Driller: An intuitive interface for designing tangled and nested shapes

Tara Butler[1], Pascal Guehl[1], Amal Dev Parakkat[2], Marie-Paule Cani[1]

[1] LIX - Ecole Polytechnique/CNRS, [2] LTCI - Telecom Paris
Institut Polytechnique de Paris, France

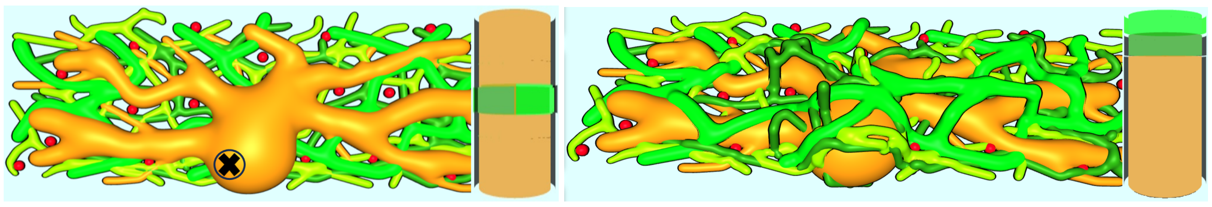

Figure 1: *Design steps for a tree model using sketch-based modeling extended with our Driller interface. All 3D shapes are initially created in the same drawing plane (left). Users can select a viewpoint (marked in black) and trigger the Driller interface (middle) to locally arrange elements, such as the depth ordering of branches and lianas (right - along with the modified arrangement in the Driller interface).*

Abstract

The ability to represent not only isolated shapes but also shapes that interact is essential in various fields, from design to biology or anatomy. In this paper, we propose an intuitive interface to control and edit complex shape arrangements. Using a set of pre-defined shapes that may intersect, our "Driller" interface allows users to trigger their local deformation so that they rest on each other, become tangled, or even nest within each other. Driller provides an intuitive way to specify the relative depth of different shapes beneath user-selected points of interest by setting their local depth ordering perpendicularly to the camera's viewpoint. Deformations are then automatically generated by locally propagating these ordering constraints. In addition to being part of the final arrangement, some of the shapes can be used as deformers, which can be later deleted to help sculpt the target shapes. We implemented this solution within a sketch-based modeling system designed for novice users.

CCS Concepts

• ***Human-centered computing*** $\rightarrow$ *Graphical user interfaces;* • ***Computing methodologies*** $\rightarrow$ ***Shape modeling;***

1. Introduction

While intuitive modeling systems have made significant progress in enabling novice users to create 3D shapes, modeling arrangements of closely related shapes remains challenging as geometric modeling systems typically concentrate on creating individual shapes. Despite organic shapes often exhibiting interferences through contact, as seen in natural phenomena such as plant growth, cellular arrangements, and vascular systems, modeling such intricate shapes proves particularly challenging. Motivated by applications such as the creation of 3D illustrations in biology and medicine, we introduce a solution to ease the creation, exploration, and manipulation of 3D shape arrangements. The user creates and positions a set of individual shapes, which may initially intersect. We then provide an interface called the "Driller," inspired by geology, which enables users to intuitively define the way shapes locally interact. This interface is associated with a deformation mechanism that allows one shape to be deformed relative to another, seen as a support canvas, in order to meet the specified constraints. Our contributions are implemented within a sketch-based modeling system that relies on skeleton-based implicit surfaces [BPCB08, ZBQC13], enabling us to make use of isosurfaces of scalar fields to solve the problem. In addition to easing the creation of shape arrangements, the resulting system enables users to sculpt the targeted shapes, thanks to extra support canvases serving as shape deformers. In our investigation of organic shapes, we assume that the objects in question possess a natural smoothness, which is expected to be maintained throughout any editing process. While we can handle simple cases of tangled and nested shape arrangements intuitively, more complex and elaborated objects need to be further investigated in the future.

DOI: 10.2312/egs.20241025

2. Related work

Layering techniques in 2D and 3D shape design: As-rigid-as-possible [IMH05] presents an approach that continuously updates depth values for deformable 2D objects to ensure proper stacking orders. In contrast, Local layering [MP09] offers a solution for handling complex 2D layering scenarios involving multiple objects and allowing users to change layer orders locally. These techniques have been extended to 3D modeling, as discussed in Apparent layer operations [IM10], where similar principles apply to the manipulation of 3D objects. Additionally, Soft stacking [MP12] introduces a continuous layer ordering approach, resembling volumes of fog, enabling layers to appear both in front of and behind others within the same pixel. Unlike prior methods limited to specific domains like 3D cloth and rope arrangements [IM10], we extend the layering operations to more generic organic 3D shapes.

Deformers, canvases and nested shapes: Inspired by anatomical drawing, Sketch-based modeling of vascular systems [PCP10] focuses on creating 3D models of branching vessels from a single sketch with future plans to explore sketching onto support surfaces. In OverCoat [SSGS11], an implicitly defined 3D scalar field representing the canvas concept is used, offering a unified painting and sculpting metaphor for canvas fine-tuning. In contrast, our method approaches this challenge differently by using the isosurface of another shape to deform and infer depth levels. In Skippy [KYC*17], 2D strokes align with a 3D object's surface to create a 3D model, but unlike our method, they require viewpoint adjustments to create complex 3D arrangements. Other papers focusing on creating nested structures are SecondSkin [DPS15], which automates the creation of layered 3D structures from sketches by interpreting stroke geometry, and Generalized Matryoshka [Jac17] that introduces a method to find the largest scale replica of an object that nests inside itself. However, these methods work directly in a 3D environment, making it difficult to create complex, tangled, and nested structures.

3. Inspiring from geology to specify shape arrangements

Our interactive sketch-based modeling system, based on Matisse [BPCB08] and SCALIS [ZBQC13], iteratively takes 2D painted regions as input sketches and generates 3D implicit surfaces. Each surface represents the 0.5 isosurface of a volumetric scalar field created from a flat skeleton—a graph of polylines approximating the medial axis of the painted region. Initially, the input shapes have flat silhouettes and require deformation to become intertwined.

Drawing inspiration from the way geologists study soil and rock layers by digging perpendicularly to the surface to generate core samples (as shown in the inset), we introduce the "Driller" interface to enable users to specify shape interaction constraints through intuitive 1D drag-and-drop operations. Our interface allows users to click on any point of the screen to display the equivalent of a "core sample," computed along a ray from the camera viewpoint, and edit the arrangement of shape layers at that point, thus avoiding

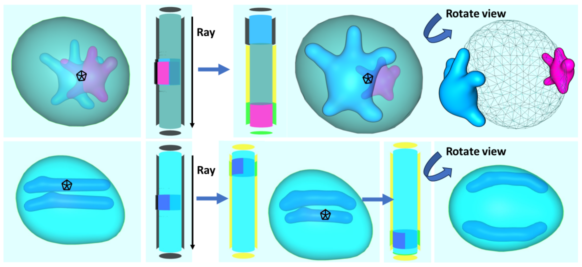

Figure 2: *Examples demonstrating Layering (Top) and Nesting (Bottom) along with corresponding Driller interfaces. User selects camera viewpoints (star mark) to locally arrange elements.*

the need for complex 3D navigation/manipulation. The screen widget representing the core sample is computed by casting a ray from the camera's viewpoint to display shape layers for each intersected shape (in the order of intersection). The local thickness and free space between layers are computed by identifying points at the 0.5 isovalue for each field function and colored accordingly to match the original drawing. Users can then reorder layers via simple 1D drag-and-drop along the selected ray, with different layer configurations managed internally by computing distances from shape centers.

At each arrangement-design stage, the user selects one shape as the "support canvas" and navigates to a viewpoint for arranging the depth of other shapes relative to this canvas. A click triggers the computation of a local driller interface, allowing manipulation of shapes relative to the canvas. A shape dragged onto the driller becomes the "active shape" and is automatically moved and/or deformed in 3D (see Figure 2). The support canvas can either be part of the final 3D model or only used for the actual deformation process and can be removed later (see Figure 5). Note that we need to pre-select a "support canvas" before deformation, since the constraints specified using the driller interface are constraints on isovalues, which should refer to some global volumetric field - namely the canvas's field.

4. Depth inference for the active shape

Preprocessing the active shape: Before processing the deformation, the active shape needs to be adequately sampled to have enough levels of detail to allow a perceptually natural, local deformation. In our framework, we consider that the thinner the shape is, the more flexible it should be, as if all the shapes were made of the same soft plastic material. Therefore, we associate our shapes with a manipulation skeleton of a sampling density proportional to the local radius of the shape computed during a traversal of the skeleton graph. The geometric primitives generating the implicit surface are resampled, if needed, by subdividing skeletal segments that are longer than the local diameter of the shape (see Figure 3).

Identifying the target isosurface of the canvas: Each time the user drops the slice representing the active shape to a new position on the Driller, it triggers the local 3D projection of the skeleton of the active shape to a well-chosen isosurface of the support canvas. We use the Driller interface to select the relevant isosurface: At

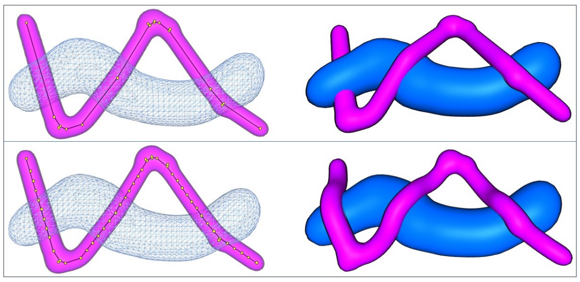

Figure 3: *Deformation without (Top) and with (Bottom) skeleton refinement (sampling density proportional to shape's local radius).*

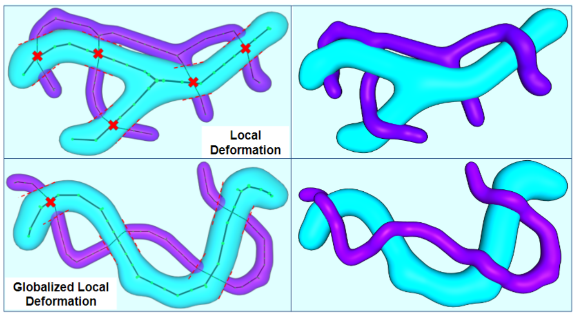

Figure 4: *Effect of local deformation (Top) and globalized local deformation (Bottom).*

each drop, a 3D point P_S corresponding to the desired depth for the active shape's skeleton is computed along the cast ray at the distance extracted from the Driller interface (scaled according to the size of the Driller interface's screen widget). The isosurface of the canvas to be used for projection is then given by $c = f_{canvas}(P_S)$.

Deforming the active shape: The skeleton of the active shape is locally projected to the isosurface $f_{canvas}(P) = c$ to create the desired local deformation (or a full displacement in the case of a small active shape), as follows: Starting at the skeletal vertex of the active shape that is the closest to the cast ray, we project the vertex to the isosurface using Newton's method in the direction of f_{canvas}'s gradient and then propagate the deformation to the neighboring vertices of the active shape's skeleton, as long as they lie inside the boundary of the support canvas. Penetrations are avoided thanks to the projection of the deformed shape to a specific isosurface of the support canvas.

Globalized depth ordering: While the locality of the Driller provides fine control over depth layering, enabling the design of tangled shapes, the user's manipulation task would become redundant if the active shape is to be positioned everywhere in the same way with respect to the support canvas upon the selected viewpoint. To reduce the user's burden in such cases, we also provide an option of 'globalized local deformation'. In this option, the user can specify the relative position of the active shape only once (using the Driller at one of the points where the shapes overlap) and use the selected isosurface $f_{canvas}(P) = c$ to deform the active shape more globally, wherever it comes in contact with the support canvas. To achieve this, we replace the above-described propagation algorithm by the projection to the targeted isosurface of all the skeletal vertices of the active shape that lie over the support canvas from the selected viewpoint. Figure 4 compares the effects of local deformation (red crosses show the points the user selected for local deformation) and globalized local deformation on a sample sketch.

Laplacian smoothing: As we are only displacing the skeletal vertices that project over the support canvas from the current viewpoint, it may result in undesired depth discontinuities in the final deformed shape. To avoid this, a constrained Laplacian smoothing step is applied to the new skeleton with displaced vertices, moving only those vertices that are not constrained to the targeted isosurface (excluding skeleton extremities). Based on this smoothed skeleton, the scalar fields are recomputed to extract the final smooth shape. In practice, we repeat n iterations of smoothing on skeletons, including branching and cycling structures, using the extended neighbors set $\mathcal{N}(i)$ of points X_i (with cardinality $|\mathcal{N}(i)|$), i.e., *adjacent* points, with a smoothing factor $\lambda \in [0,1]$:

$$X_i^{n+1} = X_i^n + \lambda\, L(X_i^n) \quad \text{with} \quad L(X_i^n) = \left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} X_j^n\right) - X_i^n$$

The overall deformation procedure is presented in Algorithm 1. The function *Find_Connected_Subset(P,S,C)* computes a subset N of the Refined Skeleton graph S, where a vertex $v \in N$ ($N \subseteq S$) if and only if: v lies inside the canvas shape C, and there exists a path $P_{v\to P}$ from v to P such that for all $p \in P_{v\to P}$, $p \in N$.

Algorithm 1 Local deformation

```
1:  procedure DEFORM(Active Shape $A$, Support Canvas $C$)
2:      Compute the Refined Skeleton Graph $S$ of $A$
3:      if the user wants to perform a local deformation then
4:          $UP$ = User-selected 2D point
5:          $P$ = Closest point of $UP$ in $S$ w.r.t. the viewing direction
6:          $N$ = Find_Connected_Subset(P, S, C)
7:          for each vertex $v \in N$ do
8:              Displace $v$ to the user-intended isosurface of $C$
9:      elseif user wants to perform globalized local deformations
10:         for each vertex $v \in S$ do
11:             if $v$ lies inside $C$ then
12:                 Displace $v$ to the user-intended isosurface of $C$
13:     Apply Laplacian smoothing to propagate the deformation
14:     return Recomputed deformed $A$
```

5. Results

Figures 1 and 6 illustrate shape arrangements modeled with our interface, and the companion video shows our system in action. Figure 5 illustrates the use of a canvas as a support for modeling by deforming shapes. For all models, we fix smoothing parameters $n = 10$ and $\lambda = 0.2$.

Implementation: We implemented our prototype system in WebGL/three.js. The complex shape in the teaser image took ten minutes to model, and all the remaining models were created in less than five minutes (with the simplest models with only two shapes taking less than one minute). While the interaction with our driller

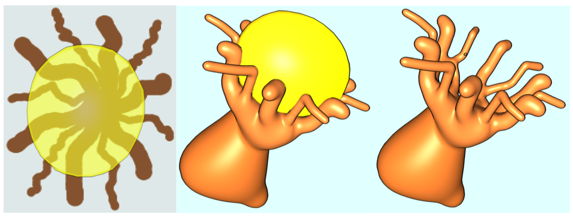

Figure 5: *Using a support canvas to ease the modeling of a complex cylinder anemone shape (the body is sketched separately).*

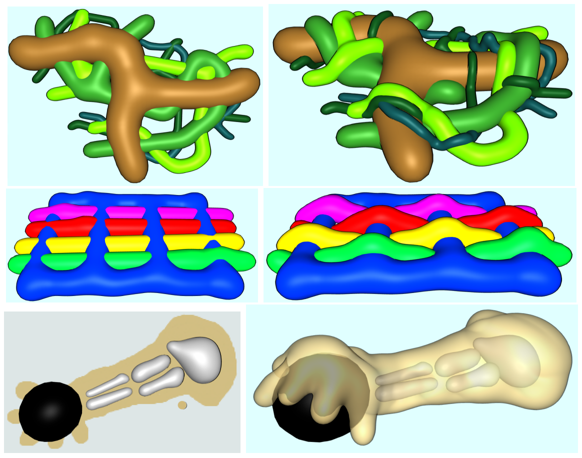

Figure 6: *Various complex shapes created using our interface. Left to right: before and after using our interface.*

interface takes only a few milliseconds, repolygonizing the shapes due to our CPU implementation took around one second.

Discussion and limitations: While we succeeded in enabling the easy creation of simple tangled and nested elongated structures, more complicated objects with either more details (e.g. a basket of laundry) or a self-intersecting surface, such as a complex, elongated shape bending over itself and creating a knot (e.g. a bowl of spaghetti) need to be further investigated in the future. Indeed, the input shapes are created by painting 2D regions, so shape parts blend rather than self-intersecting. One workaround would be to incrementally paint these shapes, adjusting the depth appropriately and locally blending them at the end. Secondly, consistency in user input is crucial for maintaining the desired deformations due to the propagation of changes. Therefore, as in any depth ordering system, inconsistent user-specified depth ordering would lead to conflicting constraints. Our driller interface currently displays in the same way sandwiched shape layers and a shape nested within a larger one. Conducting a user study would be necessary to understand how to solve this visualization problem in the most natural way.
At the moment, our implementation supposes a uniform material to deform our shapes. In the future, we plan to use a physically based approach like a mass-spring system to manage different stiffness, constrain bending, and simulate a more complex relaxation scheme. As mentioned above, our deformation method takes a smoothness hypothesis to extend the method to more rigid bodies such as articulated characters. A different algorithm (e.g., inverse kinematics) should be used for deforming the articulated skeletons, while the use of the driller interface would remain unchanged.
Lastly, thick structures may not have enough degrees of freedom to be able to interleave between thin ones - they will match the requested isosurface locally, and then the deformation will be propagated farther, which might destroy the effect of a previous positioning constraint, and cause interpenetrations.
Handling more detailed surfaces could be an interesting extension of our approach. In this case, local surface details could be associated with small skeletal segments, allowing us to apply deformation in a hierarchical order. In practice, the large skeletal structure could first be projected onto the support canvas before rigidly mapping the local details in the local frame of their parent skeletal structure.

6. Conclusion

We presented "Driller," an intuitive interface for crafting intricate 3D shape arrangements, from tangled to nested shapes. Using 1D drag-and-drop gestures, our interface enables simple interaction, triggering local or global shape deformations. In the future, we aim to explore enhancing control over deformation locality, currently determined automatically by the system based on shape thickness.

Acknowledgements

This work was funded by Prof. Cani's fellowship from Hi!Paris.

References

[BPCB08] Bernhardt A., Pihuit A., Cani M.-P., Barthe L.: Matisse: Painting 2d regions for modeling free-form shapes. In *SBM'08-Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2008), Eurographics Association, pp. 57–64. 1, 2

[DPS15] De Paoli C., Singh K.: Secondskin: sketch-based construction of layered 3d models. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 1–10. 2

[IM10] Igarashi T., Mitani J.: Apparent layer operations for the manipulation of deformable objects. In *ACM SIGGRAPH 2010 papers*. 2010, pp. 1–7. 2

[IMH05] Igarashi T., Moscovich T., Hughes J. F.: As-rigid-as-possible shape manipulation. *ACM transactions on Graphics (TOG) 24*, 3 (2005), 1134–1141. 2

[Jac17] Jacobson A.: Generalized matryoshka: Computational design of nesting objects. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 27–35. 2

[KYC*17] Krs V., Yumer E., Carr N., Benes B., Měch R.: Skippy: Single view 3d curve interactive modeling. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–12. 2

[MP09] McCann J., Pollard N.: Local layering. *ACM Transactions on Graphics (TOG) 28*, 3 (2009), 1–7. 2

[MP12] McCann J., Pollard N. S.: Soft stacking. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 469–478. 2

[PCP10] Pihuit A., Cani M.-P., Palombi O.: Sketch-based modeling of vascular systems: a first step towards interactive teaching of anatomy. In *Sketch-Based Interfaces and Modeling* (2010), pp. 151–158. 2

[SSGS11] Schmid J., Senn M. S., Gross M., Sumner R. W.: Overcoat: an implicit canvas for 3d painting. In *ACM SIGGRAPH 2011 papers*. 2011, pp. 1–10. 2

[ZBQC13] Zanni C., Bernhardt A., Quiblier M., Cani M.-P.: Scale-invariant integral surfaces. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 219–232. 1, 2