – at least a quarter of a circle/ellipse in our implementation. For the polynomial curve fitting, we find the degree of the polynomial that best approximates the curve using arc-length parameterization as the domain for the approximation and fitting three separate polynomials along the three principal axes of the curve, with the curve's center of mass as the origin. We test all polynomials with degree between 2 and $d_{\max}$, selecting the lowest degree polynomial with fitting error below a threshold. We used $d_{\max} = 5$ in our implementation.

For compound wires, we also analyze and record the relations between the sub-wires. For a pair of adjacent sub-wires, denote by $\mathbf{t}_1$ the unit tangent vector to the first sub-wire at the connection (joint) point, and $\mathbf{t}_2$ the unit tangent of the second wire. We record the following relations:



Equal connection angles: we identify sets of pairs of adjacent sub-wires that are connected at (approximately) equal angles, i.e., have similar $\mathbf{t}_1^T\mathbf{t}_2$ values. Clusters that have special values of connection angles (in our implementation, 90 and 45 degrees) are additionally noted.

Parallel connections: We are interested in clustering pairs of sub-wires, for which the tangents at the connection point form parallel planes. Namely, we marks sets of sub-wires that have similar $\mathbf{t}_1 \times \mathbf{t}_2$ vectors.

Equal lengths: cluster sub-wires that have nearly equal lengths.

These individual wire constraints are respected in subsequent edits.

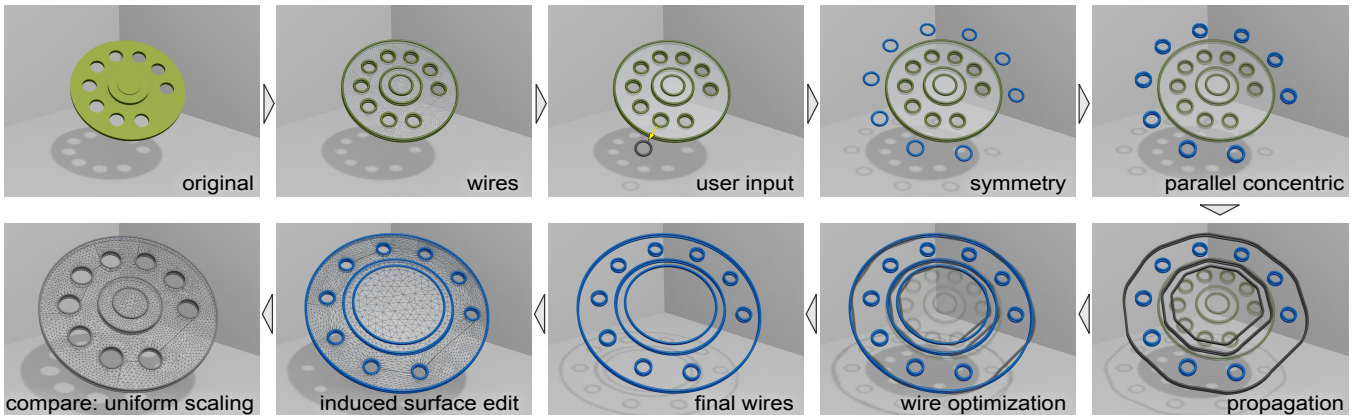


Figure 5: *Various stages of edit propagation on the* Phone Dial *model (see accompanying video).*

4.2 Mutual relations

Any man-made model typically yields a large number of wires (see Figure 8). We analyze their mutual inter-relationships to form *groups* of wires that share certain common traits. This is a key step of our algorithm, since the success and the intuitiveness of subsequent edit operations depend on it. We group wires based on two general criteria (note that the groups are not mutually exclusive):

- Common properties: We identified the following properties, although others can be easily considered as well: wires lying on the same plane; wires lying on parallel planes; wires on parallel planes with approximately aligned centers of mass. In addition to these common properties, we cluster the wires by proximity, such that a wire may belong to a group if its Euclidean distance from the group is smaller than a parameter T_{prox}. For all our examples we used $20-40\%$ of the object's bounding box diagonal length as T_{prox} (for the Lego model we used 100%).
- Symmetry: We use the method of [Mitra et al. 2006] to detect the global symmetry between each pair of wires. We then filter out any symmetry that is common to less than T_{symm} wires, and group the wires belonging to the remaining symmetry sets (we set T_{symm} as 5 in our implementation).

Note that symmetry relation between different parts is an important aspect of many shapes. Such relations often constitute the defining characteristics of objects, and it is desirable to preserve them under any deformations. While local deformation techniques successfully preserve surface details, often global symmetry relations are lost during edit operations. In our framework, we retain detected symmetry relations, specifically reflective and discrete rotational symmetry, by preserving the necessary inter-wire relations during the edit phase.

5 Wire editing

The user initiates edit operations in any of the following interaction modes: directly dragging a surface part, deforming an individual wire, or a sketch-based interaction (see Section 6). The user input defines a modeling constraint that propagates to the wires, which in turn undergo an optimization, with the goal to maintain the individual and the group properties as much as possible, while satisfying the modeling constraint. Finally, the new configuration of the edited wires induces a deformation of the surface itself.

The core of our editing method is the modification of the wires to fit the modeling constraints while preserving their original properties. The problem is naturally ill-posed: on the one hand, preserving all the characteristics of the individual wires and the wire groups exactly is usually impossible, and on the other hand, there are infinitely many ways to define an approximate preservation. We opted out of defining a single global optimization function, because the various terms describing individual wire properties and the mutual relations all have very different nature, and finding an appropriate weight assignment for all of them is difficult and non-intuitive. Instead, we propose a prioritized propagation approach, where the different wire properties are optimized at different stages, starting with the individual wires directly affected by the modeling constraints, then moving on to the groups these wires belong to, and continuing to other groups by proximity and symmetry. This approach, while being greedy, eliminates painstaking (and model-dependent) weight tweaking and global nonlinear optimization, and works well in practice (see Figure 5).

5.1 Individual wire optimization

When the shape of a wire is altered (as a result of direct user manipulation, or indirectly, when the edit operation is propagated), the original wire characteristics can be destroyed. We perform an optimization process to restore them, as much as possible. Let us denote the original (reference) shape of the wire curve by W^R, and the deformed state by W^D. We fit a new curve W^C to W^D, such that W^C possesses the original properties of W^R as much as possible, while resembling W^D in position and shape. W^C is a function of both the reference and the deformed curve states: $W^C = C(W^R, W^D)$ (see Figure 4). At any point during editing, a wire is thus associated with three entities: $W = (W^R, W^D, W^C)$.

As described in Section 4, we analyze the properties of the reference curve W^R and classify the wire as planar/non-planar; atomic (straight line, circle, ellipse, polynomial) or compound. During editing, we fit a corresponding curve type to the deformed state W^D. When the wire is atomic, we fit the particular atomic type using the center of mass of W^D as the origin of the coordinate system and the principal axes (computed by PCA) as the coordinate axes – essentially, the same procedure we used at the analysis stage (Section 4.1). For planar primitives, we define the fitting plane as the plane spanned by the two strongest principal components of W^D.

For compound wires, we fit the corresponding type to each sub-wire, and moreover, we optimize the relationships between the sub-wires (that were analyzed as described in Section 4.1). We define a nonlinear energy functional that measures the deviation from all the properties we have identified in the analysis stage, and find a curve that minimizes this energy while staying close to W^D. The

details of this process are described in the Appendix. Note that the shape optimization may result in a non-planar wire; if originally the wire was planar, we fit a plane to the optimized shape and project to restore planarity. We used a simple vertex-to-vertex distance according to arc-length parameterization; a more accurate albeit expensive approach would be to employ a shape distance metric based on feature correspondence, e.g. as in [Zimmermann et al. 2007].

5.2 Wire group optimization

When some of the wires in a group change their shape, the group properties may become invalidated. We attempt to restore the group properties as follows. For symmetry, whenever the shape of a single wire changes, the same transform (adjusted according to symmetry) is applied to the symmetric wires, i.e., to the symmetry groups the wire participates in. For planar, parallel and concentric groups, we move the wires using small steps of rigid transforms towards their *mean* configuration (using an appropriate distance measure) to iteratively restore the group property. In the case the grouped wires lie on the same plane, we iteratively translate each wire towards the average plane that best approximates all the other wires and rotate the wire towards lying on that plane. We take incremental steps by translating each wire by 5% of the distance to the average plane and rotating by 5% of the necessary angle. We iterate over all the wires in the group, recomputing the average plane each time, until convergence or a maximum number of steps is reached. A similar approach is employed for wires lying on parallel planes and for concentric wires (in the latter case, we fit a straight line to all the centers in each iteration step).

5.3 Edit propagation

Here we explain how the edit of a single wire is propagated to other wires and groups. Starting from an initial set of wires that are affected by the user input (i.e., the modeling constraints), we first propagate their influence to all the wires in the groups that contain those wires (technical details below). We then start proximity-based propagation, influencing groups of wires that are closest, according to Euclidean distance, to the ones already treated. When we optimize each individual wire, we also perform the same change on all the wires that are sharing a global symmetry with it, so treating a wire on one side of the object might influence wires on a remote part of the object. We continue to propagate the edit influence until all the wires are treated.

Influence propagation from a set of wires onto a new wire. The basic building block of the edit propagation procedure is being able to transfer the change of one or several wires to another wire in the vicinity. Assume U is an "untreated" wire that will be influenced by a collection of wires $\{W_i = (W_i^R, W_i^D, W_i^C)\}$ that have already been treated. We only have the reference state U^R of U, and we are looking for the deformed state U^D based on the influence propagation. The optimized state U^C is then computed as previously described in Section 5.1.

We employ local frames encoding, where every point $\mathbf{p}$ on a curve has a canonical orthonormal frame $(\mathbf{p}; \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ associated with it, centered at $\mathbf{p}$. To propagate the edit influence from the set $\{W_i\}$ onto U, we transfer the change between the local frames of W_i^R and W_i^C onto U. For each point $\mathbf{p}$ on U^R, we find the closest point $\mathbf{q}^R$ among $\{W_i^R\}$. We encode $\mathbf{p}$ in the local frame of $\mathbf{q}^R$ on its corresponding reference curve W_i^R, i.e.:

$$(\alpha^R, \beta^R, \gamma^R)^T = (B^R)^T(\mathbf{p} - \mathbf{q}^R), \tag{1}$$

where $B^R = (\mathbf{b}_1^R, \mathbf{b}_2^R, \mathbf{b}_3^R)$ is the matrix of the local frame axes. We then "decode" $\mathbf{p}$ using the corresponding local frame of the edited curve W_i^C:

$$\mathbf{p}^D = \mathbf{q}^C + \alpha^R \mathbf{b}_1^C + \beta^R \mathbf{b}_2^C + \gamma^R \mathbf{b}_3^C. \tag{2}$$

By performing this process for all points $\mathbf{p}$ on U^R, we obtain a deformed version $U^D = \{\mathbf{p}^D\}$. An example of the shape we may obtain is given in Figure 5 (bottom right): the set of dialing holes was transformed due to user manipulation and symmetry, and this set influences the other wires (the inner and the enclosing circles) using the procedure described above. The result is the "wavy" curves depicted in gray. The constrained optimization operator $C(U^R, U^D)$ is then applied to obtain the final curve(s) U^C (as in Section 5.1). We use a heuristic to avoid over-constraining: if a compound wire had been influenced by two or more wires, we relax the optimization operator $C(U^R, U^D)$ so that it ignores the equal angles and equal lengths properties (i.e., we remove the corresponding energy terms from the optimization, see Appendix). This allows, for instance, for the side wires of the *Lego* in Figure 2 to assume trapezoidal shape, instead of locking them as rectangles.

Overall influence propagation procedure. We start from a seed set of wires A_s that were treated as a direct result of the user manipulation. We propagate the edit operation onto all the wires in the groups that contain the wires of A_s as described above. After the propagation step, each group is optimized as described in Section 5.2. The edit propagation then continues in the same fashion to other groups, in the order of proximity to the already treated groups. The process ends when all the wires have been treated.

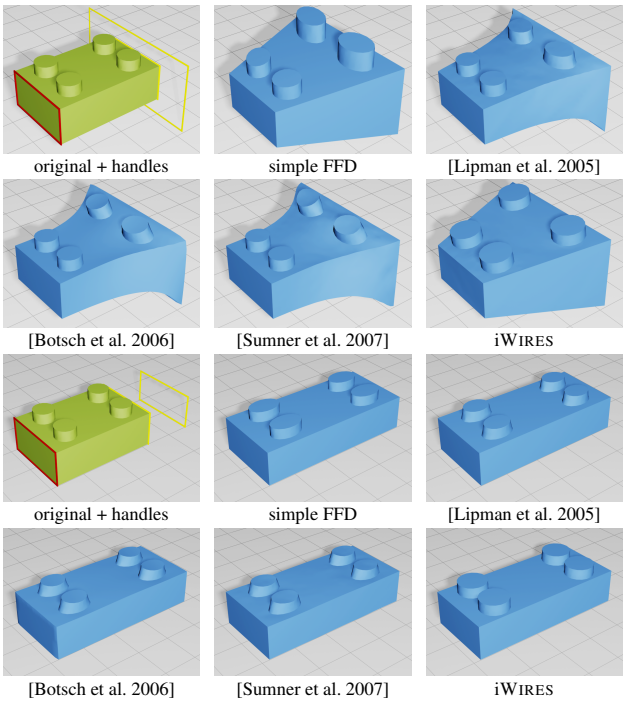


Figure 6: *Comparison with simple space deformation (*$2 \times 2 \times 2$ FFD *in 3D Studio Max) and state-of-the-art shape deformation methods. We keep one side of the* Lego *fixed (depicted in red) and scale (top two rows) or move (bottom rows) the opposite side (the handle is marked in yellow).*

Final Deformation. The result of the wire editing process is a set of wires, each with a reference curve W^R and a corresponding optimized curve W^C (see Figure 4, where reference wires are in green, and optimized versions in blue). We can now use these pairs of curves as modeling constraints to any mesh deformation method. While we used the rotation-invariant coordinates method [Lipman et al. 2005] in this work, other techniques can also be utilized. In addition to the positional constraints, we also derive rotation constraints for the wires by computing the rotation between the local frames on W^R and the corresponding frames on W^C. The method requires each connected component of the object to have at least one positional and one rotational constraint. Hence, we ensure that at least one wire is extracted per connected component of the mesh.

6 Results and discussion

In this section, we describe a few edit sessions performed using iWIRES (please refer to the accompanying video). The setup stage comprises of extracting the wires and learning their properties, as well as having the user mark the modeling constraints, as described below. Figure 8 shows the wires extracted for various models, many of which consist of multiple components. In special cases, it can be desirable for the user to mark additional curves as wires, especially those which are geometrically less distinct (and thus were not detected automatically), but carry strong semantic cues. Among the presented examples, only in the *Phone* model, we manually added one wire to facilitate a link across two disconnected components.

Although the number of individual wires can be large, they are often strongly related. We extract this information as mutual relations, and bin the wires into (possibly intersecting) groups according to their common properties and symmetry relations. The time required to propagate subsequent edits depends on the complexity of the wire relations structure established during this analysis phase, as well as on the number of wire groups that are non-trivially affected due to the desired edit . Thus, our edit framework is output sensitive. The *Toy Jeep*, the *Alien Space-object*, and the *Phone* result in 574, 250 and 75 individual wires, respectively. These wires get grouped into 80, 54, and 13 groups, respectively, before factoring out symmetry. A single edit propagation on models with less than a hundred wires takes less than 2 seconds on a 3GHz machine. On our most complex model, the *Toy Jeep*, edit propagation takes under 4 seconds.

The iWIRES editing framework can be combined with a multitude of common user interfaces: the only requirement is to be able to infer a reasonable and intuitive change to one or more wires from the user manipulation, i.e., to derive appropriate modeling constraints. While alternative metaphors are conceivable, we experimented with three basic types of interaction described below:

Grab-and-drag. The user, oblivious to the underlying wire structure, simply selects a portion of the model as a handle and applies an affine transformation to it (translation, rotation, scaling, etc.). The user also indicates a part of the object that should remain static. In our system, the wire H closest to the handle undergoes the same transformation as specified by the handle to yield the intermediate wire H^D; the wire closest to the static area is internally marked as "treated", to prevent changes during the subsequent optimization. Once these modeling constraints have been set, the optimization is applied to H^D yielding H^C, and the edit operation is propagated (see Section 5 for details).

Explicit wire manipulation. In this mode, the user can see the wires. She is expected to mark wires as fixed, or select and manipulate them using an arbitrary curve editing mechanism (for instance, Laplacian editing [Sorkine et al. 2004] or fiber editing [Nealen et al. 2007]). As before, this yields the sequence $H^R \rightarrow H^D \rightarrow H^C$ that triggers the edit propagation. This mode is useful for clean models with clear defining wires (see Figure 2).

Sketch-based interface. In this mode, the user can draw guiding strokes to affect a significant portion of the model at once. The strokes influence the spatial arrangement of a set of wires (see Figure 7). We assume that the user identifies a set of wires, which we call the *handle set*, that she would like to edit. A reference 3D curve A is computed as the polyline connecting the centers of mass of the wires in the handle set (the red curve in Figure 7). The user then draws a stroke B, shown in yellow, over-sketching the reference curve, and the handle wires around it transform accordingly. Specifically, we establish a correspondence between A and B based on arc-length parameterization (alternatively, sophisticated matching techniques can be used, as in [Zimmermann et al. 2007]), and the depth of each point on B is taken from the corresponding point on A. For each handle wire W, we compute the (rigid) transformation T between the PCA frame of W and the corresponding local frame of A. We translate W such that its center of mass lies at the corresponding point on B, and apply the same transformation T relative to the new local frame. This interaction mode is especially useful for dealing with complex models with a large number of wires, like *Lamp* or *Bench* (see the accompanying video). While this simple sketch-based interface generates quite powerful manipulations, in the future we would like to explore alternate interaction modes that do not expose the wires to users.

Figure 10 showcases a wide range of edits performed on various engineered models using iWIRES. For procedural models, or models available in parametric forms, similar edits are easy. When such representations are not available, it is difficult to reverse engineer such complex shapes. We demonstrate that by merely extracting a few important curves and properly editing/deforming them, it is possible to get intuitive results. Algorithms which are designed for detail preserving deformations of smooth shapes, are ill-suited to perform such edits (see Figure 6). Although for complex models, the number of extracted wires can reach a few hundreds, to the user the system remains simple and intuitive. For example, novice users easily edited complicated models like the *Toy Jeep*, *Phone*, or *Alien Space-object*, to create final forms as shown in Figure 10. Typical sessions, for reasonably complex models included two minutes setup time, followed by edit times of about three–four minutes. Note that edits, unless extreme ones, involve optimizing only a few wire groups and not all the wires scaffolding the models.

Limitations. iWIRES is only as powerful as the detected structure or wire relations. Spurious wires or false learned relations can lead to over-constraining the model (see Figure 9). On the

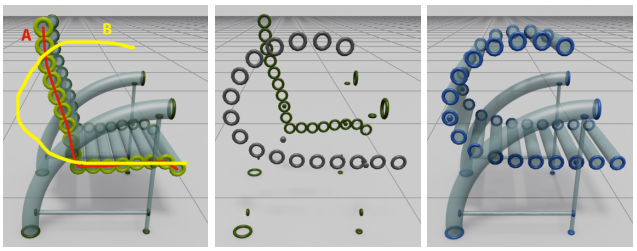


Figure 7: *Sketch-based interface. The user influences a chosen set of wires (light green, left) by drawing a stroke (in yellow). The affected wires are re-arranged accordingly (middle), and then the edit is propagated to the rest of the wires (right).*

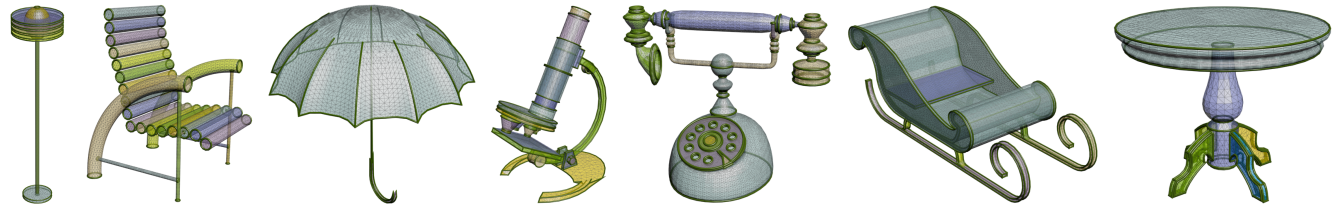


Figure 8: *Some of the models we used, with the wires highlighted, and the connected components indicated by different colors. The number of wires is denoted by #W, and the number of connected components in the model by #C.*

other hand, especially for organic shapes, we fail to retain desirable model properties when important features or appropriate interrelationships are not detected. For noisy models, denoising along with robust feature curve detection may be needed as a preprocessing step before editing the shape using iWIRES. Also, since we do not handle continuous symmetry, we may fail to detect any wires on cylindrical or spherical object parts. In such cases, user assistance may be required. In the extreme, when no relations are detected, our framework reduces to simple surface-based deformation.

In our approach, the final order of edit propagation does influence the results. However, once a wire is treated by group optimization, it is never touched again, preventing conflicts. Although in our system, we apply the constraints in the following order: symmetry, then planarity, then co-planarity, additional control can be given to the user, whereby she can possibly select preferable modes and constraint types.

7 Conclusions

We believe that the analyze-and-edit approach has further potential beyond the application demonstrated here. However, analysis of models which were carelessly generated remains a challenging task that requires more research. Many additional properties and engineering constraints can be detected and analyzed, and then preserved during editing. The approach that we took in this work, namely, using intelligent wires as basic primitives, proved to be a versatile editing framework. However, we believe that with a bit of user guidance, especially with easy-to-mark model semantics, we can provide significant gains towards intuitive control and performance. This should allow novice users to quickly generate large variations of existing non-parametric models.

Acknowledgements

We are grateful to Mario Botsch and Bob Sumner for helping us with Figure 6. The models used in this paper were collected from the Princeton Shape Benchmark, AIM@SHAPE, and via personal communication with Hongbo Fu and Vladislav Kraevoy. We are indebted to Marc Alexa, Andrew Nealen, Denis Zorin and the anonymous reviewers for their valuable comments and suggestions, and thank Andrew Nealen for his lucid narration of the accompanying video. This work was supported in part by the Israeli Ministry of Science, the Israel Science Foundation, and by an ADVANCE Research Challenge Grant funded by the NSF ADVANCE-PAID award HRD-0820202. Niloy was supported by a Microsoft outstanding young faculty fellowship.

Figure 9: *Limitations: (a) Spurious wires on the cylindrical parts cause over-constraining, and these parts no longer move during editing. (b) We may fail to detect meaningful relationships between the wires, resulting in limited effects when editing. (c) Our simple wire extraction procedure may fail to detect any wires; the editing framework then simply reduces to detail-preserving deformation.*

References

ANGELIDIS, A., CANI, M.-P., WYVILL, G., AND KING, S. 2004. Swirling-sweepers: Constant-volume modeling. In *Proc. of Pacific Graphics*, 10–15.

ATTENE, M., ROBBIANO, F., SPAGNUOLO, M., AND FALCIDIENO, B. 2007. Semantic annotation of 3D surface meshes based on feature characterization. *Lecture Notes in Computer Science 4816*, 126–139.

AU, O. K.-C., FU, H., TAI, C.-L., AND COHEN-OR, D. 2007. Handle-aware isolines for scalable shape editing. *ACM Trans. Graph. 26*, 3, 83.

BENKÖ, P., MARTIN, R. R., AND VÁRADY, T. 2001. Algorithms for reverse engineering boundary representation models. *Computer Aided Design 33*, 11, 839–851.

BOTSCH, M., AND KOBBELT, L. 2003. Multiresolution surface representation based on displacement volumes. In *Proc. of Eurographics*, 483–491.

BOTSCH, M., AND KOBBELT, L. 2005. Real-time shape editing using radial basis functions. In *Proc. of Eurographics*, 611–621.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Trans. on Visualization and Computer Graphics 14*, 1, 213–230.

BOTSCH, M., PAULY, M., GROSS, M., AND KOBBELT, L. 2006. PriMo: Coupled prisms for intuitive surface modeling. In *Proc. of Sym. on Geometry Processing*, 11–20.

BOTSCH, M., PAULY, M., WICKE, M., AND GROSS, M. 2007. Adaptive space deformations based on rigid cells. In *Proc. of Eurographics*, 339–347.

CABRAL, M., LEFEBVRE, S., DACHSBACHER, C., AND DRETTAKIS, G. 2009. Structure preserving reshape for textured architectural scenes. In *Proc. of Eurographics*, 469–480.

Figure 10: iWIRES *workshop. The original model is rendered in green, and the edited results in blue.*

COLEMAN, T., AND LI, Y. 1996. An interior, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization 6*, 418–445.

COQUILLART, S. 1990. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. In *Proc. of ACM SIGGRAPH*, 187–196.

FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. *ACM Trans. Graph. 23*, 3, 652–663.

HUANG, J., SHI, X., LIU, X., ZHOU, K., WEI, L.-Y., TENG, S., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. Subspace gradient domain mesh deformation. *ACM Trans. Graph. 25*, 3, 1126–1134.

JOSHI, P., MEYER, M., DEROSE, T., GREEN, B., AND SANOCKI, T. 2007. Harmonic coordinates for character articulation. *ACM Trans. Graph. 26*, 3, #71.

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24*, 3, 561–566.

KRAEVOY, V., SHEFFER, A., COHEN-OR, D., AND SHAMIR, A. 2008. Non-homogeneous resizing of complex models. *ACM Trans. Graph. 27*, 5, #111.

LEE, Y., AND LEE, S. 2002. Geometric snakes for triangular meshes. In *Proc. of Eurographics*, 229–238.

LIPMAN, Y., SORKINE, O., LEVIN, D., AND COHEN-OR, D. 2005. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. 24*, 3, 479–487.

LIPMAN, Y., COHEN-OR, D., GAL, R., AND LEVIN, D. 2007. Volume and shape preservation via moving frame manipulation. *ACM Trans. Graph. 26*, 1.

LIPMAN, Y., LEVIN, D., AND COHEN-OR, D. 2008. Green coordinates. *ACM Trans. Graph. 27*, 3.

MASUDA, H., AND OGAWA, K. 2007. Application of interactive deformation to assembled mesh models for CAE analysis. In *ASME Int. Design Engineering Technical Conferences*.

MASUDA, H., YOSHIOKA, Y., AND FURUKAWA, Y. 2007. Preserving form features in interactive mesh deformation. *Computer Aided Design 39*, 5, 361–368.

MILLIRON, T., JENSEN, R. J., BARZEL, R., AND FINKELSTEIN, A. 2002. A framework for geometric warps and deformations. *ACM Trans. Graph. 21*, 1, 20–51.

MITRA, N. J., GUIBAS, L., AND PAULY, M. 2006. Partial and approximate symmetry detection for 3D geometry. *ACM Trans. Graph. 25*, 3, 560–568.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. FiberMesh: Designing freeform surfaces with 3D curves. *ACM Trans. Graph. 26*, 3, 41.

OHTAKE, Y., BELYAEV, A., AND SEIDEL, H.-P. 2004. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph. 23*, 3, 609–612.

ORZAN, A., BOUSSEAU, A., WINNEMÖLLER, H., BARLA, P., THOLLOT, J., AND SALESIN, D. 2008. Diffusion curves: a vector representation for smooth-shaded images. *ACM Trans. Graph. 27*, 3.

PAULY, M., MITRA, N. J., WALLNER, J., POTTMANN, H., AND GUIBAS, L. 2008. Discovering structural regularity in 3D geometry. *ACM Trans. Graph. 27*, 3, #43, 1–11.

POPA, T., JULIUS, D., AND SHEFFER, A. 2007. Interactive and linear material aware deformations. *Proc. of Shape Modeling International 13*, 1, 73–100.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Proc. of ACM SIGGRAPH*, 151–160.

SHI, X., ZHOU, K., TONG, Y., DESBRUN, M., BAO, H., AND GUO, B. 2007. Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph. 26*, 3.

SINGH, K., AND FIUME, E. 1998. Wires: a geometric deformation technique. In *Proc. of ACM SIGGRAPH*, 405–414.

SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Proc. of Sym. on Geometry Processing*, 109–116.

SORKINE, O., LIPMAN, Y., COHEN-OR, D., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proc. of Sym. on Geometry Processing*, 179–188.

SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded deformation for shape manipulation. *ACM Trans. Graph. 26*, 3.

VON FUNCK, W., THEISEL, H., AND SEIDEL, H.-P. 2006. Vector field based shape deformations. *ACM Trans. Graph. 25*, 3.

ZHOU, K., HUANG, J., SNYDER, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2005. Large mesh deformation using the volumetric graph Laplacian. *ACM Trans. Graph. 24*, 3, 496–503.

ZIMMERMANN, J., NEALEN, A., AND ALEXA, M. 2007. SilSketch: automated sketch-based editing of surface meshes. In *Proc. of Sketch-based Interfaces and Modeling*, 23–30.

Appendix

Here are the details of the nonlinear optimization of a compound wire (Section 5). We define the following energy terms to measure the deviation of W^C from the properties of the sub-wires of W^R:

- E_{len} for each cluster of equal length sub-wires, we compute the standard deviation of the lengths of the sub-wires, to penalize deviation from the equal length property.
- E_{ang} for each cluster of equal connection angles between adjacent sub-wires, we compute the standard deviation of the angles. For the special connection angle clusters (90 or 45 degrees in our implementation) we measure the deviation from these constant values.
- E_{plan} for each cluster of parallel connections, we compute the standard deviation of the vector products $\mathbf{t}_1 \times \mathbf{t}_2$ from their mean direction.

We also define E_{dist}, the sum of distances of each sample on W^C from W^D, to penalize the deviation of W^C from W^D. We find W^C by solving the nonlinear optimization:

$$\min_{W^C} \; E_{\text{len}} + w_1 E_{\text{ang}} + w_2 E_{\text{plan}} + w_3 E_{\text{dist}} \,, \tag{3}$$

with weights $w_1 = 10^3$, $w_2 = 10^4$, $w_3 = 10^{-5}$. For the optimization we used a subspace trust-region method, which is based on the interior-reflective Newton method [Coleman and Li 1996]. Each iteration involves the approximate solution of a linear system using preconditioned conjugate gradients.