Abstraction of Man-Made Shapes

Ravish Mehra[†⋆] Qingnan Zhou[†] Jeremy Long[§] Alla Sheffer[†] Amy Gooch[§] Niloy J. Mitra[⋆‡]

[†]Univ. of British Columbia [⋆]IIT Delhi [‡]KAUST [§]Univ. of Victoria

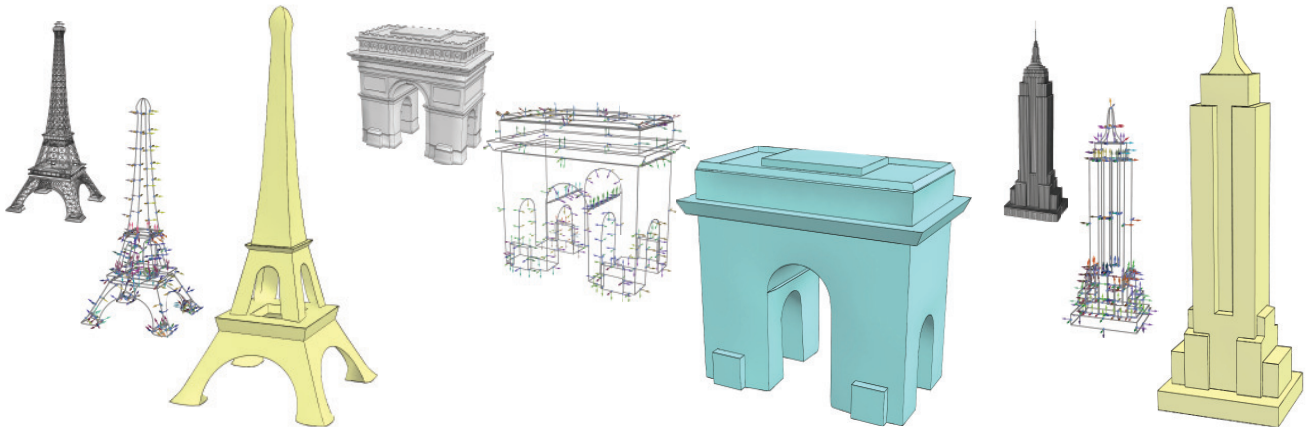

Figure 1: *Given three-dimensional models of man-made objects, often containing multiple self-intersecting components, we extract characteristic curve networks along with auxiliary normal information providing a compact* vector *representation of the main model features and use those to generate 3D abstractions capturing the essence of the inputs.*

Abstract

Man-made objects are ubiquitous in the real world and in virtual environments. While such objects can be very detailed, capturing every small feature, they are often identified and characterized by a small set of defining curves. Compact, abstracted shape descriptions based on such curves are often visually more appealing than the original models, which can appear to be visually cluttered. We introduce a novel algorithm for abstracting three-dimensional geometric models using characteristic curves or contours as building blocks for the abstraction. Our method robustly handles models with poor connectivity, including the extreme cases of polygon soups, common in models of man-made objects taken from online repositories. In our algorithm, we use a two-step procedure that first approximates the input model using a manifold, closed *envelope* surface and then extracts from it a hierarchical abstraction curve network along with suitable normal information. The constructed curve networks form a compact, yet powerful, representation for the input shapes, retaining their key shape characteristics while discarding minor details and irregularities.

Keywords: curve network, shape analysis, perception, NPR

1 Introduction

Engineered objects constitute a large fraction of the models populating virtual environments such as games, movies, or simulations. In recent years, easy access to 3D modeling tools and the rapid growth of online modeling communities have resulted in large collections of such models. Most models of man-made objects present in such collections do not satisfy the notion of "good" geometry processing models [Kraevoy et al. 2008]. They frequently consist of numerous disconnected components, have self-intersections, and lack accurate information about part junctions and interconnections. From a processing point of view, such models can at best be considered to be polygon soups (Figure 5).

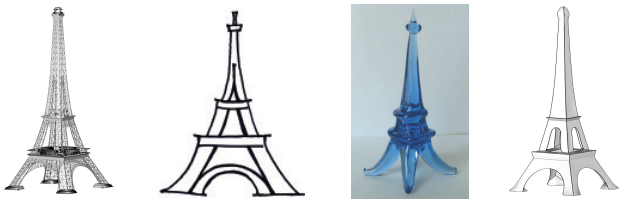

Figure 2: *(Left to right) Detailed 3D model, an artist's drawing, a crystal souvenir, and our abstraction of the Eiffel Tower.*

Models of man-made objects can be very detailed, capturing every hole and protrusion. However, such shapes are often characterized and identified by just a few defining features (Figure 2). Shape descriptions based on those features, commonly involving a handful of characteristic curves, potentially mimic the minimalist representations that we, as humans, possibly store and use for our inference needs [Poggio et al. 1985]. These compact, abstracted descriptions are visually more appealing than the detailed original ones, which may appear visually cluttered (Figure 2-left). Using this observation, artists frequently create recognizable images or icons of known objects by employing only a few brush strokes (see drawing in Figure 2). Such stylization is also common in tourist maps (Figure 3), where landmark buildings are depicted by just a few strokes that highlight their main features (see also [Grabler et al. 2008]).

In this paper, we introduce a novel algorithm for abstracting three-dimensional shapes. Inspired by human shape perception litera-

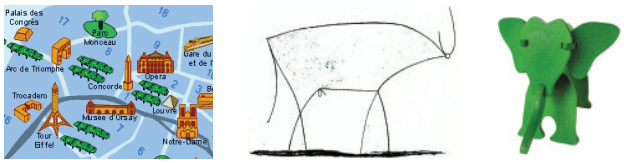

Figure 3: *(Left to right) Tourist map depicting landmark buildings, Picasso's famous abstraction of a bull (© Succession Picasso), and a wooden toy designed by Calder (© Calder Foundation) all make use of distinctive attributes to generate minimalist, yet powerful, abstractions.*

ture [Attneave 1954; Costa and Cesar 2001] and artistic techniques, we use characteristic curves or contours as building blocks for the abstraction. The choice is motivated by the observation that the shape of many man-made objects is clearly delineated by contour lines and can be faithfully modeled as a union of smooth patches welded together along these junction curves. Specifically, our method extracts a sparse network of space curves and associated normals as an abstraction of input models. This compact representation makes explicit the main features of the shape, which are challenging to identify from a polygon mesh or other low-level representations.

While shape abstraction can be attempted at the rendering level by developing suitable NPR tools, applying it directly to the models has a number of advantages. Model-level abstraction allows consistent rendering of a shape from a variety of views and is independent of the rendering resolution or zoom level. The set of extracted curves forms a minimalist representation, or an icon, of the modeled object. By maintaining the extracted curve network's properties, subsequent processing can generate new models that automatically retain the defining characteristics of the original one [Gal et al. 2009]. Analogous to the diffusion curves for images [Orzan et al. 2008], our curve network, along with suitable normal information, presents a *vectorized* representation of the input models.

Our abstraction method operates in two stages. First, it maps the given geometry to a voxel grid of suitable resolution, and extracts a corresponding closed, manifold *envelope surface*, that wraps around the input model while smoothing out minor details (Figure 7 middle). This allows us to robustly handle non-manifold meshes and multiple component meshes, including the extreme case of polygon soups. In the second stage, the method extracts a network of curves or vectors from the envelope. After extracting the network connectivity using a mesh segmentation approach, it establishes the network geometry using a combination of regularization and approximation criteria (Figure 7 right).

An alternative approach for filtering out insignificant shape details is simplification, which, operating at triangle-vertex level, aims to reduce polygon count while controlling the deviation of the simplified object from the original one. Unfortunately, in the process, characteristic shape curves are likely to be disturbed, especially under extreme simplification [Garland and Heckbert 1997], see Figure 4. Additional artifacts arise when the input is a polygon soup instead of a well-formed manifold mesh. In contrast, *abstraction* attempts to directly extract the high-level structure of objects, intentionally removing insignificant visual details and potentially allowing significant topological changes. Abstraction, by attempting to perform a higher level of shape understanding, can prove to be a better candidate for low bit budget representations where simplification results may be unsuitable.

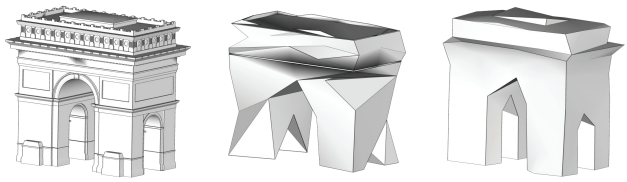

Figure 4: *(Left to right) Input model, model simplified to* 200 *triangles, simplified envelope surface with* 200 *triangles. While the simplified envelope surface fares better, it does not preserve the core features of the Arc de Triomphe.*

Abstraction, which involves shape analysis and is strongly linked to human perception and cognition, is more of a qualitative rather than a quantitative operation. Hence, when considering the degree of abstraction, or the search for the *right* abstraction of a given object, the answer often lies in the area of perception, relating to the observer's prior knowledge. Researchers have studied questions like "where do humans draw lines" to achieve similar results algorithmically [Cole et al. 2008]. Our goal is to understand shape abstractions from a human perception viewpoint. Based on the perception studies, we hypothesize that an abstracted shape should have its minor details stripped off, while retaining its major characteristic features. The emphasis on *characteristic* features implicitly links the effectiveness of abstraction with the degree of user familiarity with the object.

In addition to the main contribution of providing a method to perform abstraction of 3D geometric shapes, our two secondary contributions are: a novel vector-based representation of 3D geometry, which can be used for a variety of mesh editing tasks; and a simple yet robust mechanism for approximating polygon soup models by a manifold surface envelope.

2 Related Work

Abstraction, the process of identifying characteristic properties and extracting their mutual relationships and topology [Falcidieno and Spagnuolo 1998], has been studied in many fields and disciplines including art, non-photorealistic rendering and modeling, and human perception, for purposes such as shape analysis, generation of compact descriptors, and recognition.

Artistic and human perception of shape. In the twentieth century, artists like Kandinsky, Mondrian, and Picasso pushed the boundaries of geometric abstraction in 2D representations of the surrounding world to the extremes. Alexander Calder ingeniously used wires in addition to sheet metal, wood, and bronze to create abstract 3D sculptures. While automatically generating abstraction levels like those created by Kandinsky and Mondrian is unrealistic, we draw motivation from the curve-based abstraction portrayed by Calder in his 3D wire sculptures.

Koenderink and Doorn [1979] hypothesized that humans internally represent shapes as functions that measure the visual complexity of shapes. Later, Nackman and Pizer [1985] differentiated between representation and description of an object where an object representation contains enough information to enable an approximate reconstruction, while an object description needs only to contain

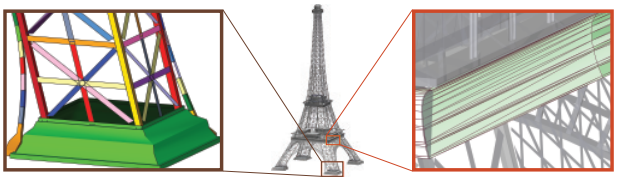

Figure 5: *Commonly available man-made models often have multiple connected components, self-intersections, and triangles with bad aspect ratios.*

enough information to identify an object as a member of some object class, which is exactly what abstraction aims to do.

Non-realistic rendering and modeling. A major goal of non-photorealistic rendering (NPR) is to highlight or amplify defining object characteristics. Since low-level geometry does not provide a natural prioritization of the shape features (Figure 6), NPR techniques strive to identify view-specific important features, which should be rendered or exaggerated to convey form [Cole et al. 2008]. Significant research has been devoted to identifying candidate feature lines including silhouettes, ridge or valley lines [Na et al. 2005], contours, and suggestive contours [DeCarlo et al. 2003]. The curve networks extracted by our abstraction method can be used to create view-independent NPR effects, which remain persistent across motion and animation.

Little work exists for stylizing or creating iconic representations of 3D models. A notable exception is the research by Gal et al. [2007], which creates 3D collages on top of target shapes using a database of objects as primitive building blocks. In a parallel thread of work, Theobalt et al. [2007] generate collages from mesh animation. The resulting collages, though artistically powerful, are not intended for other uses.

Shape analysis and reverse engineering. Motivated by procedural modeling and constructive solid geometry, researchers have long proposed to approximate a given 3D model with parametric parts [Várady and Martin 2002; Attene et al. 2006]. Such parametric descriptions enable creation of different abstraction levels by direct part manipulation, for instance by removing some of the parts while preserving others. However, most discrete digital models lack such semantic information and reverse engineering structure and regularity from 3D geometry is a difficult problem [Pauly et al. 2008], with algorithmic solutions unlikely to reach human performance levels in the near future. Instead, we present a method that uses low-level analysis of the models to automatically extract the main feature curves, providing a compact vector representation of the model at a desired abstraction level. Our approach bypasses the difficult reverse engineering task of detecting the global structure, while implicitly preserving the main characteristic features of the models.

In the domain of 2D shapes, Bengtsson et al. [1991] obtained abstractions by studying contours at different scales and recently attempts have been made to learn abstractions using a set of exemplars [Demirci et al. 2009]. Other avenues explored for abstraction include rule-based simplification [Brown et al. 1993], tracking a symbolic representation while the user constructs a parametric model [Falcidieno and Spagnuolo 1998], or topology-based inference [Biasotti et al. 2002].

While the general shape analysis problem is difficult even for manifold, connected meshes, we show that for man-made objects, due to their inherent regularity and structure, creating effective abstraction, even from polygon soups, is possible. Our abstraction comes in the form of a curve network, which can be used as input for modeling and editing systems like FiberMesh [Nealen et al. 2007] and iWires [Gal et al. 2009].

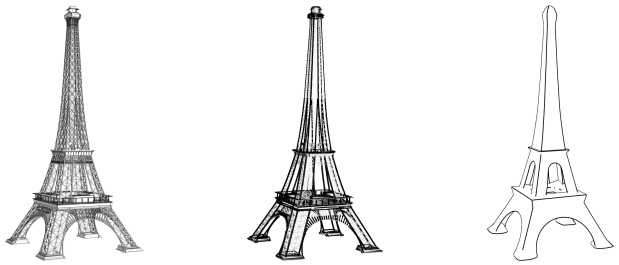

Figure 6: *Eiffel tower rendered using crease lines (left), suggestive contours (middle), and crease lines in our 3D abstraction (right).*

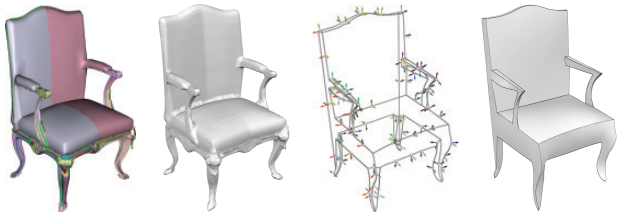

Figure 7: *(Left to right) Input model with* 353 *components, envelope surface, vector representation, and reconstructed model.*

3 Algorithm Overview

Our goal is to extract a *vector* representation for three-dimensional shapes, targeted specifically toward abstraction of man-made objects, i.e., objects whose main features can be captured by a few smooth surfaces glued together along characteristic curves. Our representation satisfies the following properties:

Reconstruction: The network of curves, or vectors, combined with the normals prescribed along them, is sufficient to define the abstracted shape, encoding both the connectivity and the geometry of the reconstructed model (Section 6). To describe the connectivity, the network is required to be a connected B-Rep representation, as this significantly simplifies the reconstruction step. To adequately reconstruct the geometry we define the surface normals across the model as weighed combinations of the curve normals.

Abstraction: Depending on the desired level of abstraction, our representation controls which geometric features are appropriate, while smoothing out the less significant ones.

Structure: Regularity and structure, which lead to simplicity of design, fabrication, and installation, are properties common to most man-made objects [Merrell and Manocha 2008]. Viewers are known to be sensitive to breakup of regular structures present in the input models when those are processed [Kraevoy et al. 2008], and they seem to remember or identify shapes based on symmetry and regularity, ignoring the deviations [Arnheim 1956]. Thus we expect abstractions of man-made shapes to be as regular or structured as possible. This translates to the curves being locally as simple as possible, i.e., being planar, linear, circular arcs, etc., while satisfying global regularity requirements such as symmetry, parallelism, and orthogonality.

Our abstraction method generates such vector representations in two steps. First, it constructs an *envelope* surface, a closed, manifold surface approximating or "enveloping" the input model (Section 4). The surface provides a manifold approximation of the potentially poorly-connected data, enabling subsequent extraction of a meaningful network of curves. It also helps to approximate the input at a desired degree of abstraction, controlling the genus of the final model as well as smoothing out minor details (Figure 7).

The second step extracts the curve network that serves as the vector representation of the input (Section 5). First we extract the network connectivity using mesh segmentation and then establish the network geometry using a combination of regularization and approximation criteria. In the geometry computation, we simultaneously optimize for the vector representation and an approximation of the reconstructed surface obtained from this representation. This controls the tradeoff between the smoothness and regularity of the reconstruction, and the level of approximation of the input.

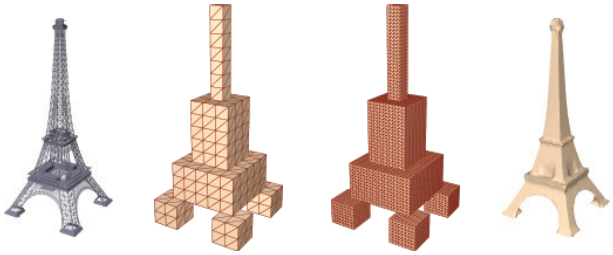

Figure 8: *(Left to right) Input model, voxel hull, subdivided voxel hull, envelope after iterative fitting.*

4 Envelope Construction

The envelope construction stage defines a tight manifold approximation of the input model, providing both a first level of abstraction and a well-defined domain for further processing. This step can be skipped if the input model is *a priori* described by a single manifold surface. However, in this case the abstracted model will preserve the topology of the input.

Man-made models in public databases are often formed as a union of overlapping components, where the hidden interior parts of the components are left intact effectively splitting the models into several closed regions. Algorithms, such as [Cohen et al. 1996], that assume well-formed manifold input cannot be applied to such polygon soups. The existence of interior surfaces prevents the use of methods that grow the surface from inside [Sharf et al. 2006], as they are likely to get stuck at a local minimum when encountering such interior surfaces. It also prevents the use of ball-pivoting methods [Bernardini et al. 1999] that essentially attempt to reconstruct the non-manifold structures. The method of Shen et al. [2004], when used in an approximating setup, does indeed satisfy most of our requirements for the envelope generation step. However, the method rounds off sharp features, thus making subsequent feature-curve extraction challenging. Our envelope generation step is simple, effective, and designed to satisfy the requirements of the subsequent abstractions steps.

We start by constructing an initial envelope that contains the input model and loosely follows its geometry. Using an ICP-like [Besl and McKay 1992] local refinement process, the envelope is attracted toward the model to obtain the desired approximation quality. The iterative fitting process helps satisfy the potentially conflicting goals of bringing the envelope mesh close to the input model while maintaining a quality triangulation of the envelope. The fitting process, which provides a tradeoff between envelope resolution and input approximation, smoothes out narrow concavities, yielding a first level of abstraction.

Initialization. We embed the input model in a regular grid and define the *voxel hull* of the model as the set of all grid voxels that intersect any of the model triangles (Figure 8). Subsequently the outer

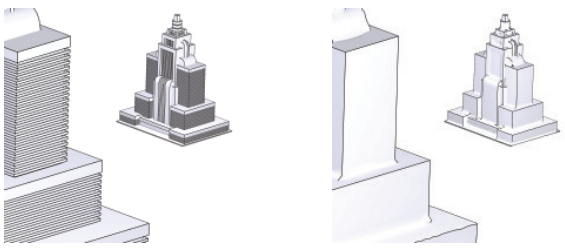

Figure 9: *The deformation step, during envelope fitting, smooths out narrow concavities and suppresses details of the input model.*

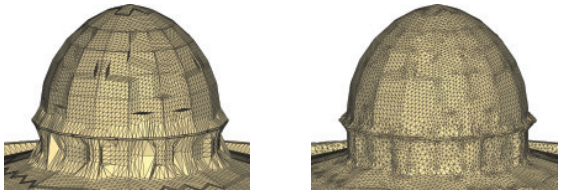

Figure 10: *Zooms of the mesh envelope, for the Dome of the Rock model, after four iterations of fitting without (left) and with (right) regularization.*

visible surface of the voxel hull is extracted as the initial envelope. While it is conceivable to have an implicit, distance function based approach followed by isosurface extraction for initializing the envelope [Bischoff et al. 2005], such an approach is ill-suited for our setup, as our input models often contain many self intersections and may have inconsistent triangle orientation, making distance computations problematic. The resolution of the grid controls the topology, specifically the genus, of the final envelope, and is defined by the user. Once the envelope is constructed we refine the initial coarse mesh using one or two iterations of regular subdivision to enable better subsequent approximation.

Iterative fitting. The fitting process iteratively pulls the envelope toward the input model while preserving the overall shape and mesh quality. Each iteration consists of the following steps.

- *Matching*, which maps the vertices of the envelope to the closest positions on the input model;
- *deformation*, which deforms the envelope to better approximate the input based on the computed matches;
- and *mesh regularization*, which maintains the quality of the envelope mesh as it deforms.

The process terminates when the envelope stabilizes or a maximum number of iterations is reached. We now describe the steps in detail.

Matching. Each vertex of the envelope is matched to the closest position on the input model. Similar to ICP setup, we discard outlier matches, i.e., ones where the distance from the envelope to the model is significantly larger than the average distance.

Deformation. The target positions provided by the correspondences established in the matching step can be inconsistent, leading to self-intersections and other mesh degeneracies. To avoid such artifacts we introduce a deformation formulation that provides a tradeoff between enforcing the matches and maintaining the shape and position of the current envelope. Specifically we optimize

$$\min_{\mathbf{v}_i} \sum_i c_1 \|(\mathbf{v}_i - \frac{1}{|(i,j)|} \sum_{e=(i,j)} \mathbf{v}_j) - \mathbf{l}_i\|^2 + c_2 \|\mathbf{v}_i - \mathbf{v}_i'\|^2 + c_3 \|\mathbf{v}_i - \mathbf{w}_i\|^2 \quad (1)$$

where $\mathbf{v}_i$ are the new positions of the mesh vertices, $\mathbf{v}_i'$ are the current vertex positions, $\mathbf{l}_i$ are the Laplacian vectors [Sorkine et al. 2004] given by $(\mathbf{v}_i' - \sum_{e=(i,j)} \mathbf{v}_j'/|(i,j)|)$ computed on the current envelope, and $\mathbf{w}_i$ are the positions proposed by the matching. The first two terms preserve the current envelope shape, effectively controlling the speed at which the envelope is pulled toward the input model, penalizing self-intersections and other artifacts. We used $c_1 = 1$, $c_2 = 0.5$, $c_3 = 2$ for all the models shown in the paper. The resulting linear system is solved using a sparse linear solver [Toledo et al. 2003]. The shape preservation terms combined with the regularization step help smooth over narrow concavities in the input model where the size of the narrow features is smaller than the initial grid resolution (see Figure 9). This aids the abstraction process by removing potentially deep, but poorly visible, features.

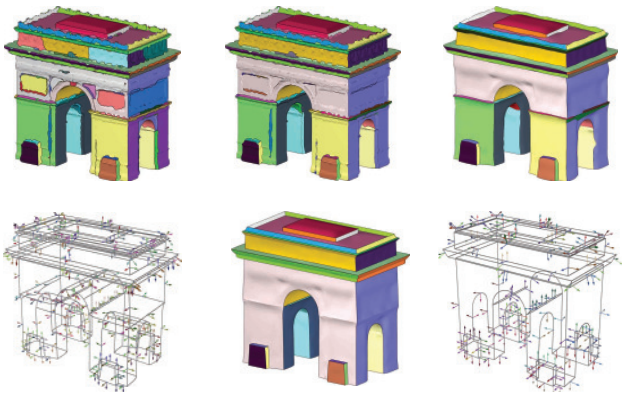

Figure 11: *Vectorization stages: (Left to right) VSA segmentation, segmentation after boundary improvement, smooth approximation geometry, extracted regularized curve network, surface after hierarchical simplification, regularized simplified curve network.*

Mesh Regularization. As the envelope deforms, the quality of the mesh triangles can increasingly deteriorate as the initial connectivity may not accurately reflect the fitted geometry. Such mismatching connectivity can negatively affect the quality of the approximation (Figure 10). Our iterative fitting addresses this concern by performing mesh regularization after every iteration. The basic regularization step is based on [Surazhsky and Gotsman 2003] and involves a combination of local mesh smoothing, edge collapses, and edge flip operations aimed at maintaining reasonable mesh quality. In addition to remeshing for improving the mesh quality, in later iterations of the envelope contraction we use heuristic remeshing operations to improve the approximation quality. Specifically, we flip edges if the mid-point of the flipped edge is closer to the input model than that of the current edge. If the flip creates poorly shaped triangles, we split the flipped edge at the midpoint. The order of flipping is determined by the approximation improvement amount, i.e., the edge for which the improvement is largest gets flipped first.

The resulting envelope surfaces are manifold triangle meshes with fair mesh quality that approximate both the topology and the geometry of the input model up to a desired resolution. The envelope generation step fills up small through-holes, smooths out narrow concavities, and removes self-intersections.

5 Vectorization

Having generated an envelope surface we now proceed to extract the network of curves and associated normals sufficient to reconstruct an abstracted replica of the envelope and hence the input shape. To avoid storing unnecessary geometric information, we distinguish between two types of curves in our network: *regular* curves that have a geometric definition (positions and normals) along the curve, and *virtual* or *connectivity-only* curves that have no such information and which are used solely to define the connectivity of the network.

The network is extracted in two steps. We first extract the network connectivity using a mesh segmentation mechanism. We then finalize the curve geometry using smoothing and regularization enforcing spatial, relational, and metric constraints (Figure 11).

5.1 Extracting Network Topology

We recast the network extraction problem as one of mesh segmentation, with the curve network defined as the set of chart boundaries. Explicitly aiming at a segmentation that satisfies the reconstruction and approximation criteria in Section 3 is likely to be rather time consuming, since even evaluating a chart quality would be computationally expensive. Instead we opt for a simpler, conservative approach, requiring charts to be relatively planar. Clearly any chart that satisfies a near-planarity requirement would satisfy the other two. Though alternative criteria such as chart developability [Julius et al. 2005] are likely to work as well, the advantage of the planarity criterion is the simplicity and speed of the resulting method. While this approach may initially result in an over-segmented network, the number of charts is later reduced when the network is hierarchically simplified (Section 5.3).

We employ the Variational Segmentation Algorithm (VSA) [Cohen-Steiner et al. 2004] to obtain the near-planar segmentation, using the approximation error to control the number of charts. We start VSA with an initial number of charts (typically just one) and measure the total approximation error. If the error is above the user-prescribed threshold we add another chart, using the triangle with the maximal error as the seed and repeat the process. The segmentation is improved using a standard post-processing step [Julius et al. 2005] of straightening boundaries and merging small charts with their neighbors, while bounding the per-chart error (we allow up to 10% increase in per-chart error for straightening and 20% for merging).

In the last step, to create a connected network, we split charts with multiple boundary loops into simple ones using a bottom-up triangle clustering. The new boundaries are defined as connectivity-only as they are unnecessary from a geometry point of view.

5.2 Extracting Network Geometry

We expect the reconstructed surface to consist of smooth charts bounded by the prescribed curve network. Our goal is to define positions and normals along the curves (c.f. diffusion curves [Orzan et al. 2008]), that enable such a reconstruction. We achieve this using an optimization that simultaneously edits the chart boundaries and the chart geometry such that the optimized charts reflect the reconstructed geometry. The optimization should balance the following aspects:

- **Surface Smoothness:** The normals across each surface chart should change smoothly. This ensures that the reconstruction using the normals along the chart boundaries will approximate the optimized charts.
- **Approximation:** The optimized and hence the reconstructed surface should remain close to the original one.
- **Curve Smoothness:** The boundary curves should be smooth. This helps to regularize the curve network and thus the subsequent reconstruction result.

Normals being a non-linear function of the vertex positions, an optimization function aiming to simultaneously satisfy the above goals would be challenging to minimize. Instead we use a solution that decouples the normals and the positions, splitting the solution into three steps: normal solve, per-triangle vertex positioning, and, finally, global assembly.

Normal Solve: We first compute new triangle normals which provide a tradeoff between smoothness and normal-level approximation of the envelope surface,

$$\min_{\{\mathbf{n}_i\}} \sum_i \|\mathbf{n}_i - \frac{1}{|N(i)|} \sum_{j \in N(i)} \mathbf{n}_j\|^2 + \omega_1 \sum_i \|\mathbf{n}_i - \mathbf{n}_i'\|^2 \quad (2)$$

where i indexes mesh triangles, $N(i)$ denotes the set of triangles adjacent to i that belong to the same chart, and $\mathbf{n}_i$ and $\mathbf{n}_i'$ are the new and current triangle normals, respectively. We set ω_1 to 0.25 for interior chart triangles, and to 0.1 for triangles adjacent to chart

boundaries. We use a smaller weight for boundary triangles since their envelope normals often deviate from those of the input models and hence are less critical to preserve (Figure 11 top, center).

Per-Triangle Solve: Next we compute the vertex positions that generate the desired normals while staying close to their original positions. The computation is performed on a per-triangle basis.

- For interior triangles we solve

$$\min_{\{\mathbf{v}_k\}} \sum_k \|\mathbf{v}_k - \mathbf{v}'_k\|^2 + \omega_2(\mathbf{n}_i\mathbf{v}_k + d_i)^2,$$

  where k indexes the three triangle vertices and d_i is the unknown distance component of the triangle's plane equation (normal form). The weight ω_2 is set to 1000, effectively ensuring that the new positions $\mathbf{v}_k$ satisfy the desired normal $\mathbf{n}_i$.
- For boundary triangles, we incorporate an additional boundary curve smoothness requirement. For every boundary edge $\mathbf{e}_b = \overrightarrow{\mathbf{v}_b^1\mathbf{v}_b^2}$ of the triangle, we add an additional component to the minimization:

$$\min_{\{\mathbf{v}_k\}} \sum_k \|\mathbf{v}_k - \mathbf{v}'_k\|^2 + \omega_2(\mathbf{n}_i\mathbf{v}_k + d_i)^2 + \omega_3 \sum_b \|\mathbf{e}_b - \mathbf{s}_b\|^2,$$

  where $\mathbf{s}_b$ is the smoothed boundary edge computed by averaging the edge vectors in the local neighborhood along the boundary using the current mesh. We used a neighborhood size of 1.5x the average edge length in the mesh and ω_3 set to 2 in all our examples.

Global Assembly: Finally, to obtain a connected mesh we reconcile the different per-triangle positions computed for each vertex. At this stage the shape and orientation of each individual triangle can be viewed as optimal, hence we aim for new vertex positions such that the per-triangle transformation gradient is close to identity. We express this requirement using a similar formulation to [Sumner and Popović 2004]. For each triangle we define $V_k = [\mathbf{v}_3 - \mathbf{v}_0, \mathbf{v}_3 - \mathbf{v}_1, \mathbf{v}_3 - \mathbf{v}_2]$ where $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$ are the three vertices of the triangle computed in the previous step and $\mathbf{v}_3 = \mathbf{v}_0 + \mathbf{n}_i$. We define W_k similarly but using the unknown shared vertex positions $\mathbf{w}$. We optimize over $\mathbf{w}$ using

$$\min_{\{\mathbf{w}_j\}} \sum_i ||W_i V_i'^{-1} - I||_F^2 + \omega_4 \sum_j \|\mathbf{w}_j - \mathbf{v}'_j\|^2$$

where F denotes the Frobenius norm, j indexes mesh vertices, and ω_4 set to 10. We iterate the steps until the positions and normals converge, typically in three to four iterations. After each iteration we perform topological cleanup, reapplying the segmentation post-processing step (Section 5.1).

Finally, we associate per-chart normals with curve vertices, using the average normal of the adjacent triangles around the vertex in each chart (see Figures 11 and 12). Notice that points have a pair of normals associated with them.

5.3 Network Regularization and Simplification

The obtained curve network satisfies the abstraction and reconstruction criteria (Section 3). We now further regularize the network to capture structure, and then simplify it by removing appropriate curves, producing a sequence of abstractions.

Regularization. The vector representation we obtain at this stage is sufficient to faithfully reconstruct an approximation of the input. However, our goal is to create abstractions for man-made objects, highlighting regularity and structure, while ignoring minor deviations or inconsistencies. Hence, we regularize individual curves and loops of the network, and enhance their mutual relations (see also [Gal et al. 2009]).

Our *local regularization* replaces near-regular geometric features by perfectly regular ones, in an effort to retain the essence of shapes while ignoring small variations. The process considers both positions and normals across the network. First near-planar curves are made exactly planar using a least squares plane fit. Curves, once optimized, are kept fixed during the later iterations, as side constraints for least squares optimization. Subsequently, near-linear curves are converted to line segments. Similarly, curves that fit well to circular arc segments are regularized to perfect circular arcs. Such tests are incremental, where once a curve or a sequence of curves fit into one of these categories, we incrementally check if consecutive curves sharing the same boundary loop with the current sequence also fit the same plane, line, or arc. We use a similar incremental approach to detect sequences of curves with nearly identical normal values along a shared face loop, indicating a locally near planar face region and make the normals identical. Standard least squares fitting solutions are employed, and considered acceptable if the residual error falls below threshold margin (0.02 and 0.1 for fine and coarse, respectively). Normals are projected accordingly: for example, when planarizing curves, the normals are projected to the least squares plane, and re-normalized.

In man-made or engineered objects, symmetry and regularity play a dominant role due to aesthetic considerations, as well as convenience in design and manufacturing. A good abstraction should capture such relations, making them explicit. In the *global regularization* step we enforce mutual relations between curve pairs that local processing may have missed. First, planar loop pairs that are nearly parallel (or orthogonal) are made exactly so. Such an approach clearly depends on the order in which the loops are processed. As a (heuristic) solution, we process the loop pairs in a greedy fashion (i.e., the loop-pair, which is closest to being parallel, is processed first), taking small steps towards making the loops parallel (or orthogonal), and iterate until convergence. Most of the examples presented in the paper converged in less than five rounds. Finally, we detect global reflective and (discrete) rotational symmetry in the input model [Mitra et al. 2006]. When detected, we enforce symmetry in the curve network, minimally moving the curves to make them exactly symmetric using a symmetrization approach [Mitra et al. 2007]. In the regularization steps, we only update positions and normals without changing connectivity or topology of the network.

Hierarchical Simplification. Our original method for network topology extraction is fairly conservative, often resulting in more charts than strictly necessary for reconstruction purposes. Hence it is often possible to simplify the network further without compromising the abstraction quality. Simplification can also be used to create higher levels of abstraction. To define the hierarchical simplification mechanism we reuse the smoothness and approximation criteria targeted during network geometry extraction.

We compute a per-curve simplification error measuring the impact of removing the curve from the network using two terms. We use integral dihedral angle along a curve to measure smoothness error. To quickly estimate the approximation or reconstruction error, we integrate the deviation of the average normal along the curve from the arc-length interpolation of the curve end-point normals at that point. This measure provides a rough estimate of the difference between the reconstructed surfaces with and without the curve in question. To control the simplification process we set separate thresholds on the two terms.

During simplification, rather than removing curves completely, we simply demote them to connectivity-only discarding all the geometric information associated with them (Figure 12). Leaving the net-

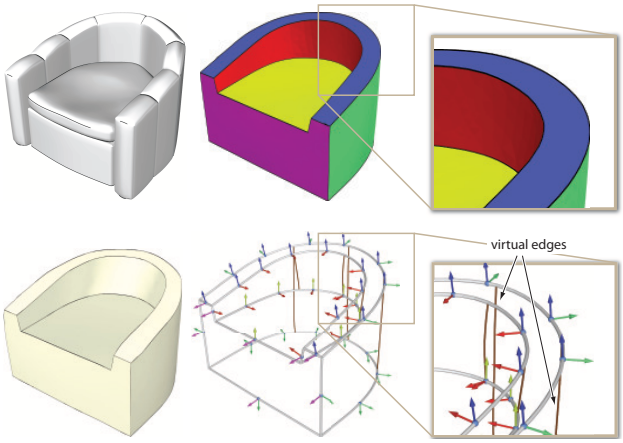

Figure 12: *Input model, processed segments, vectorized curve network, and reconstructed abstraction. Zoom panels show section of curve network, and normals along the curves. Connectivity-only or virtual edges are marked in brown. For ease of visualization, normals from same curve loops are marked in identical colors.*

work topology untouched, helps the reconstruction process while the overhead of storing connectivity-only curves is negligible. In order to maintain reconstruction consistency, once a curve is demoted, we update the curve network, re-assigning normals along the affected edges using our chart optimization (see Section 5.2) and regularization procedures. Recomputing the surrounding geometry after each edge removal is expensive. However, the number of curves in our curve networks is small, allowing such expensive iterations. Thus we get multiple curve networks and associated reconstructions or abstractions of the input (see Figure 11).

6 Reconstruction

The reconstruction process constructs the abstracted model's shape from the vector representation (Figure 7 right), establishing both the underlying mesh connectivity and the actual mesh geometry.

Initial Reconstruction. The reconstruction starts by triangulating the faces of the curve network, aiming to construct a connectivity that reflects the target geometry. It first embeds the boundary of each face in the plane, using Multi-Dimensional Scaling (MDS) [Kruskal and Wish 1978] aiming to preserve the shape of the boundary. The MDS embedding best preserves the Euclidean distances between all pairs of vertices on the boundary, which in case of planar boundary loops results in preservation of the boundary shape and even for non-planar boundaries generates a reasonable embedding. As MDS allows for self intersections, we explicitly check for them. When an intersection is observed, we embed the loop on a circle using arc-length parameterization. Once a planar boundary is computed, its interior is triangulated [Shewchuk 1996], providing the desired connectivity reconstruction.

To generate the geometry of the abstracted model we deform the computed planar meshes, enforcing the boundary vertex positions and normals prescribed by the input vector representation. We use the shape preserving deformation method of Popa et al. [2006] whose rotation propagation approach is well suited to our setting. The deformation is applied simultaneously to clusters of faces connected with connectivity-only network curves as we expect the normal impact to propagate across them.

Improvement. Although for many surfaces the initial reconstruction produces satisfactory results, it is not always the case. First, if the normal differences across a reconstructed face are fairly large, our deformation may "under-rotate" the triangles involved as it satisfies the normal constraints only in a least-squares sense. This problem is easily resolved by iterating through the deformation step using the current surface as the undeformed model.

Second, if the initial embedding of the boundary differs significantly from the "natural" parameterization of the corresponding face, the mesh triangles can undergo significant deformation leading to visible artifacts. This is resolved by repeating the connectivity and geometry construction steps. Since we now have an interior triangulation, we use standard mesh parameterization methods to find the optimal planar domain, e.g ABF++ [Sheffer et al. 2005]. We then retriangulate the obtained boundary and repeat the deformation as above.

7 Results and Discussion

We tested our abstraction algorithm on a variety of man-made models mostly described by polygon soups. The abstracted models are generated using a combination of local processing with subsequent consolidation of global relations common in man-made forms. The abstracted shapes, having highlighted the mutual relations of parts or curves, are significantly more compact and lightweight compared to any low level representation. Here we take advantage of the observation that engineered forms are often defined by a few 1D curves, having relatively low information content compared to organic objects. The resulting models are extremely concise, while explicitly encoding information about relations between parts (see Table 1). For the models shown in the paper, the average time taken for generating an abstraction was less than 2 minutes on a 3GHz machine with 2GB RAM, with the envelope surface creation phase taking the majority of computing time.

Figures 13 and 14 showcase abstraction results for various models of man-made buildings, furniture, and equipment of varying level of complexity at high and low-resolution abstractions. After prescaling the objects to a unit box, the abstraction level can be controlled using the following parameters: the voxel size (0.01 and 0.05 for fine and coarse, respectively), VSA approximation error (0.05 and 0.1, respectively), regularization thresholds (deviations from straight lines, circles, etc.). The perceptual effect of the parameters, in terms of their visual impact, is harder to quantify, and we plan to explore this relationship using a user study in the future.

Figure 15 indicates how the abstraction result degrades with increasing noise and ambiguity in the data. Since the input models consist of many disconnected components, little is achieved by local smoothing. In contrast, the abstraction results, which enforce local and global regularization, are stable and capture the charac-

Model	# triangles	# components	level	#lines	# arcs	# curves
Empire State	16k	17	low	34	0	4
			high	146	2	4
Eiffel Tower	15.6k	2417	low	47	19	19
			high	85	31	24
building	3.7k	89	low	70	0	12
			high	288	2	9
Arc-Triomphe	13k	8	low	127	2	10
			high	163	16	14
Pagoda	37k	1173	low	64	0	1
			high	54	0	22
dome	3.8k	2	low	24	2	0
			high	116	23	6
rocking chair	32k	22	low	26	33	39
			high	55	35	32
baroque chair	164k	353	low	25	25	16
			high	20	33	42

Table 1: *Abstraction statistics.*

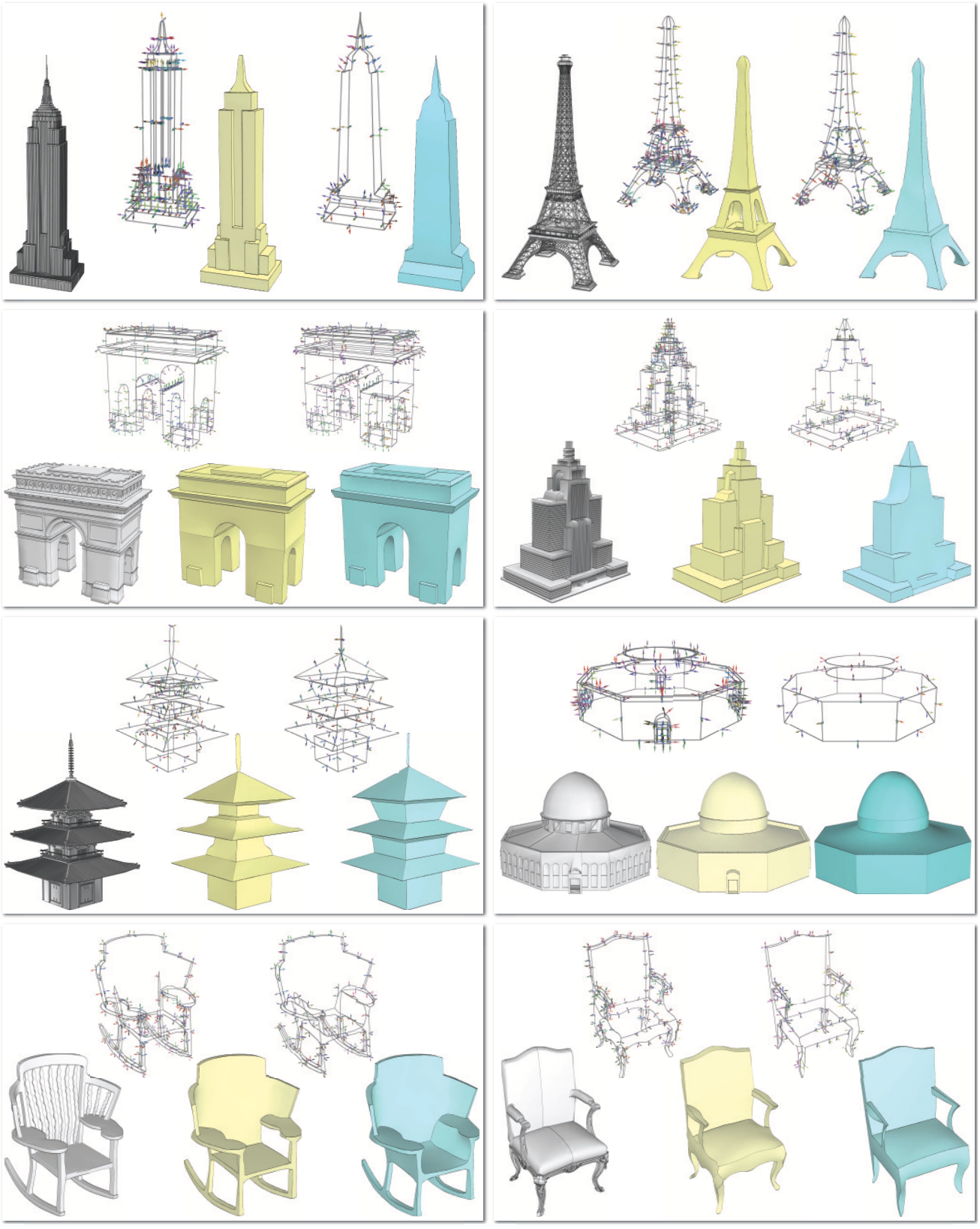

Figure 13: *Result gallery showing various input models, extracted curve networks (with normals), and reconstructed abstractions. The high-resolution abstractions are rendered in yellow, while the low-resolution ones are in blue.*

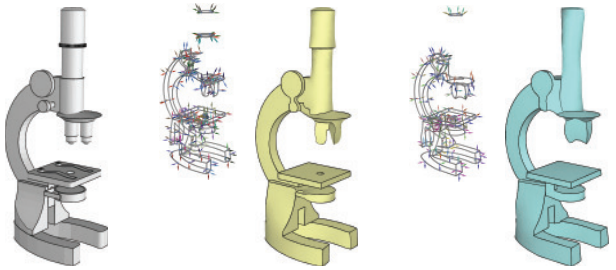

Figure 14: *Abstractions of a microscope model at high- and low-resolutions. For such models with fine details, the abstraction results are often subtle, while geometrically minor but semantically significant parts can get suppressed by our purely geometric abstraction procedure.*

teristic features. More specifically, we achieve noise resistance via normal smoothing, topological cleanup, and regularization.

We construct the network topology in two main stages: VSA segmentation and hierarchical simplification. Thus we have the option of balancing or shifting the algorithmic sophistication between these steps. We could indeed make the segmentation step more sophisticated, using higher level primitives for advanced VSA techniques (e.g. [Wu and Kobbelt 2005; Attene et al. 2006; Yan et al. 2006]). However, given our hierarchical simplification step we found this unnecessary.

It has been hypothesized that abstract shapes are easier to recognize and classify compared to detailed ones [Hou and Ramani 2008]. Besides adding highlights for NPR applications, curves or feature lines are also used for detecting partial symmetries [Bokeloh et al. 2009], for shape editing [Gal et al. 2009], etc. Such methods usually work on manifold, connected geometric models or rely on edges with sharp dihedral angles to identify feature edges, and thus cannot be readily applied to polygon soups or noisy models. Our abstraction procedure, which robustly generates a curve network capturing the high-level features of the model while ignoring finer details or deviations, provides an abstraction network that can be used as auxiliary information by the above mentioned algorithms.

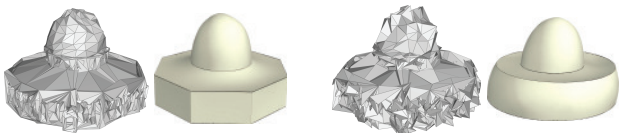

Figure 15: *Even for noisy input models, the regularization step allows creation of quality abstractions. We present abstractions with 2% and 5% (of bounding box diagonal length) noise added to the dome model.*

Limitations. Our method's ability to remove topological details is currently strongly linked to the resolution of the envelope. Once the envelope separates features like the fine diagonal bars in the water tower in Figure 16, they persist throughout the algorithm stages, independent of size. Topology level abstraction that removes such features is an important topic to address in future research.

On a more perceptual note, not all objects have recognizable abstract representations. For instance, the abstraction of the Burj Al-Arab building (see Figure 16) may not be identified by all.

8 Conclusion

We presented an algorithm for creating abstractions of 3D geometric shapes, specifically targeting man-made objects, using hierar-

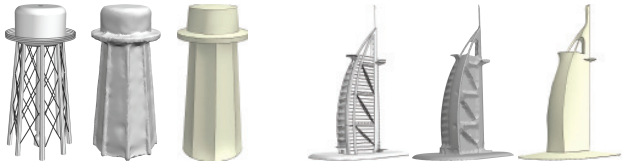

Figure 16: *(Left) Fine topological features are easily combined by our envelope construction stage. However, once such features are extracted by a finer grid resolution, we have no easy method to remove them, independent of their size. (Right) Some objects, perhaps those less familiar to us, have no obvious natural abstraction.*

chical curve networks that capture the defining characteristics of the inputs. Our method is robust, designed to handle models with disconnected components, self-intersections and noise, and uses a regularization step to make specific both individual and mutual curve relations. The curve networks, along with the associated normal information, can be used to reconstruct clean manifold meshes retaining the *essence* of the input forms. We believe that the extracted curve network provides a powerful handle to the input shape, since the representation contains specific and explicit global information about the shape of object parts and the relations between them.

Acknowledgements

This work was partially supported by Adobe Inc., MITACS NCE, and the NSERC discovery program. Niloy was supported by a Microsoft Outstanding Young Faculty Fellowship. We thank Karan Singh for the simulating discussions, and the anonymous reviewers for their helpful suggestions. We would also like to thank Tiberiu Popa for help with Graphite and MAMD code, Vladislav Kraevoy for providing the CML code, and finally Xi Chen, Benjamin Cecchetto and Derek Bradley for helping in the production of the supplementary video. The 3D models were collected from the artist-3d, Google 3D warehouse, Princeton Benchmark, and TurboSquid.

References

ARNHEIM, R. 1956. *Art and Visual Perception: A Psychology of the Creative Eye*. Faber and Faber.

ATTENE, M., FALCIDIENO, B., AND SPAGNUOLO, M. 2006. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer 22*, 3, 181–193.

ATTNEAVE, F. 1954. Some informational aspects of visual perception. *Psychological review 61*, 3, 183–193.

BENGTSSON, A., AND EKLUNDH, J.-O. 1991. Shape representation by multiscale contour approximation. *IEEE Trans. on PAMI 13*, 1 (Jan), 85–93.

BERNARDINI, F., MITTLEMAN, J., RUSHMEIER, H., SILVA, C., AND TAUBIN, G. 1999. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. on Visualization and Computer Graphics 5*, 4, 349–359.

BESL, P. J., AND MCKAY, N. D. 1992. A method for registration of 3D shapes. *IEEE Trans. on PAMI 14*, 2, 239–256.

BIASOTTI, S., FALCIDIENO, B., AND SPAGNUOLO, M. 2002. Shape abstraction using computational topology techniques. *From geometric modeling to shape modeling*, 209–222.

BISCHOFF, S., PAVIC, D., AND KOBBELT, L. 2005. Automatic restoration of polygon models. *ACM Trans. Graph. 24*, 4.

BOKELOH, M., BERNER, A., WAND, M., SEIDEL, H.-P., AND SCHILLING, A. 2009. Symmetry detection using line features.

Computer Graphics Forum, Proc. of Eurographics 28, 2, 697–706.

BROWN, G., FORTE, P., MALYAN, R., AND BARNWELL, P. 1993. A non-linear shape abstraction technique. In *CAIP*, 223–230.

COHEN, J., VARSHNEY, A., MANOCHA, D., TURK, G., WEBER, H., AGARWAL, P., BROOKS, F., AND WRIGHT, W. 1996. Simplification envelopes. In *Proc. SIGGRAPH*, 119–128.

COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational shape approximation. *ACM SIGGRAPH Trans. Graph.*, 905–914.

COLE, F., GOLOVINSKIY, A., LIMPAECHER, A., BARROS, H. S., FINKELSTEIN, A., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2008. Where do people draw lines? *ACM SIGGRAPH Trans. Graph. 27*, 3, #88, 1–11.

COSTA, L., AND CESAR, R. M. 2001. *Shape Analysis and Classification: Theory and Practice*. CRC Press.

DECARLO, D., FINKELSTEIN, A., RUSINKIEWICZ, S., AND SANTELLA, A. 2003. Suggestive contours for conveying shape. *ACM SIGGRAPH Trans. Graph. 22*, 3 (July), 848–855.

DEMIRCI, M., SHOKOUFANDEH, A., AND DICKINSON, S. J. 2009. Skeletal shape abstraction from examples. *IEEE Trans. on PAMI 31*, 5, 944–952.

FALCIDIENO, B., AND SPAGNUOLO, M. 1998. A shape abstraction paradigm for modelling geometry and semantics. In *Computer Graphics International*, 646–656.

GAL, R., SORKINE, O., POPA, T., SHEFFER, A., AND COHEN-OR, D. 2007. 3D collage: expressive non-realistic modeling. In *Proc. of NPAR*, ACM, New York, NY, USA, 7–14.

GAL, R., SORKINE, O., MITRA, N. J., AND COHEN-OR, D. 2009. iWIRES: An analyze-and-edit approach to shape manipulation. *ACM SIGGRAPH Trans. Graph. 28*, 3, #33, 1–10.

GARLAND, M., AND HECKBERT, P. S. 1997. Surface simplification using quadric error metrics. In *Proc. SIGGRAPH*, 209–216.

GRABLER, F., AGRAWALA, M., SUMNER, R. W., AND PAULY, M. 2008. Automatic generation of tourist maps. *ACM SIGGRAPH Trans. Graph. 27*, 3, 1–11.

HOU, S., AND RAMANI, K. 2008. Structure-oriented contour representation and matching for engineering shapes. *Computer Aided Design 40*, 1, 94–108.

JULIUS, D., KRAEVOY, V., AND SHEFFER, A. 2005. D-charts: Quasi-developable mesh segmentation. In *Computer Graphics Forum, Proc. of Eurographics*, vol. 24, 581–590.

KOENDERINK, J. J., AND VAN DOORN, A. J. 1979. The internal representation of solid shape with respect to vision. *Journal Biological Cybernetics 32*, 4, 211–216.

KRAEVOY, V., SHEFFER, A., SHAMIR, A., AND COHEN-OR, D. 2008. Non-homogeneous resizing of complex models. *ACM SIGGRAPH Trans. Graph. 27*, 5, 1–9.

KRUSKAL, J. B., AND WISH, M. 1978. Multidimensional scaling. *Sage University Paper series on Quantitative Application in the Social Sciences 07-011*.

MERRELL, P., AND MANOCHA, D. 2008. Continuous model synthesis. *ACM Trans. Graph. 27*, 5, 1–7.

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

MITRA, N. J., GUIBAS, L., AND PAULY, M. 2007. Symmetrization. In *ACM SIGGRAPH Trans. Graph.*, vol. 26, #63, 1–8.

NA, K., JUNG, M., LEE, J., AND SONGA, C. G. 2005. Redeeming valleys and ridges for line-drawing. In *Advances in Mulitmedia Information Processing*, 327–338.

NACKMAN, L., AND PIZER, S. 1985. Three-dimensional shape description using the symmetric axis transform. *IEEE Trans. on PAMI 7*, 2, 187–201.

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

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

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

POGGIO, T., TORRE, V., AND KOCH, C. 1985. Computational vision and regularization theory. *Nature 317*, 314 – 319.

POPA, T., JULIUS, D., AND SHEFFER, A. 2006. Material-aware mesh deformations. In *SMI*, 22.

SHARF, A., LEWINER, T., SHAMIR, A., KOBBELT, L., AND COHEN-OR, D. 2006. Competing fronts for coarse-to-fine surface reconstruction. *Computer Graphics Forum, Proc. of Eurographics 25*, 3, 389–398.

SHEFFER, A., LÉVY, B., MOGILNITSKY, M., AND BOGOMYAKOV, A. 2005. ABF++: fast and robust angle based flattening. *ACM Trans. Graph. 24*, 2, 311–330.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. In *ACM SIGGRAPH Trans. Graph.*, ACM Press, 896–904.

SHEWCHUK, J. 1996. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, vol. 1148. Springer-Verlag, 203–222.

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

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. In *ACM SIGGRAPH Trans. Graph.*, 399–405.

SURAZHSKY, V., AND GOTSMAN, C. 2003. Explicit surface remeshing. In *Proc. of Symp. of Geometry Processing*, 17–28.

THEOBALT, C., RÖSSL, C., DE AGUIAR, E., AND SEIDEL, H.-P. 2007. Animation collage. In *Proc. of Symp. of Computer Animation*, 271–280.

TOLEDO, S., CHEN, D., AND ROTKIN, V., 2003. TAUCS: A library of sparse linear solvers. http://www.tau.ac.il/ stoledo/taucs/.

VÁRADY, T., AND MARTIN, R. R. 2002. Reverse engineering. In *Handbook of Computer Aided Geometric Design*, G. Farin, J. Hoschek, and M. S. Kim, Eds. Springer, 651–681.

WU, J., AND KOBBELT, L. 2005. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum, Proc. of Eurographics 24*, 3, 277–284.

YAN, D. M., LIU, Y., AND WANG, W. 2006. Quadric surface extraction by variational shape approximation. In *Geometric Modeling and Processing*, 73–86.