# Automatic Line Handles for Freeform Deformation

Leïla Schemali [1,2], Jean-Marc Thiery[1] and Tamy Boubekeur[1]

[1]Telecom ParisTech - CNRS [2]Useful Progress

**Abstract**

*Interactive freeform surface deformation methods allow to explore the space of possible shapes using simple control structures. While recent advances in variational editing provide high quality deformations, designing control structures remains a time-consuming manual process. We propose a new automatic control structure generation based on the observation that the most salient visual structures of a surface, such as the one exploited in Line Drawing methods, are tightly linked to the potential deformations it may undergo. Our basic idea is to build control structures from those lines in order to provide users with an automatic set of deformation handles to grab and manipulate, avoiding the tedious task of region selection and handle positioning. The resulting interface inherits view-dependency and adaptivity from line definitions, reduces significantly the modeling session time in a number of scenarii, and remains fully compatible with classical handle-based deformations.*

## 1. Introduction

Interactive freeform deformation (FFD) tools allow to edit easily the shape of a surface mesh through a high level control structure. Typical examples of such structures include cages, skeletons and handle sets. We focus on the later, for which a number of recent variational techniques have been introduced. The basic interface for all these techniques is usually the same: the user selects a *region of interest* (ROI) which will be deformed while the remaining part of the object stays static. The ROI is decomposed in two components: a *rigid motion handle*, manipulated explicitly by the user, and an *influence area*, interpolating the handle and the static part boundaries smoothly. The critical computation workload is usually dedicated to this smooth interpolation, for which we seek for feature preservation under rotation and/or translation invariance. To achieve such a behavior, non-linear [BPGK06, SA07] and linear techniques [BS08] offer a performance gradient ranging from interactive to real time for several tens or hundreds of thousands of vertices.

With these powerful deformation tools in hands, one of the main challenges remains the definition of the handles, which is a tedious task even for expert users. We can classify the type of handles in two categories. First, handles aiming at introducing new geometrical structures on a base surface: stemming from the user wishes, they may take place in flat areas and cannot be inferred automatically. Second, handles capturing existing components that the user wants to modify. Here, we argue that important geometric components on a surface can be faithfully captured with a set of lines defined at its most salient features. Indeed, this is a phenomenon explored by *line drawing* algorithms to convey shape expressiveness with minimal data.

Therefore, we propose an automatic handle generation process for freeform deformation based on feature lines which provides controllable *line* handles and their associated influence area dynamically during mesh editing. The resulting FFD interface is view-dependent and allows to quickly edit the shape by simply grabbing and moving line handles. As the deformation is entirely performed through a constrained generic variational framework [LSLCO05], the user can switch back to classical manual mode at anytime e.g., adding individual constraints. Beyond the system, we make two technical contributions:

- a regularization procedure which converts feature lines to line handles;
- a supervised ROI generation algorithm based on a fast diffusion process.

As a result, our system provides automatic handles fitting important characteristics of the shape (e.g., main anisotropic lines, contours, etc) and providing an intuitive modeling interface for arbitrary 3D surface meshes.

## 2. Background.

**Shape Editing with Line Handles.** *Wires* [SE98] allows to place lines on a shape and to transfer the deformation from the lines to the mesh. Lines are placed manually on the surface and offer a great flexibility to design a complete modeling structure on top of the surface, less restrictive than cages for instance. However, lines have to be modeled manually.

The SilSketch [ZNA07, NSACO05] algorithm allows the user to "over-sketch" silhouettes of an object. The system finds the closest silhouette correponding to the sketched line and deforms the surface using Laplacian deformation so that its actual silhouette fits the sketched line. This system allows to quickly redesign the global outline of a shape, but is restricted to interior and exterior contours and does not allow to control the area of influence of the resulting deformation.

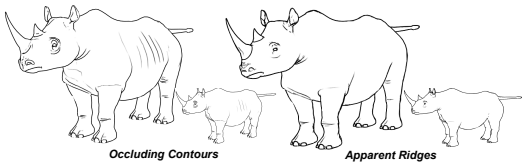



**Figure 1:** *We build upon state-of-the-art line drawing methods and their ability to capture important view-dependent visual structures to construct FFD handles automatically.*

FiberMesh [NISA07] proposes to create a mesh by drawing 2D lines. Improving over existing sketch-based modeling tools such as Teddy [IMT99], the surface is defined through a variational framework from positional and curvature constraints located at the sketched lines. The generating lines can be edited to deform the shape and new lines can also be inserted dynamically. However, all lines must be explicitely drawn by the user.

The closest approach to ours is the system by Zhou et al. [ZWS11] which uses ridges and valleys to define handles from which deformation and curvature can be edited. This method allows to edit CAD meshes easily, but is not adapted to arbitrary shapes. Moreover, many view-dependent salient structures cannot be captured by curvature lines.

Finally, [AMSF08] detects important features by constructing a hierarchical structure representing the 3D shape and uses it to select features at different scales. However, the proposed features are static and do not adapt to the viewpoint.

**Linear Rotation-Invariant Deformation.** Although our approach is compatible with any handle-based deformation technique [SA07, BS08], we choose the *linear rotation-invariant* (LRI) coordinates method [LSLCO05] as the underlying deformation machinery for our experiments. In particular, this method is linear and guarantees a unique surface reconstruction from positional constraints, with a good trade-off between speed and quality. LRI encodes geometry as differential coordinates between vertices and user deformation as positional constraints, resulting in an over-determined system, solved in the least-squares sense. In our framework, all line handles are eventually converted to such positional constraints for interactive update during line handle manipulation.

**Lines Drawing** Line drawing algorithms make use of only few binary lines to convey the shape of (potentially complex) 3D models faithfully [CSD*09] (see Fig. 1). We make use of those lines to generate line control handles for freeform deformation. In particular, our system allows to use any combination of *Occluding Contours* (OC), which are the isolines corresponding to null dot product between view direction and surface normal; *Suggestive Contours* [DFRS03] (SC), which correspond to OC visible from nearby point of view; *Apparent Ridges* [JDA07] (AR) which model surface curvature maxima once projected in the image plane; and *Ridges and Valleys* [IFP95] (RV) which correspond to curvature maxima and minima. Whith the exception of RV, all these lines are *view-dependent*, which means that the deformation control structure (i.e., line handle set) will adapt (smoothly) when moving the camera (see Fig. 1).

## 3. Line-based Deformation Control

**Overview.** Our modeling pipeline builds upon feature lines extraction to generate plausible handles automatically and can be summarized in a 5-stages loop (Fig. 2):

1. starting from a triangle surface mesh, a set of lines is extracted according to the current point of view;
2. lines are regularized to reduce their tangent variation;
3. a ROI is computed by diffusion on the surface, starting from the line selected by the user;
4. the ROI can be interactively adjusted to fit the target deformation scale;
5. the user manipulates the so-defined line handle and the surface is deformed accordingly using LRI.

**From Feature Lines to Line Handles.** At every frame, our system extracts feature lines stemming from line drawing algorithms (e.g., SC, AR, RV). Depending on the mesh structure and the view point, feature lines can be highly irregular, made of many disconnected components and exhibit a strong tangent variation (e.g., contour lines are not contained in front parallel planes). This often leads to a counter-intuitive behavior of the ROI generation and subsequent deformation influence. Therefore, we start by applying an *on-surface* regularization scheme to feature lines. First, we sort feature lines by feature type (e.g, SC, AR). Second, we progressively aggregate small components by inspecting the 1-neighborhood of each line extremity and connecting neigh-

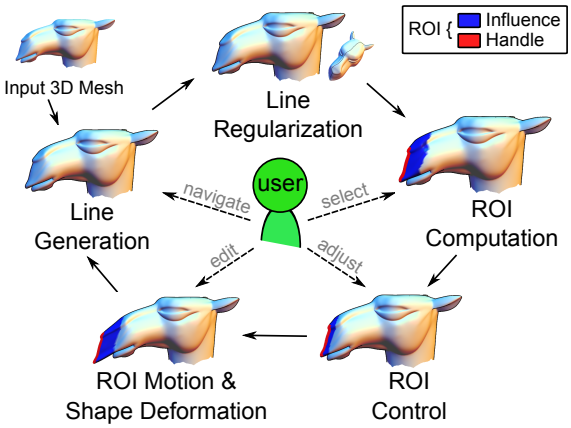



**Figure 2:** ***Overview****: the user controls the deformation interactively in a* select-and-grab *loop. The handle's thickness is enhanced for readability.*

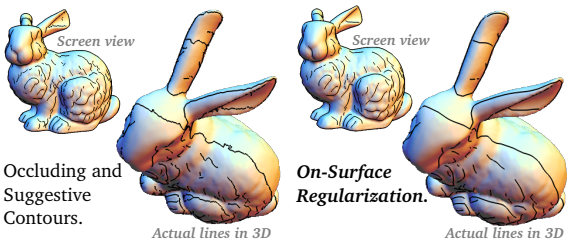



**Figure 3:** ***On-Surface Line Regularization*** *converts irregular and quaking feature lines to plausible handles. Three iterations were done in this example.*

boring lines on a per-type basis. Components with a small screen-space projection are removed after this step according to a user-defined threshold (typically around 10 pixels in our experiments). Third, we regularize the resulting lines. As we have to express the line regularization in the surface subspace – high frequencies removal should not make lines crossing or leaving the surface – we interleave a weigthed least squares filtering restricted to the line samples with a surface projection in an iterative procedure. More formally, let $\{\mathbf{p}_i\}$ be the discrete set of points forming a feature line $\mathbf{L}$ on the input surface $\mathbf{S}$. $\mathbf{L}$ is usually **not** a subset of the vertices of $\mathbf{S}$. We define a regularized line $\mathbf{L}' = \{\mathbf{p}'_i\}$ by computing

$$\mathbf{p}'_i = \Pi^k(\mathbf{p}_i) = (\Pi_\mathbf{S} \circ F_\mathbf{L})^k(\mathbf{p}_i)$$

with $\Pi_\mathbf{S}(\mathbf{x})$ the surface projection of $\mathbf{x}$ onto the closest location on $\mathbf{S}$ and $F_\mathbf{L}(\mathbf{x})$ a weighted combination of geodesic neighbors of $\mathbf{x}$ w.r.t. $\mathbf{L}$:

$$F_\mathbf{L}(\mathbf{x}) = \frac{\sum_{\mathbf{p}_i \in N_\mathbf{x}} \omega(||\mathbf{x}-\mathbf{p}_i||)\mathbf{p}_i}{\sum_i \omega(||\mathbf{x}-\mathbf{p}_i||)}$$

with $N_\mathbf{x}$ the geodesic neighbors of $\mathbf{x}$ in $\mathbf{L}$. In practice we approximate this set by collecting $N_\mathbf{x}$ through a floodfilling process performed on $\mathbf{L}$, starting at $\mathbf{x}$ and restricted to an euclidean ball of radius $\varepsilon$, the user-defined regularization scale (typically 5% of the bounding box diagonal). We use a compactly supported quartic polynomial weighting kernel [Wen95] behaving similarly to a gaussian kernel for a smaller computational footprint:

$$\omega(t) = \begin{cases} (1-\frac{t}{\varepsilon})^4(\frac{4t}{\varepsilon}+1) & \text{if } 0 \le t \le \varepsilon \\ 0 & \text{if } t > \varepsilon \end{cases},$$

This interleaved procedure is iterated $k$ times or stopped when stabilizing. The algorithm is fast enough so that the user can adjust $\varepsilon$ interactively. Figure 3 shows the result of this regularization procedure on a typical case.

**ROI Definition.** We recall that the ROI is composed of a handle $H$ (Fig. 2, red), which strictly undergoes the user-controlled rigid motion, and an influence area $I$ (Fig. 2, blue) with geometry solved using LRI. As the handle defines positional constraints in the LRI system, we gather its elements by collecting surface vertices located within a small neighborhood (within a ball of radius of 2% of the bounding box diagonal) around the selected (and regularized) line and apply the user's rigid transformation to them. Optionnaly, the user can apply non-rigid transformations using a linear blending decaying from full motion at the exact pointed location (e.g. mouse pointer) to null motion at the extremities of the line handle.

The influence area is also made of surface vertices and is gathered using a *diffusion* process starting from H. Let $\{\mathbf{v}_i\}$ be the vertex set of $\mathbf{S}$. As we want to grow uniformly $I$ around $H$, we define the (discrete) distance function $D_H$ to be fast enough for interactive control:

$$D_H(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \in H \\ \underset{\mathbf{v}_i \in \mathcal{N}_\mathbf{x}}{\text{argmin}} \left(D_H(\mathbf{v}_i)\right) + \|\mathbf{x}-\mathbf{v}_i\|_2) & \text{otherwise} \end{cases}$$

with $\mathcal{N}_\mathbf{x}$ the 1-neighborhood of vertex $\mathbf{x}$. Thus, we collect $I = \{v_i \mid D(\mathbf{v}_i) \le \delta_I\}$ using a Dijkstra flood-filling, starting from $H$. This diffusion process is bounded by $\delta_I$, the supervised size of $I$ (30% of the bounding box diagonal in our experiments). The user can control independently the size of $H$ and $I$ to adjust the scale of the deformed region: $H$ is controlled by bounding the distance to mouse pointer when gathering surface vertices composing it, while $I$ is ruled by setting $\delta_I$. We provide an intuitive interface allowing to control this 2-dimensional bound using 2D mouse motions on the screen, with x-component sizing $H$ and y-component sizing $I$. Last, at each user motion, the new positions of $H$ are updated in the LRI system and $I$ is solved.

## 4. Implementation and Results

We implemented our system in C++ using OpenGL for rendering, RTSC [rts] for line extraction and CholMod [cho] for solving LRI deformation systems. In Table 1, we report tim-

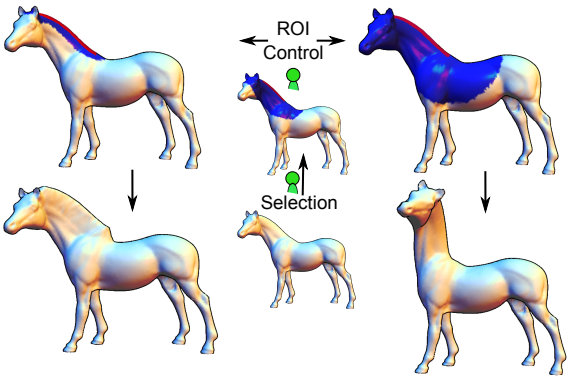



**Figure 4:** ***ROI control:*** *adaptating the default ROI size (middle) enables a large panel of various deformations from a single handle. The handle's thickness is enhanced for readability.*

| Model | Faces | Selection | Regul. | ROI |
|---|---|---|---|---|
| Camel | 22,000 | 20 ms | 147 ms | 26 ms |
| Horse | 40,000 | 58 ms | 445 ms | 70 ms |
| Donut | 51,000 | 55 ms | 114 ms | 91 ms |
| Bunny | 70,000 | 79 ms | 191 ms | 116 ms |

**Table 1:** *Timing for the different steps of our approach ( Intel Core2Duo at 2.2GHz with 4Gb of main memory).*

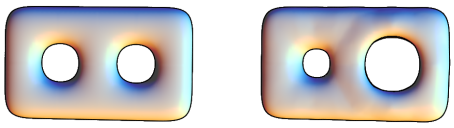
ings for the different steps of our line handle generation procedure, which clearly shows that our approach is fast enough for interactive editing on various models. In Fig. 5, we show simple examples of deformations which would have been tedious if the ROI had to be designed manually. Fig. 6 shows more complex examples, where different kinds of lines help modeling different structures on the surface. All of these deformations where made in one or less that one minute and from a single view point, except for the *Goro* that took four minutes and required a tens of view point.

## 5. Conclusion

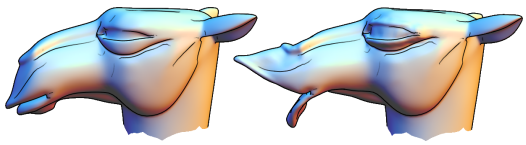
We have introduced a new automatic handle generation technique exploiting the faithful salient features captured by *Line Drawing* algorithms to define a plausible interface for surface-based shape deformation techniques. We proposed efficient strategies to regularize the so-defined line handles and define a proper ROI dynamically, both processes being controlled by the user. As a result, the system allows to quickly edit complex shapes (e.g., piecewise smooth, arbitrary topology, boundaries) without resorting to a tedious manual selection step. Still, as our approach focuses on modifying existing surface features rather than inserting new ones, manual handle design remains available when needed.

## References

[AMSF08] ATTENE M., MORTARA M., SPAGNUOLO M., FALCIDIENO B.: Hierarchical convex approximation of 3d shapes for fast region selection. In *Proc. SGP* (2008), pp. 1323–1332. 2



(a) Scale editing from OC lines.



(b) Shape posing from OC and AR lines

**Figure 5:** *Using expressive lines as deformation control structures makes trivial editing actions that would be difficult with manual point-based or brush-based ROI selection.*

**Figure 6:** *Our approach allows to deform complex models in few minutes. Left: using SC lines to deform the* Goro. *Right: using RV lines to deform* Jane.

[BPGK06] BOTSCH M., PAULY M., GROSS M., KOBBELT L.: Primo: coupled prisms for intuitive surface modeling. In *Proc. SGP* (2006), pp. 11–20. 1

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE TVCG 14*, 1 (2008), 213–230. 1, 2

[cho] Cholmod. www.cise.ufl.edu/research/sparse/cholmod. 3

[CSD*09] COLE F., SANIK K., DECARLO D., FINKELSTEIN A., FUNKHOUSER T., RUSINKIEWICZ S., SINGH M.: How well do line drawings depict shape? In *ACM SIGGRAPH 2009 papers* (2009), pp. 28:1–28:9. 2

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *SIGGRAPH* (2003). 2

[IFP95] INTERRANTE V., FUCHS H., PIZER S.: Enhancing transparent skin surfaces with ridge and valley lines. In *Viz.* (1995). 2

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *Proc. SIGGRAPH* (1999), pp. 409–416. 2

[JDA07] JUDD T., DURAND F., ADELSON E. H.: Apparent ridges for line drawing. *ACM Trans. Graph.* (2007). 2

[LSLCO05] LIPMAN Y., SORKINE O., LEVIN D., COHEN-OR D.: Linear rotation-invariant coordinates for meshes. In *Proceedings of ACM SIGGRAPH 2005* (2005). 1, 2

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: Designing freeform surfaces with 3d curves. *ACM SIGGRAPH 2007* (2007). 2

[NSACO05] NEALEN A., SORKINE O., ALEXA M., COHEN-OR D.: A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph. 24*, 3 (2005), 1142–1147. 1

[rts] Rtsc. http://gfx.cs.princeton.edu/proj/sugcon/. 3

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proc. SGP* (2007), pp. 109–116. 1, 2

[SE98] SINGH K., EUGENE F.: Wires: a geometric deformation technique. In *Proc. SIGGRAPH* (1998). 1

[Wen95] WENDLAND H.: Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math. 4*, 4 (1995), 389–396. 3

[ZNA07] ZIMMERMANN J., NEALEN A., ALEXA M.: Silsketch: automated sketch-based editing of surface meshes. In *Proc. SBIM* (2007). 1

[ZWS11] ZHOU Q., WEINKAUF T., SORKINE O.: Feature-based mesh editing. In *Proc. Eurographics, Short Papers* (2011). 2