Sketch-based Modeling

Frederic Cordier

Yotam Gingold

Even Entem

Marie-Paule Cani

Karan Singh

Sketch stroke acquisition & processing

Karan Singh

Issues in digital sketching

• Stroke filtering	*fairing, curve-fitting.*
• Stroke processing	*segmentation, recognition, regularization.*
• Stroke dynamics	*pressure, tilt, speed, temporal order.*
• Stroke appearance	*NPR, stylization, perception.*
• Stroke-based UI Control	*widgets, crossing, gestures.*

Stroke filtering: noise & error sources

- User error
  - Intent (wants a square but draws a rectangle).
  - Execution (unsteady hand).
  - Ergonomic (awkward drawing posture).
- Device error
  - Input (tablets better than mice or trackpads).
  - Resolution (projected better than surface capacitance).
  - Signal Noise.

What are desirable strokes?

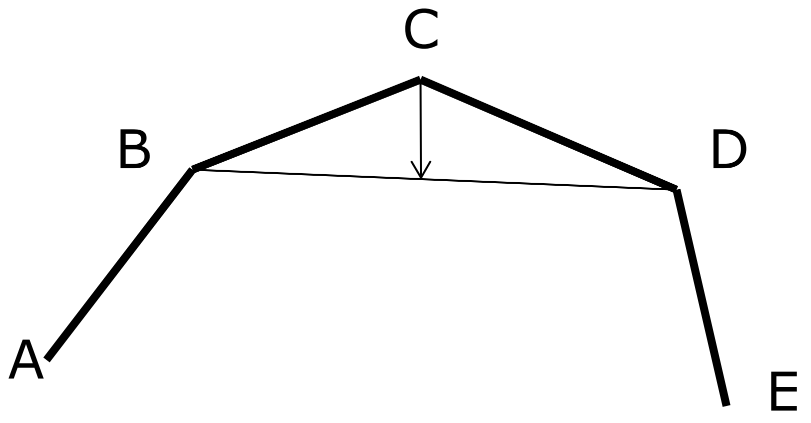
Smoothness: “*tangent and perhaps curvature continuous curves*”

[Farin et al. 87].

Simple smoothing approaches

- Laplacian. (neighbour averaging).
- Bi-Laplacian.
- LSQ spline fitting.

Simple smoothing: Laplacian

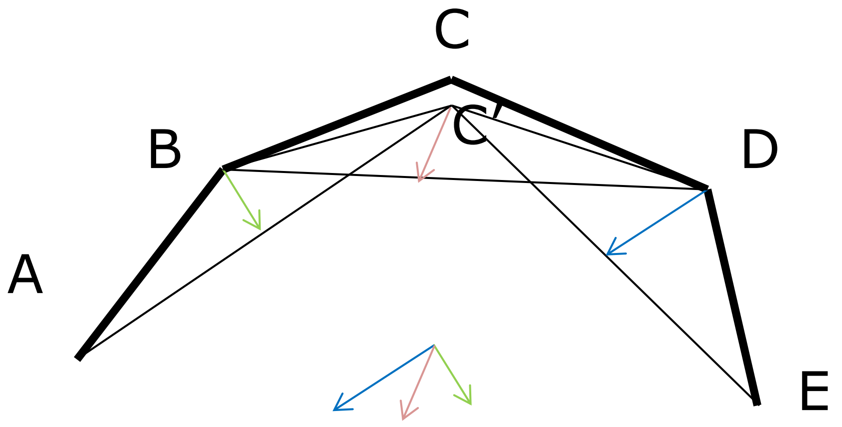


lap(C) = (B+D)/2-C

C'= C+ d**lap*(C)   0<d<1

Best to run many iterations with
A small d, for eg. 5 iterations d=0.2.

Simple smoothing: Bi-Laplacian



Find a C' such that:

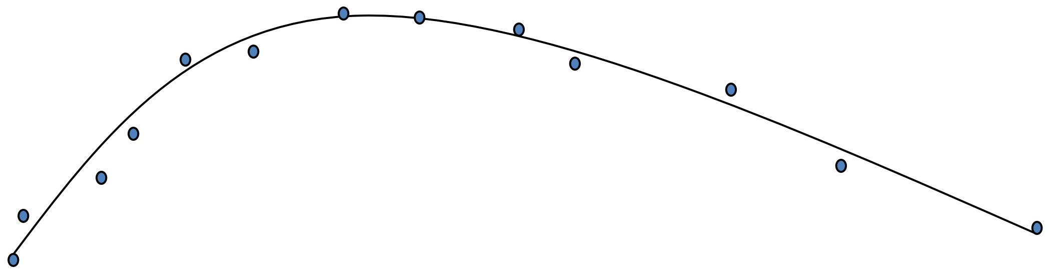
lap(C')   = (*lap*(B)   +   *lap*(D)   )/2

(B+D)/2-C'= (((A+C')/2-B)+((E+C')/2-D))/2

C'= 2/3 (B+D-A/4-E/4)

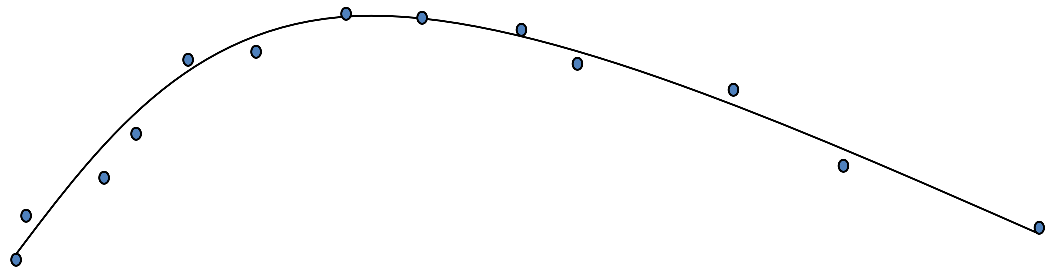
bi-lap(C)=C'-C

Simple smoothing: LSQ fitting



f(t)=(x,y) from points (x_i, y_i)

Simple smoothing: LSQ fitting



LSQ solves for f to minimize error $\Sigma_i\ |f(t_i)-(x_i, y_i)|^2$

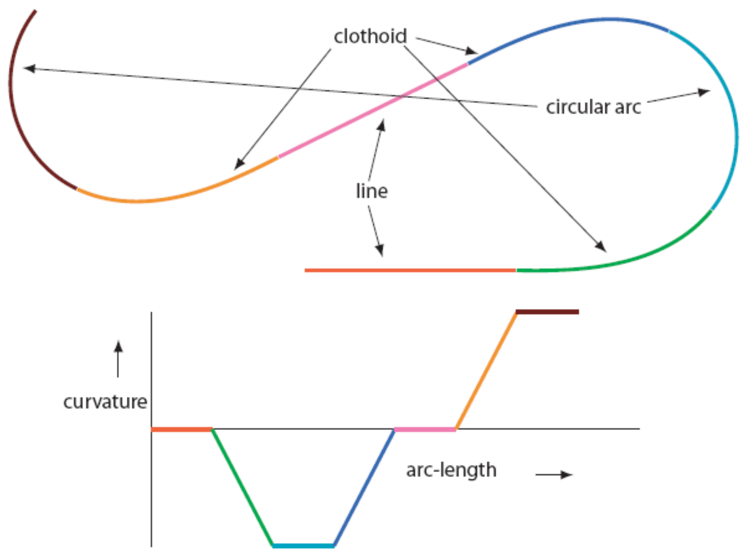
```
Approach:       guess $t_i$ ;
                LSQ solve for f ;
                refine $t_i$ for current f ;

                iterate…
```

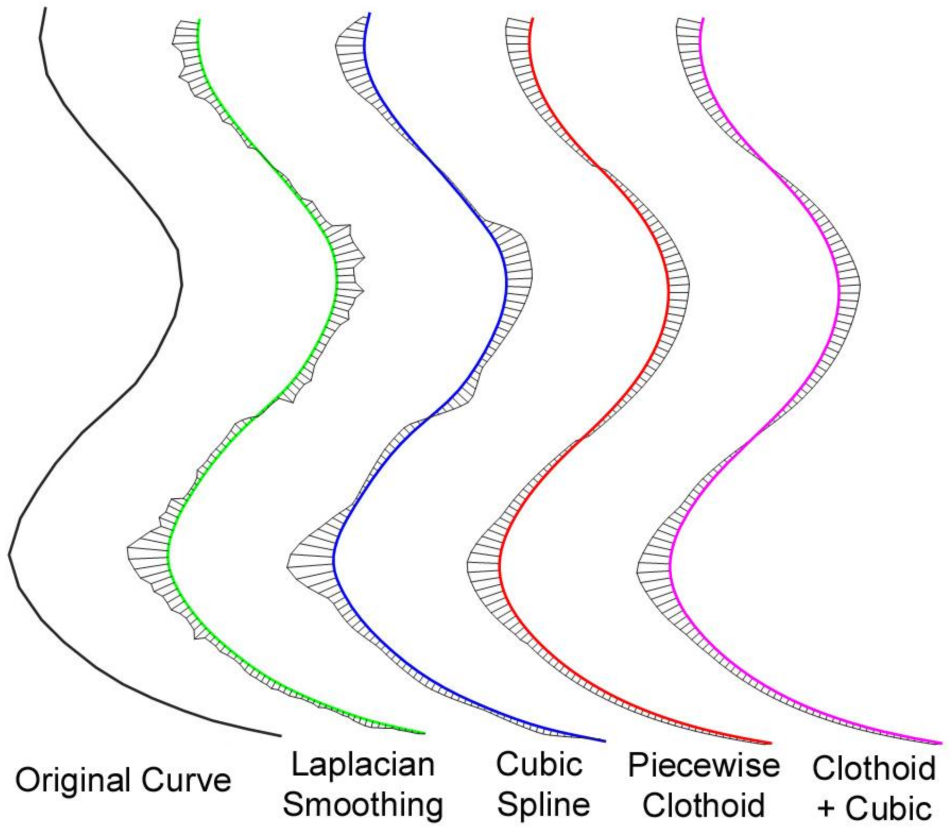
What are desirable strokes?

- **Fairness**: *“curvature continuous curves with a small number of segments of almost piecewise linear curvature”* [Farin et al. 87].

- Lines, circles and clothoids are the simplest primitives in curvature space.



Comparative approaches to fairing



[**McCrae & Singh**, Sketching Piecewise Clothoid Curves, *SBIM 2008*]
source code: http://www.dgp.toronto.edu/~mccrae/clothoid/

Desirable strokes

- **Neatness**: *“a combination of fairness and fine detail as intended by the user”.*

- Requires either implicit knowledge of user-intent, or an explicit neatening directive by the user.

Stroke neatening: French curves

Physical tools, used to model curves.

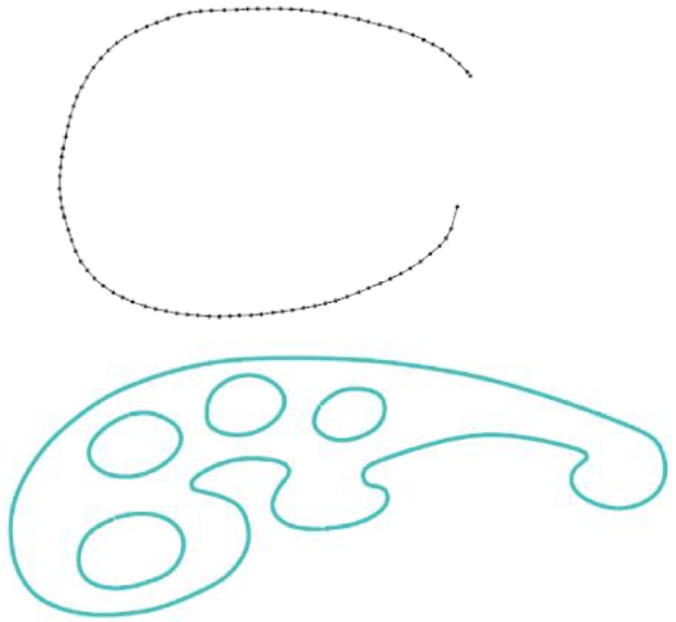
French curves + sketch interface

smooth shape priors,
specify a style/standard

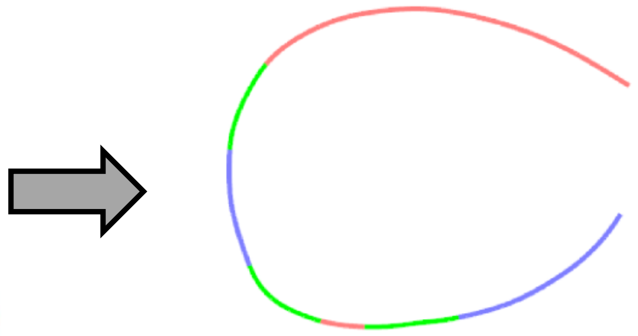
fluid free-form

Stroke neatening: French curves

input polyline



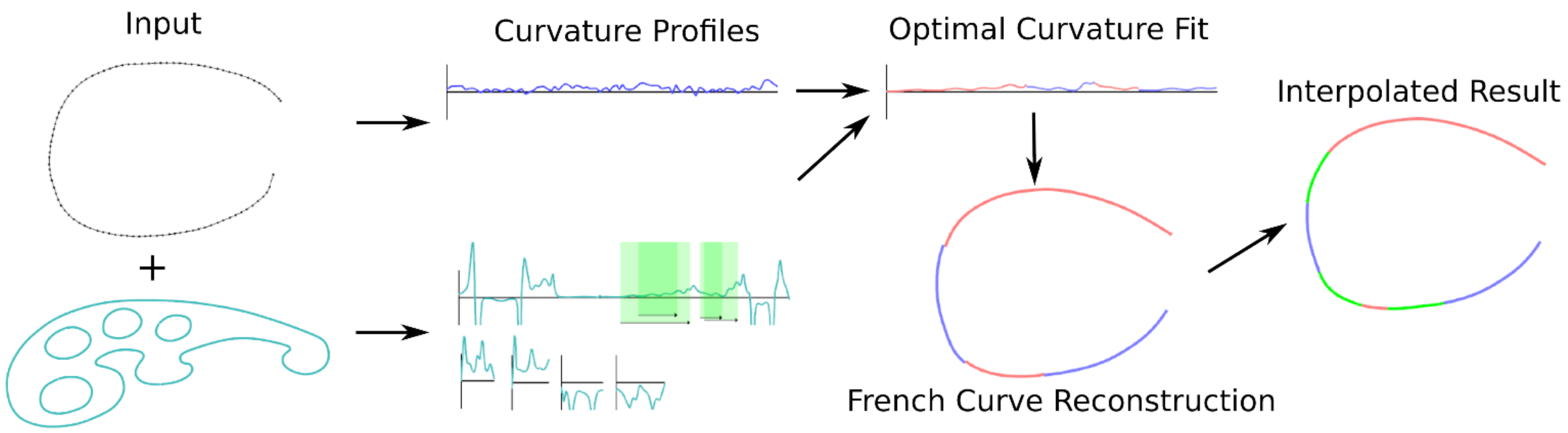
French curve



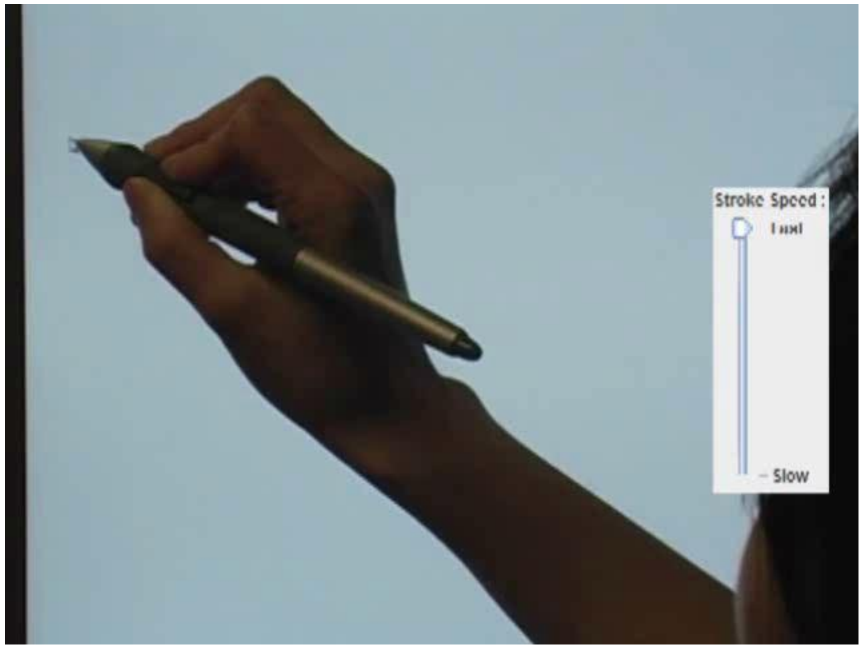
optimally fit pieces of the French curve to the input

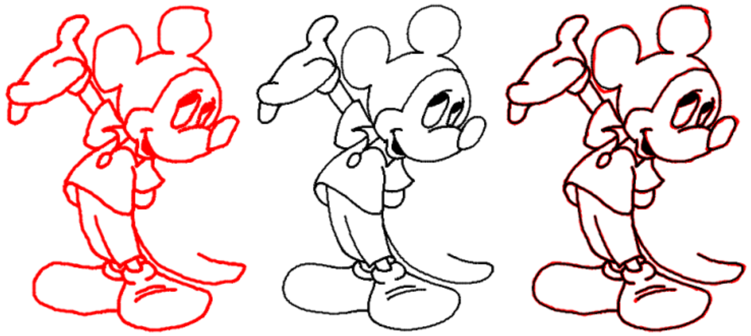
[**McCrae & Singh**, *Neatening sketched strokes using piecewise French Curves*, *SBIM 2011*]

Stroke neatening: French curves



Stroke neatening & dynamics: elasticurves

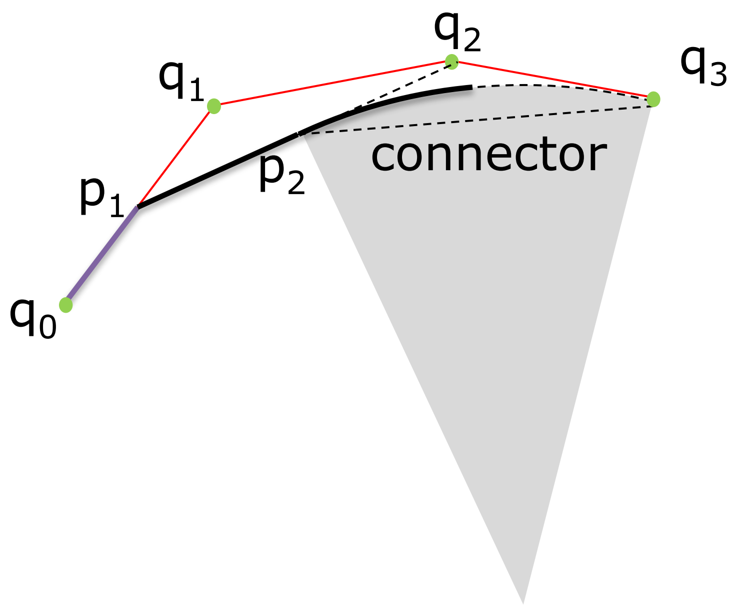




[**Thiel, Singh**, **Balakrishnan** Elasticurves: Exploiting Stroke Dynamics and Inertia for the Real-time Neatening of Sketched 2D Curves, *UIST 2011*]
java applet: http://www.dgp.toronto.edu/~ythiel/Elasticurves/

Elasticurve

Input q_i's sampled at a time interval of *dt*

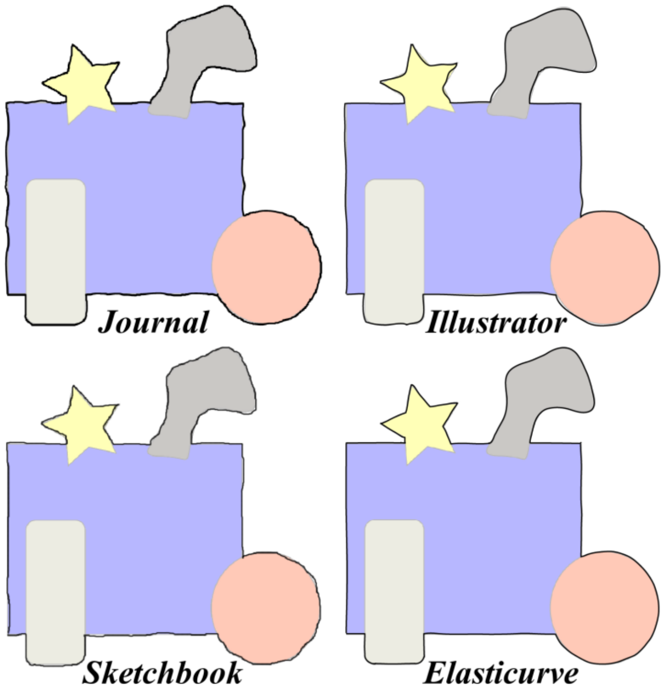


responsiveness = connector arc-length fraction extending an elasticurve.

Elasticurve Properties

- **Explicit and real-time**: neatness is directly correlated to drawing speed and *responsiveness*.
- **Analytic:** resilience to *dt* sampling variation.
- **Precise:** embodies desirable shapes as connectors.

Elasticurve evaluation & curve quality

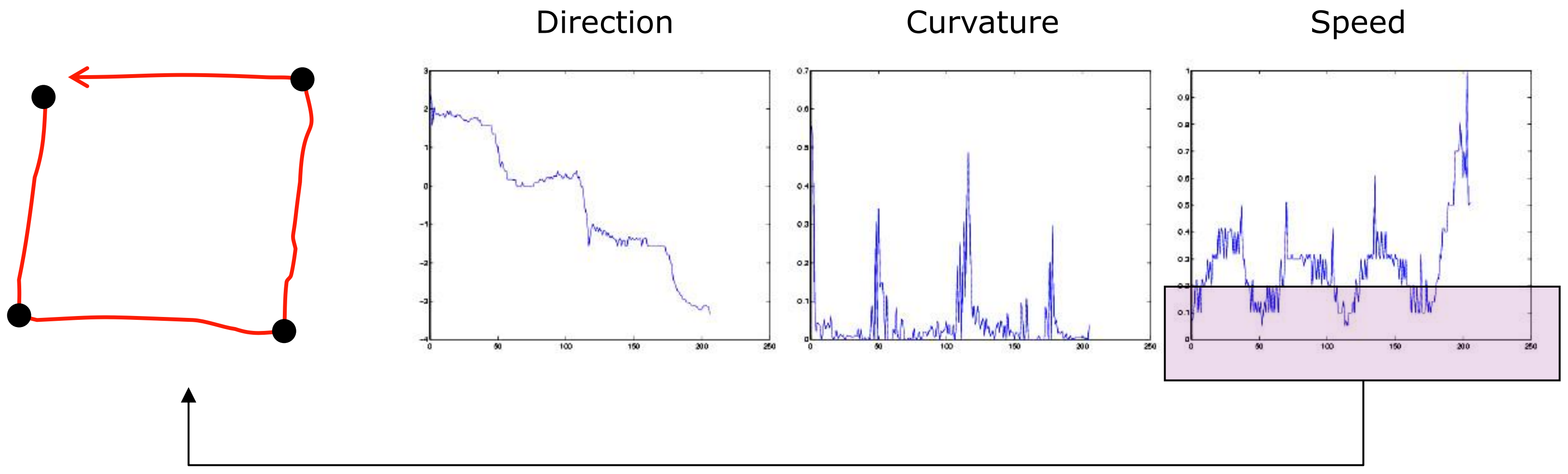


Intermediate user, trackpad, visual best of 7 attempts.

Stroke Processing

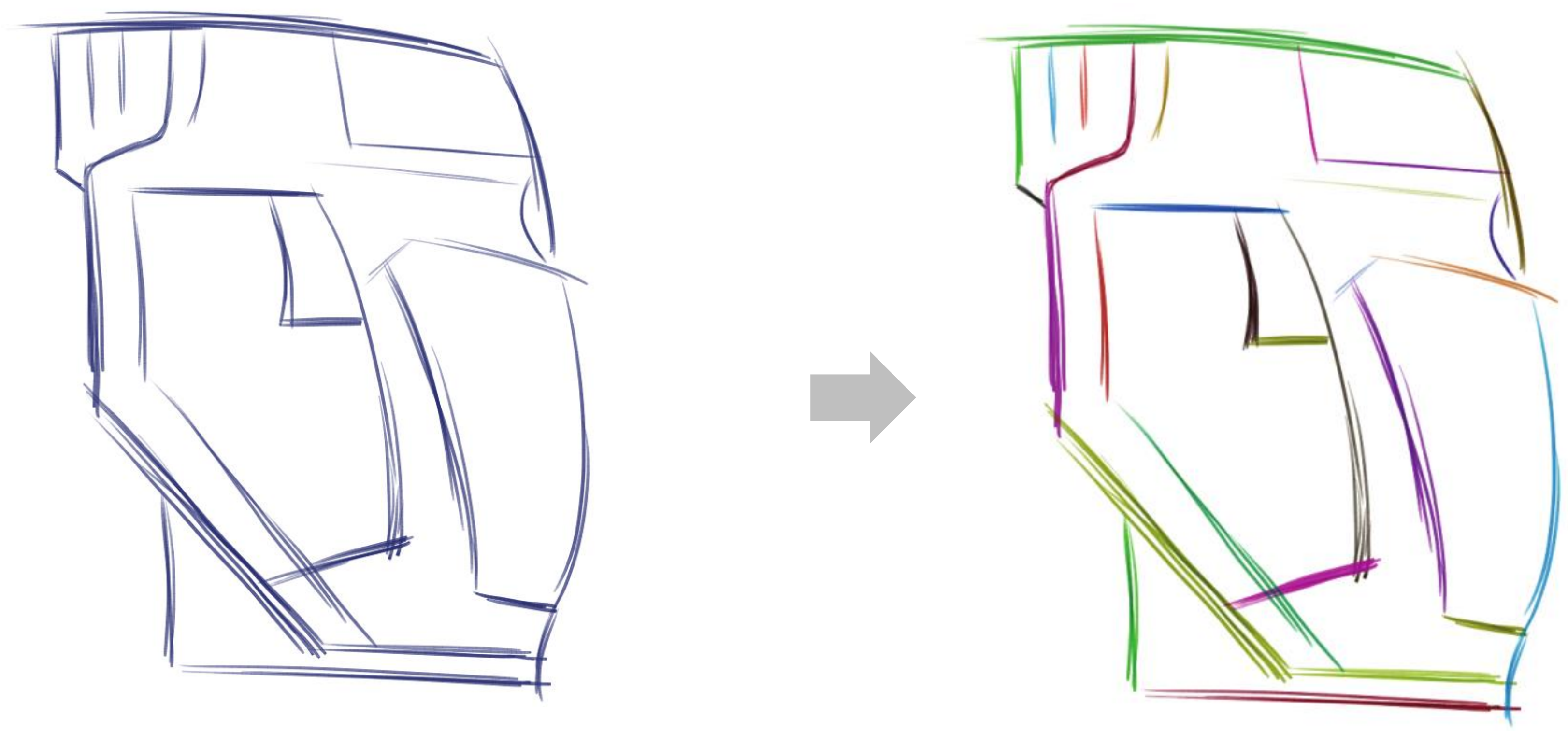
- *Filtering, neatening, beautification* can also be considered as stroke processing.
- Segmentation, classification, recognition.
- Regularization.
- Abstraction.

Stroke segmentation: finding corners



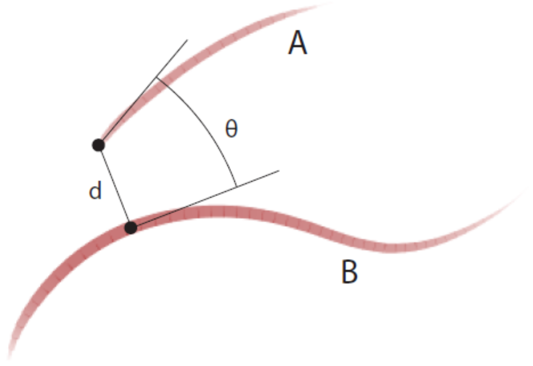
[**T. Sezgin et al.**, *Sketch Based Interfaces: Early Processing for Sketch Understanding*, Workshop on Perceptive User Interfaces, 2001.]

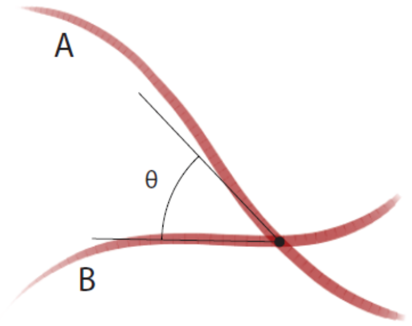
Stroke classification: pentamenti

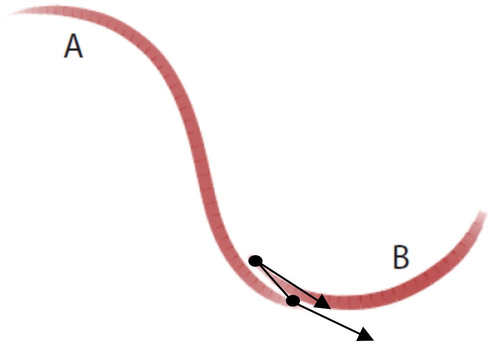


[**G. Orbay & L. Kara.**, *Beautification of Design Sketches Using Trainable Stroke Clustering and Curve Fitting.* IEEE Transactions on Visualization and Computer Graphics 17, 5 (May 2011).]

Geometric Stroke Features







Proximity

$$d_{AB} = ||\mathbf{x}_i - \mathbf{x}_j||$$

Alignment

$$a_{AB} = \frac{|\angle(\mathbf{n}_A, \mathbf{n}_B)|}{\pi/2}$$

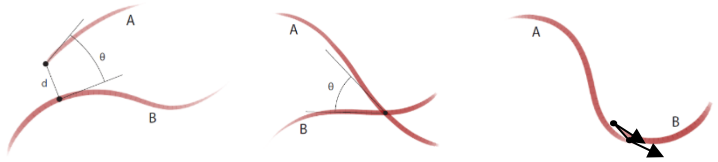
Continuity

$$c_{AB} = \frac{\|(\mathbf{n}_A \times \mathbf{n}_B)\| + \|(\mathbf{n}_A \times \mathbf{n}_s)\| + \|(\mathbf{n}_B \times \mathbf{n}_s)\|}{3}|s|$$

- Pairwise features
- Stroke proximity
- Local learning

EUROGRAPHICS 2016

Group Strokes by Affinity



Affinity = Proximity + Alignment + Continuity

learning approaches with or without examples:

neural network

spectral clustering

greedy grouping (single-link clustering)

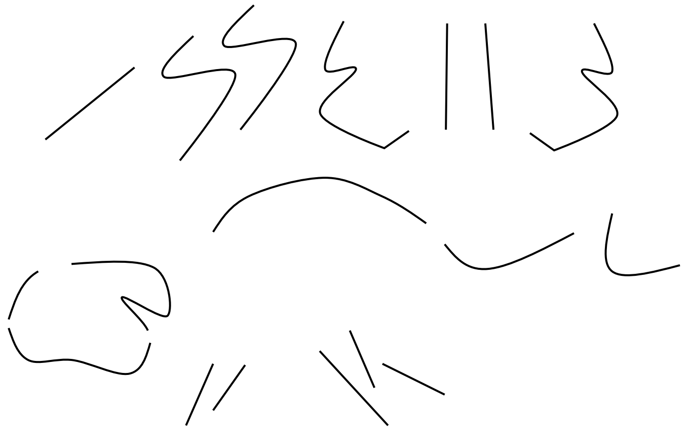
Stroke grouping and regularization

Gestalt Principle

“The whole is greater than the sum of its parts”

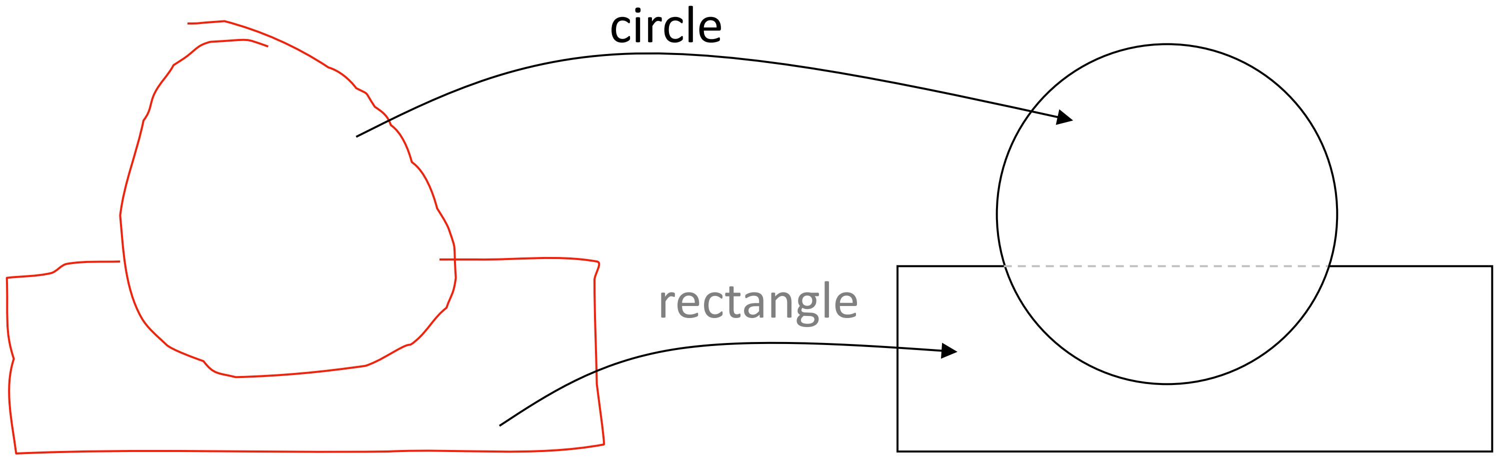
Gestalt grouping and regularization

- Similarity
- Symmetry
- Continuation
- Closure
- Proximity



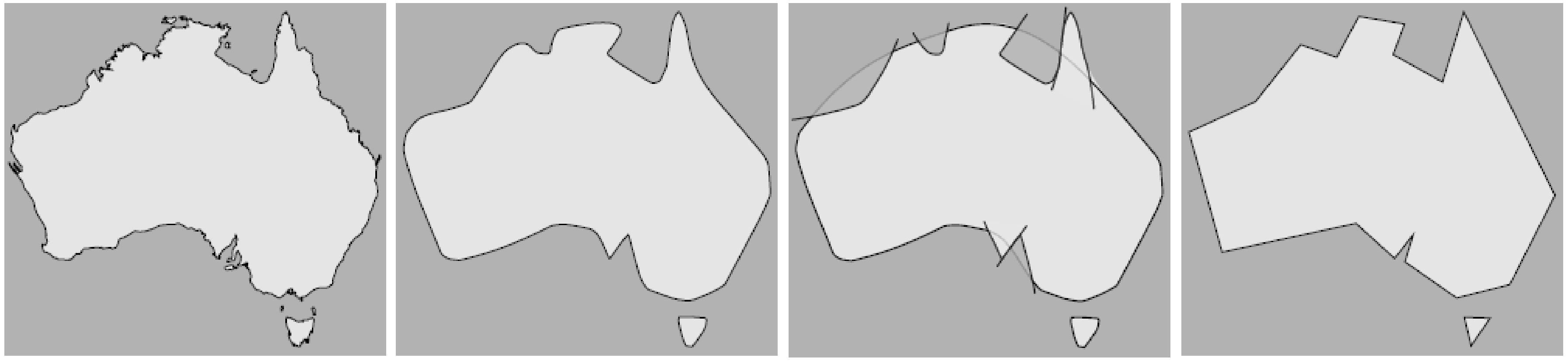
Regularization makes strokes that are nearly isometric, parallel, symmetric, perpendicular etc. precisely so!

Stroke recognition



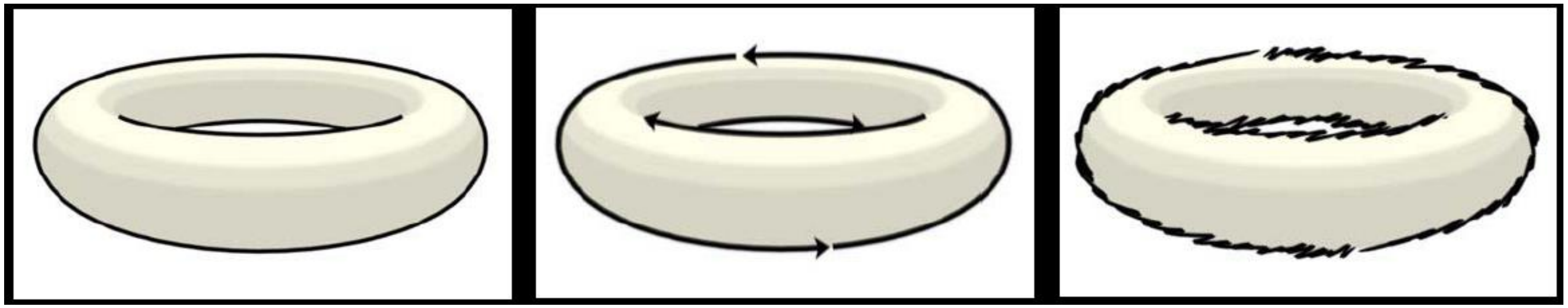
Stroke Abstraction

Stroke neatening that captures the essence of the stroke.



[**D. De Carlo & …**, *Fitting*.]

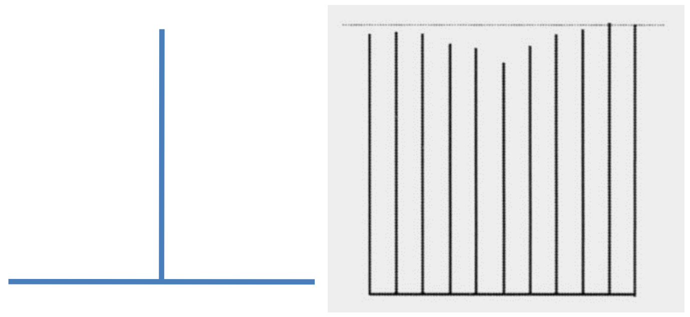
Stroke Appearance: NPR



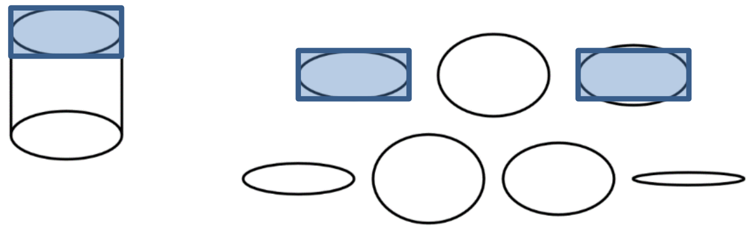
silhouettes | brushes | strokes

ink color
brush width
brush texture
paper texture

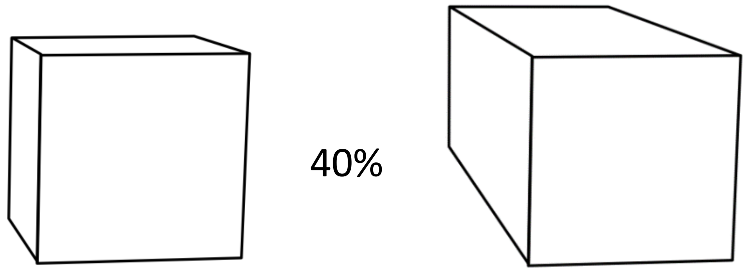
Stroke Perception



[**Wolfe, Maloney & Tam**, Distortions of perceived length in the frontoparallel plane: tests of perspective theories, *Perception & pyschophysics, 2005*]

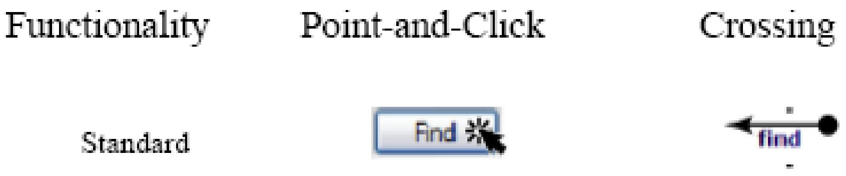


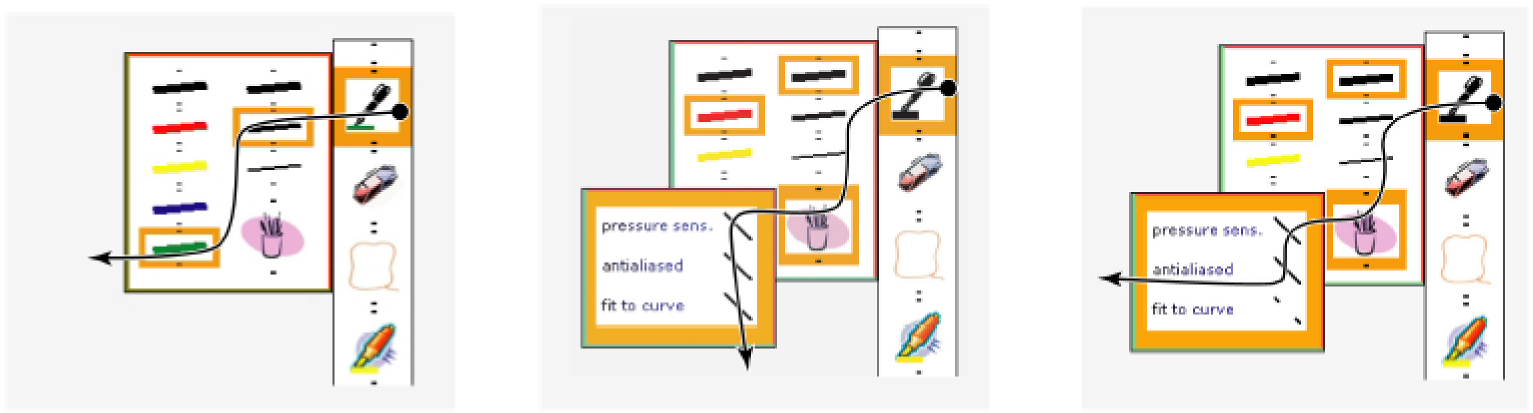
[**Taylor & Mitchell**, Judgements of apparent shape contaminated by knowledge of reality: viewing circles obliquely, *British Jnl. of Psych., 1997*]



[**Schmidt, Khan, Kurtenbach, Singh,** On expert performance in 3D curve drawing tasks. *SBIM 2009*]

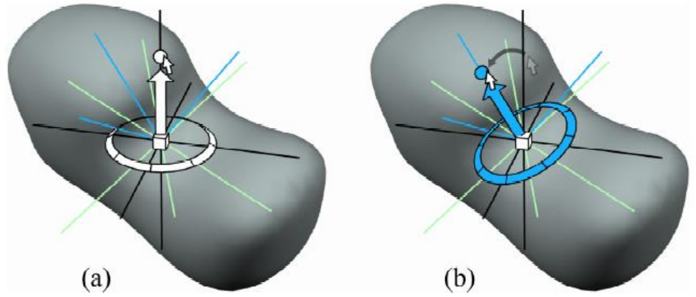
Stroke UI: crossing



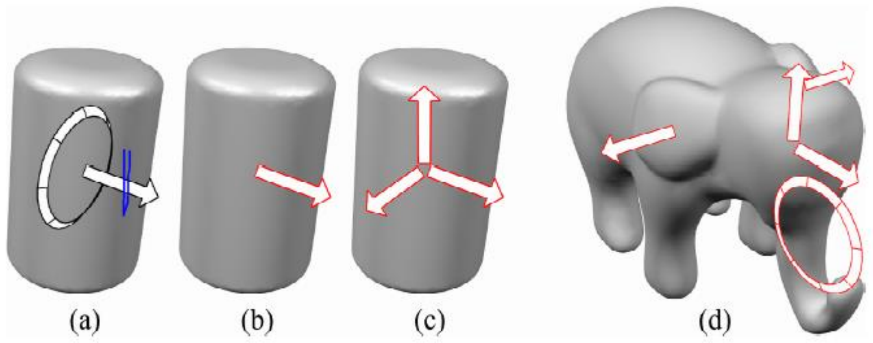


[**Apitz, G. and Guimbretière, F.** *CrossY: A Crossing-Based Drawing Application ACM UIST, 2004*]

Stroke UI: widgets



suggested axes



crossing interaction and composition

[**Schmidt, Singh & Balakrishnan** Sketching and Composing Widgets for 3D Manipulation, *Eurographics 2008*]

Stroke UI: gestures

- Ad-hoc or pre-defined:
  - Application specific: shorthand, chinese Brush Painting, musical scores, chemical formulas.
  - Platform specific: gesture libraries.

- Template-based:
  - Toolkit or framework
  - Simple algorithm based on geometric matching

Ad-hoc vs. template-based

- Ad-hoc can recognize more complex gestures.
- Harder to train template-based gestures.
- Better consistency of gestural use in ad-hoc systems.
- Better gesture collision handling in ad-hoc systems.

- Ad-hoc doesn’t allow new gestures and limited customization.

GRANDMA

1. Encode gestures as a linear function of 13 features.
2. Draw a gesture ~15 times.
3. Train asset of feature weights for each gesture.
4. Classify gestures based on highest feature function score.

[**D. Rubine.** *Specifying gestures by example.* SIGGRAPH 1991]

$1 recognizer goals

- Resilience to sampling.
- Require no advance math.
- Small code.
- Fast.
- 1-gesture training.
- Return an N-best list with scores.

[**J. Wobbrock, A. Wilson & Y. Li.** 2007. *Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes.* ACM UIST '07.]

$1 algorithm

- Resample the input
  - N evenly spaced points
- Rotate
  - “Indicative” angle between centroid and start point
- Scale
  - Reference square
- Re-rotate and Score
  - Score built from average distance between candidate and template points

$1 limitations

- Cannot distinguish aspect ratios, orientations.
  - Square from rectangle
  - Up arrow from down arrow
- Cannot be distinguished based on speed.
- Only single strokes.
- Stroke order is important.
- Closed strokes?
- Gestalt gestures!

Take-aways

- Understand your application:
  - Does it need strokes?
  - Are strokes natural and of low-complexity, 2D or 3D?
- Understand source of stroke error before filtering?
- Ensure users can control stroke dynamics before you exploit it.
- Both clean and sketchy stroke appearances are useful.
- Understand perceptual bias in drawn strokes.
- Develop a GUI suited to stroke interaction.

Multi-view sketch-based modeling of 3D curves and surfaces

Yotam Gingold

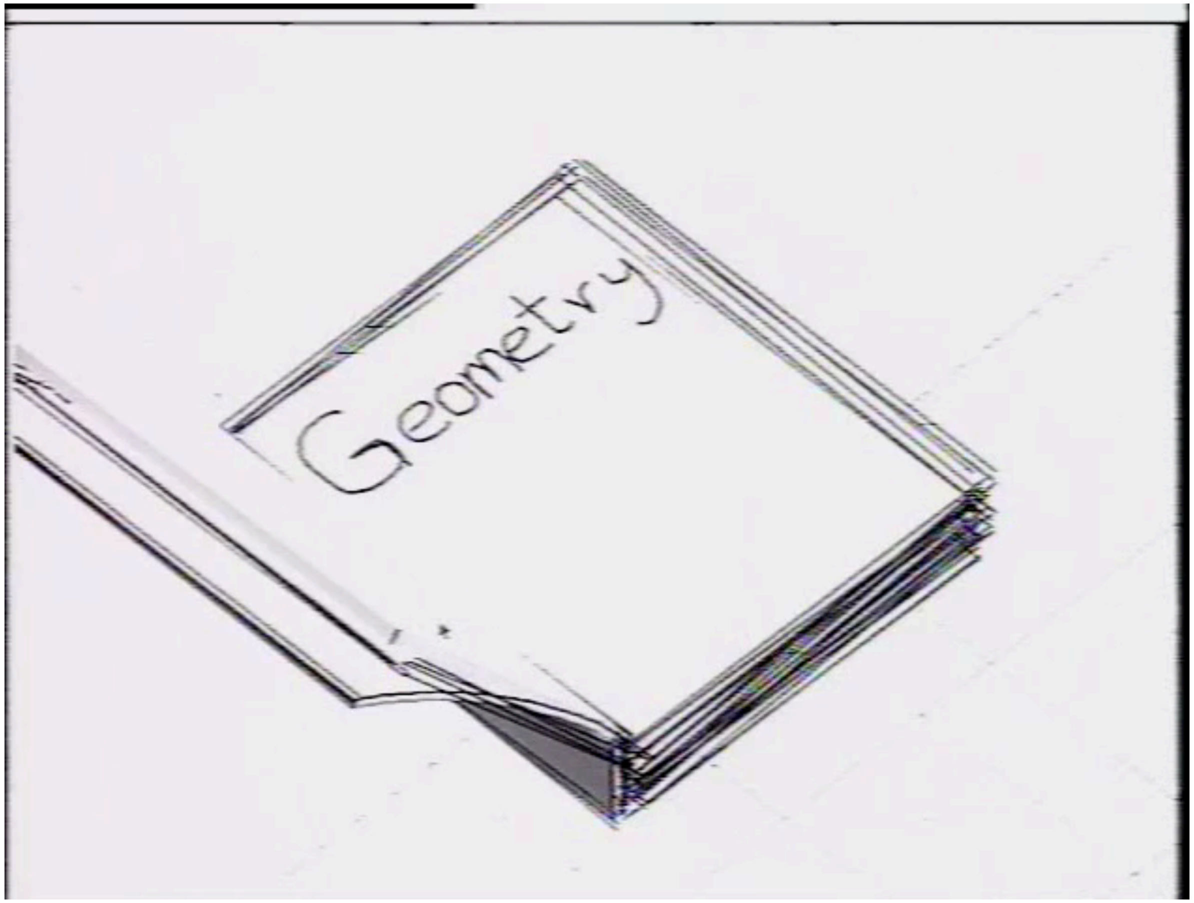
How can we turn sketch strokes into 3D shapes?

- Interpreting them as gestures
- Interpreting them as silhouettes
- Projecting them in 3D

- General principle: Drawing from different points of view.

Interpreting stokes as gestures

SKETCH



SKETCH: An Interface for Sketching 3D Scenes [Zeleznik et al. 1996]

Interpreting strokes as silhouettes

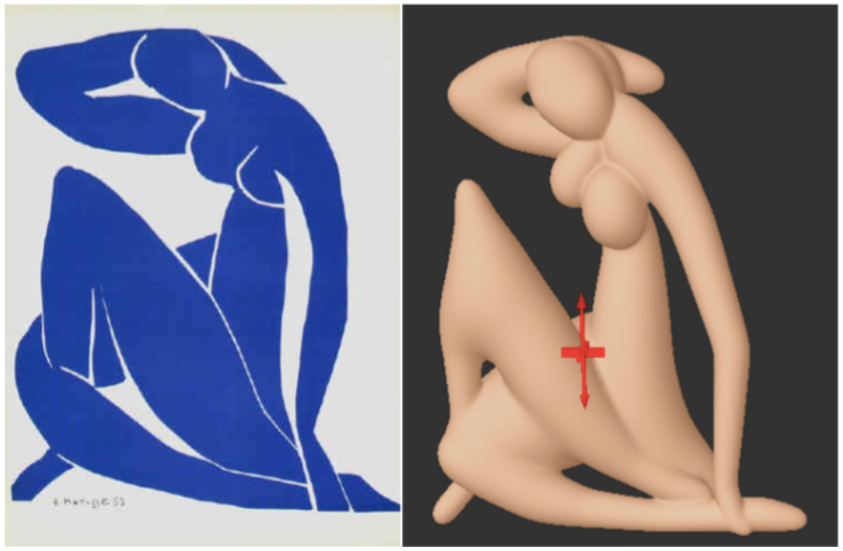
Teddy

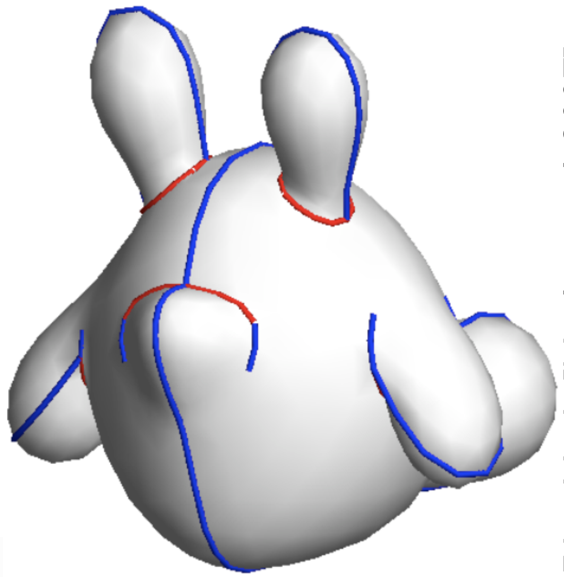
Takeo Igarashi
Hidehiko Tanaka
University of Tokyo
Satoshi Matsuoka
Tokyo Institute of Technology

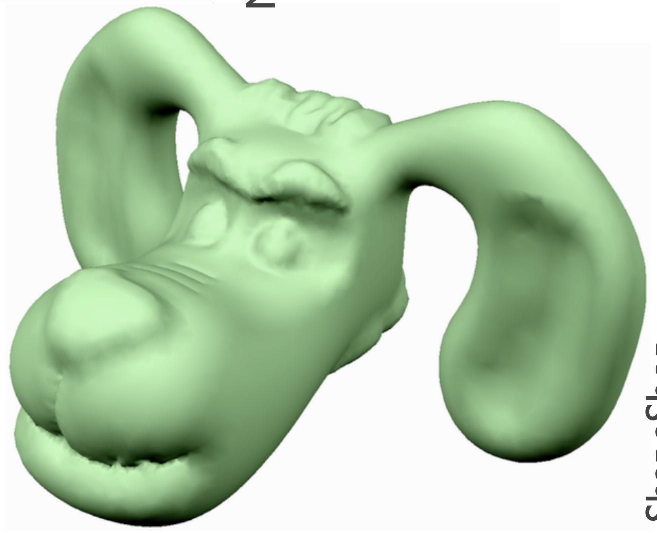
Teddy:
A Sketching
Interface for 3D
Freeform Design

Teddy: A Sketching Interface for 3D Freeform Design [Igarashi et al. 1999]

More freeform inflation approaches


Matisse [Bernhardt et al. 2008]

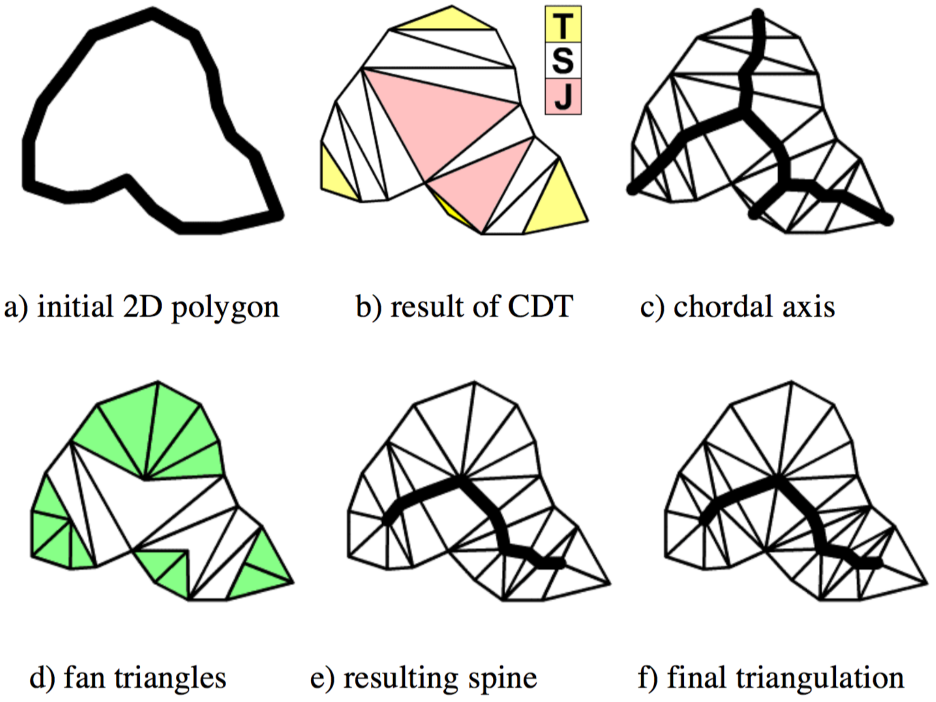

FiberMesh [Nealen et al. 2007]


ShapeShop [Schmidt et al. 2005-8]

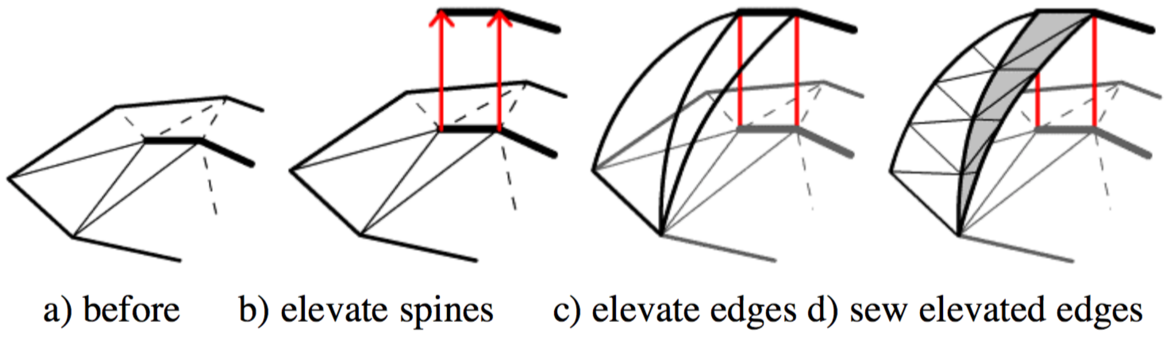
EUROGRAPHICS 2016

Inflation (Teddy)

- Step 1

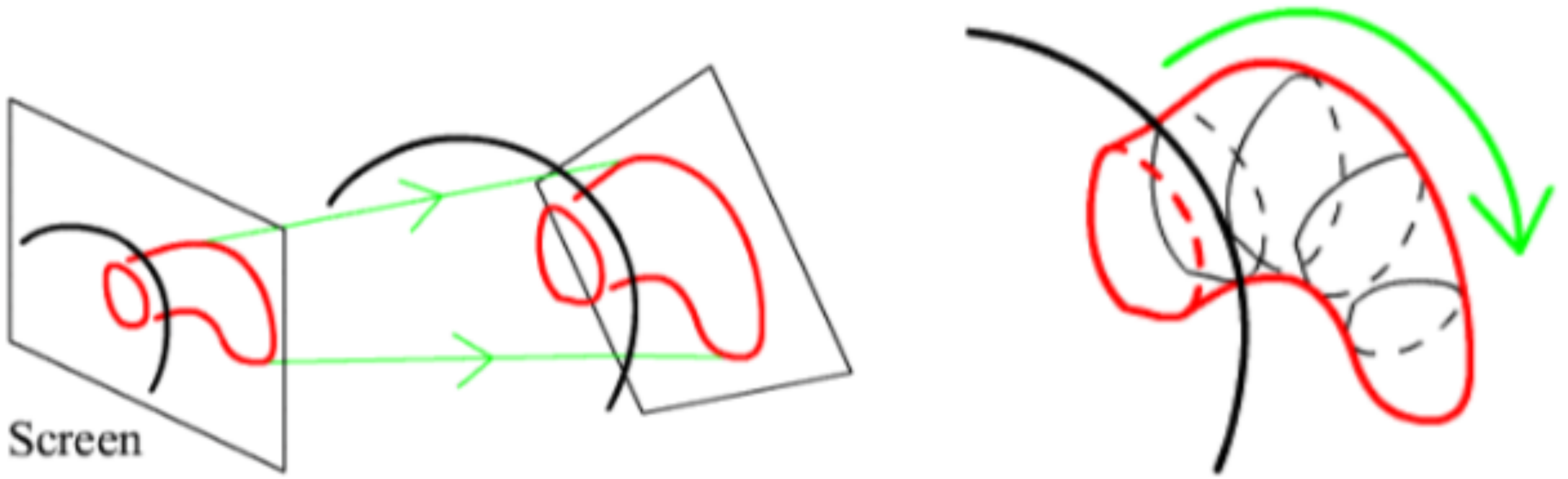


- Step 2



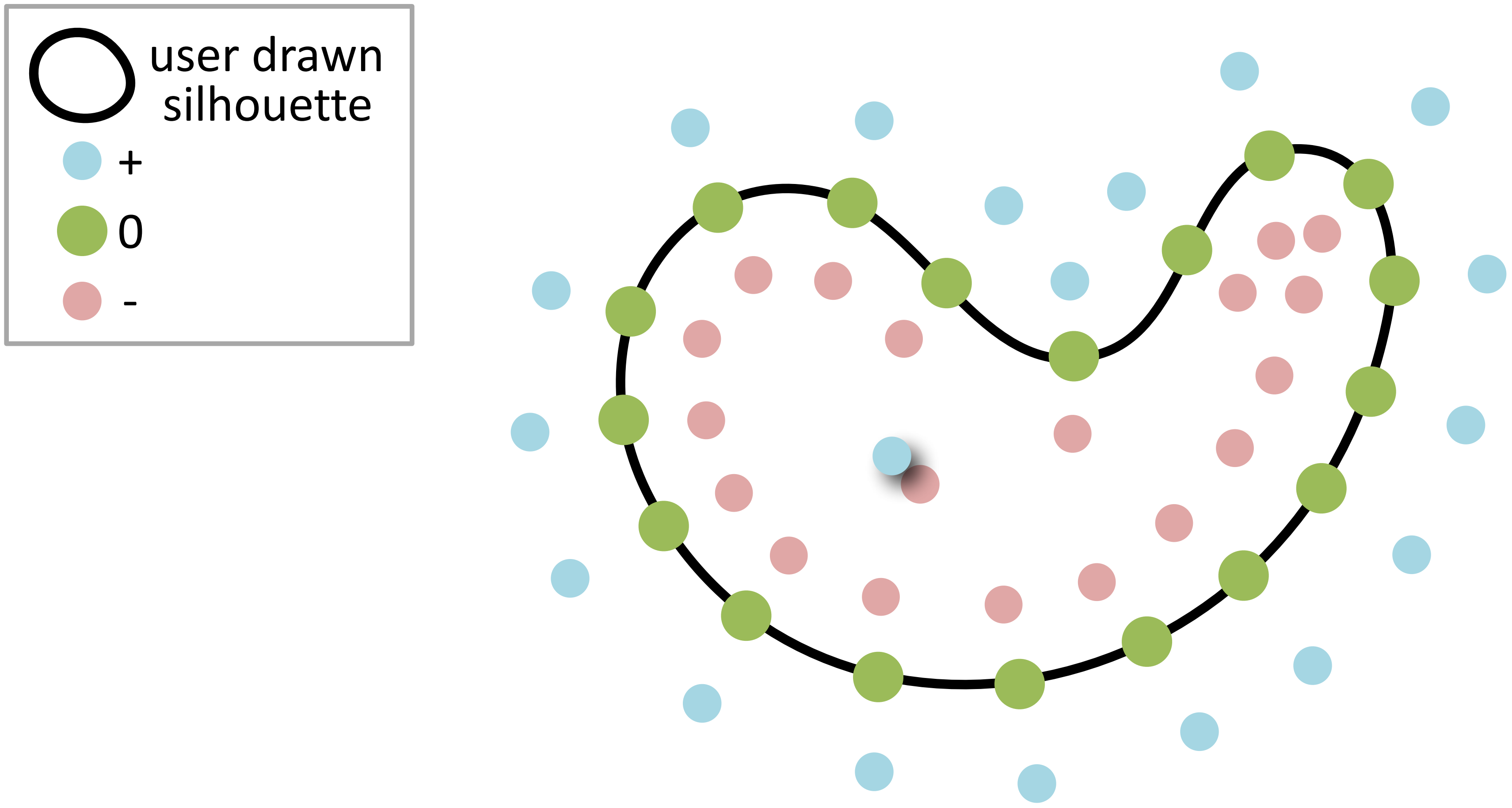
EUROGRAPHICS 2016

Extrusion (Teddy)



a) projection of the stroke b) sweep along the projected stroke

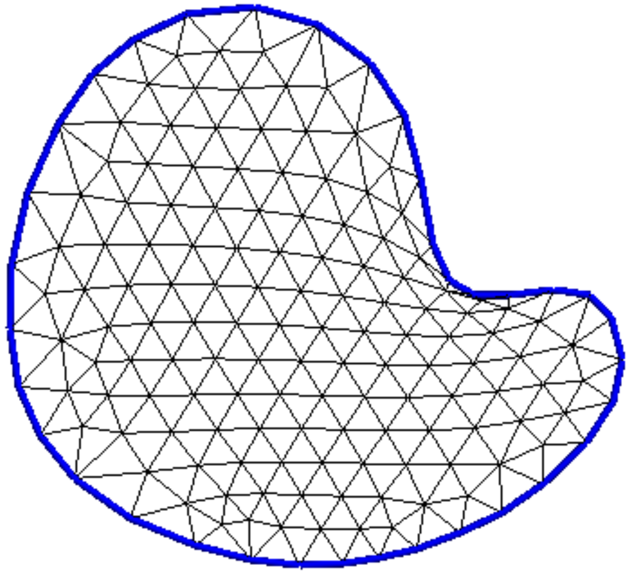
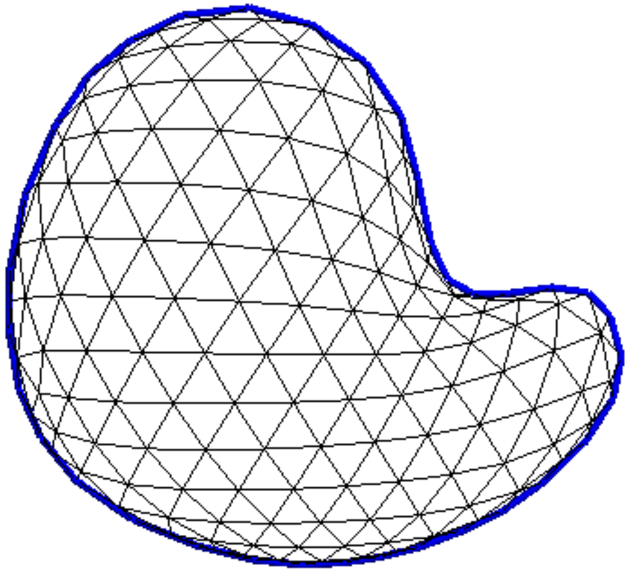
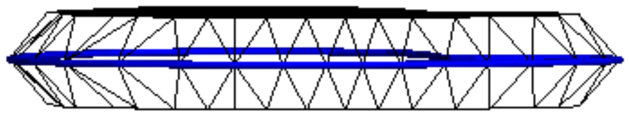
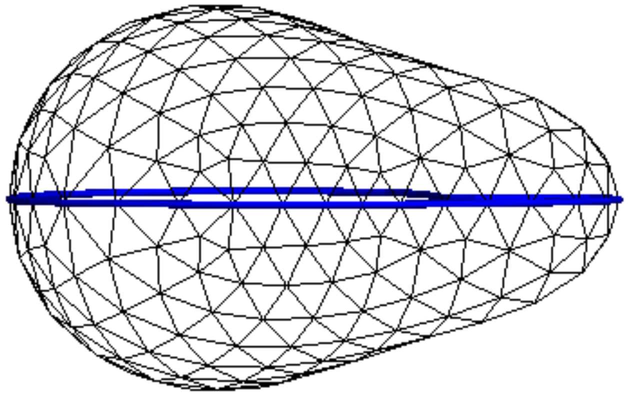
Inflation (implicit surface)



Inflation (surface optimization)

initial mesh

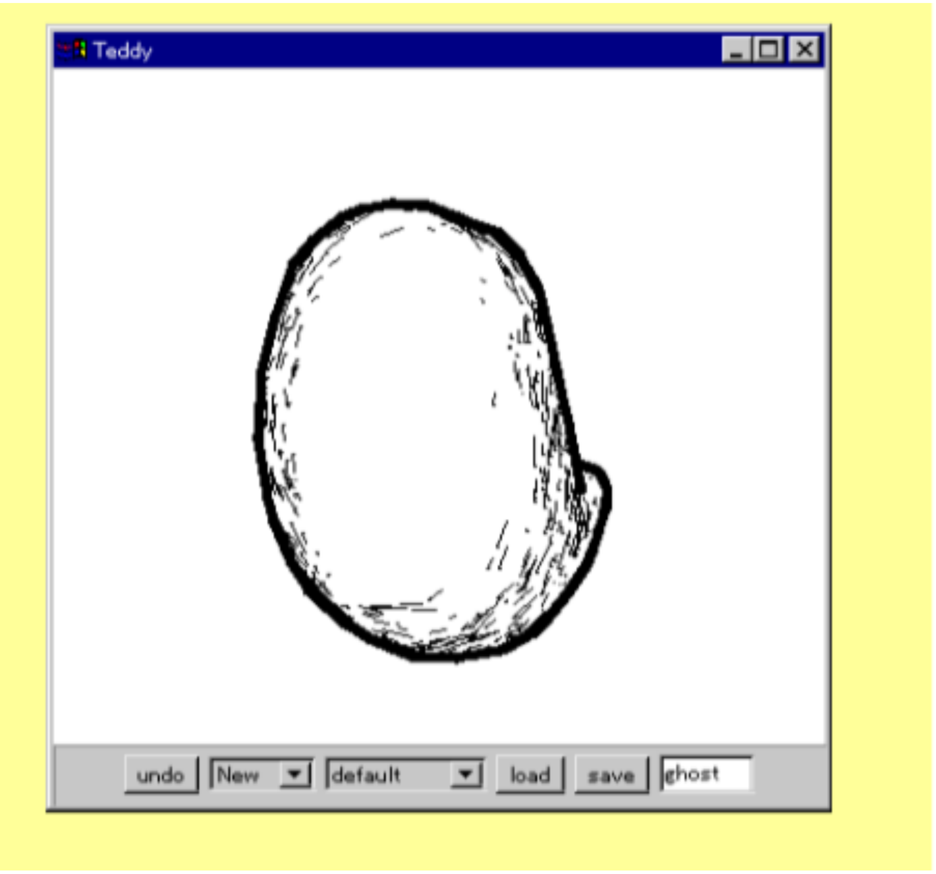
minimum variation of curvature

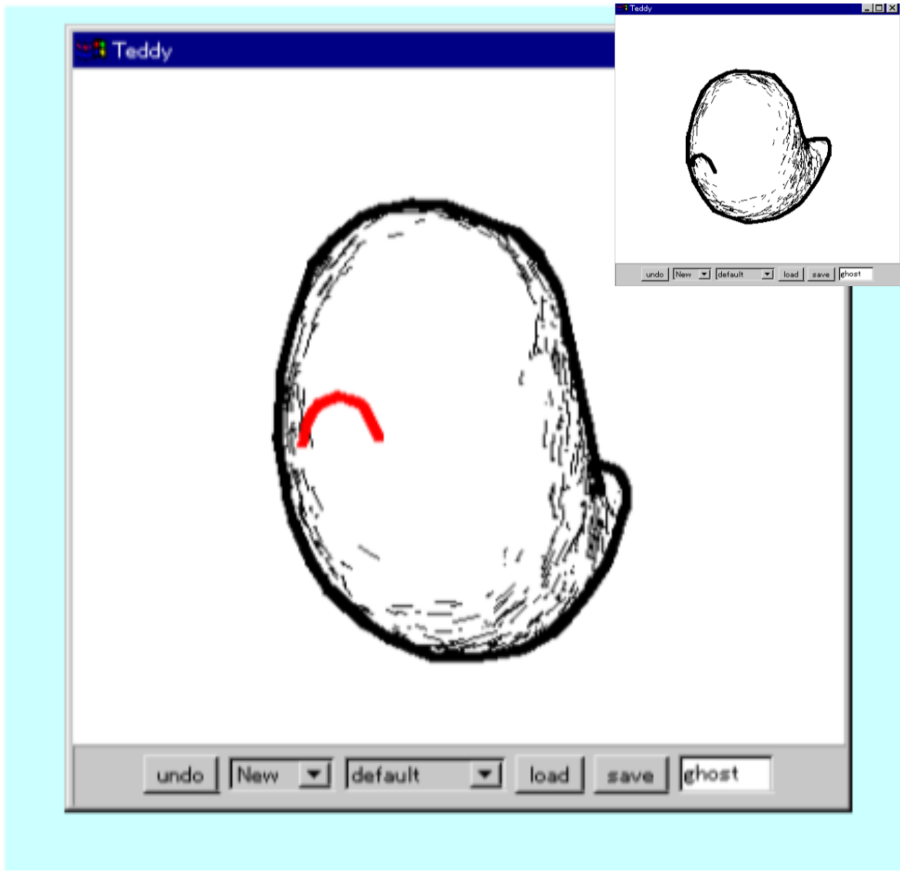
FiberMesh [Nealen et al. 2007]

EUROGRAPHICS 2016

Painting (Teddy)

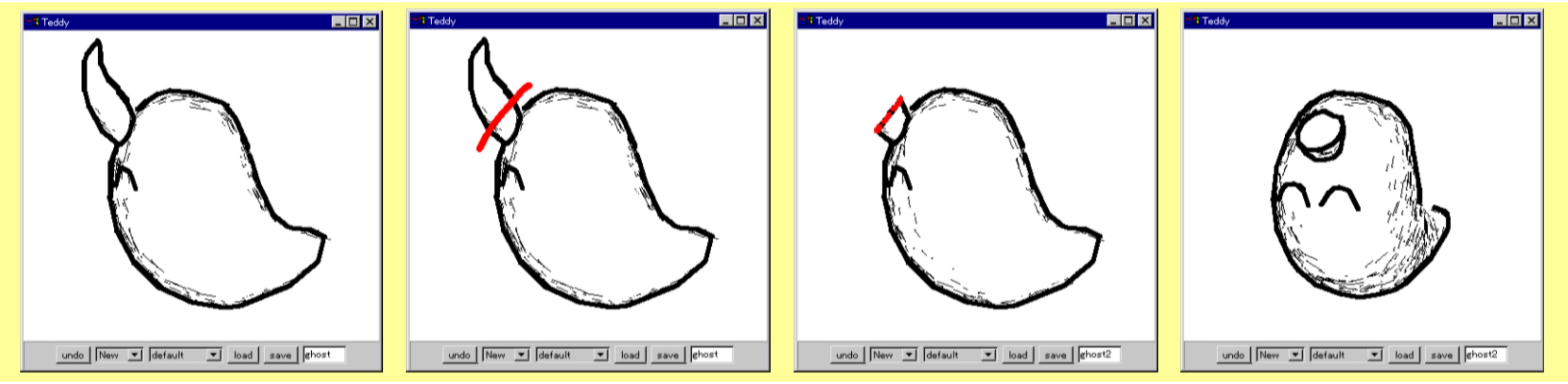


Before



After

Cutting (Teddy)



Before Cutting stroke After

Projecting strokes in 3D

Ambiguity

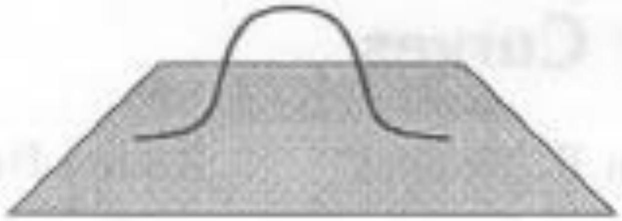


Figure 1: A single stroke creates the initial curve.

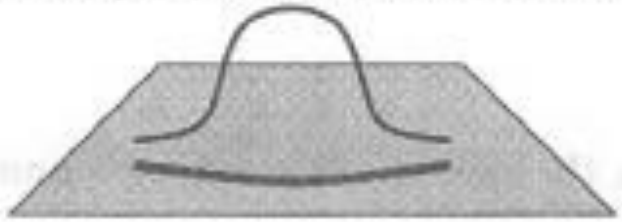


Figure 2: A second stroke defines the curve's shadow and hence its 3D shape.

An Interface for Sketching 3D Curves [Cohen et al. 1999]

Ambiguity

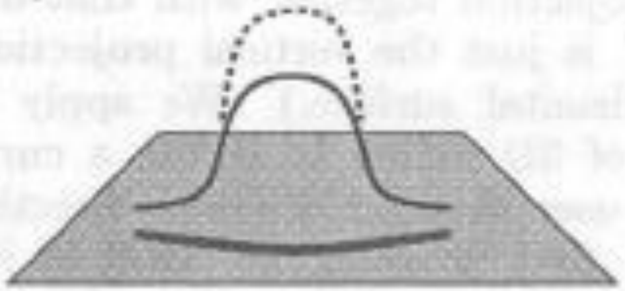


Figure 3: The dashed line indicates an overdraw stroke.

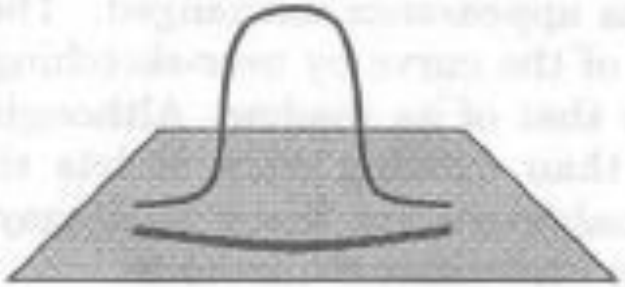


Figure 4: The system blends the overdraw with the original curve to get the final result.

An Interface for Sketching 3D Curves [Cohen et al. 1999]

Ambiguity

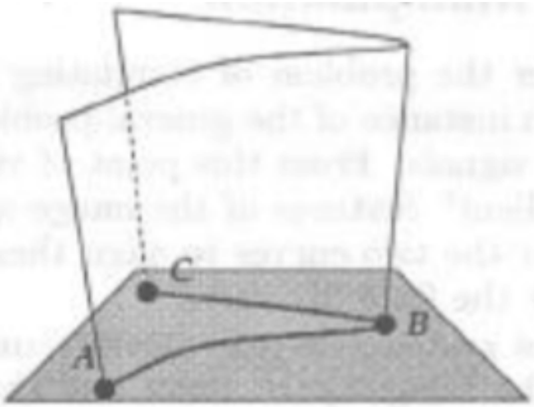


Figure 7: The shadow defines a ruled surface with a silhouette above the interior critical point of the shadow, B.

An Interface for Sketching 3D Curves [Cohen et al. 1999]

Ambiguity

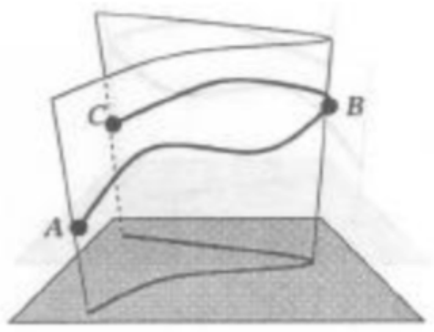


Figure 8: The curve must turn around at B to stay on the surface.

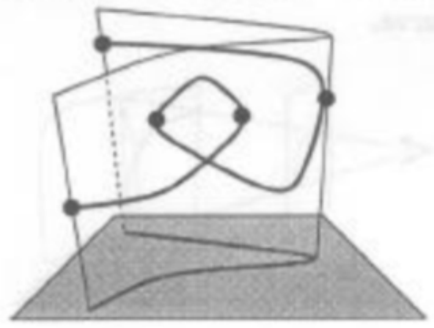


Figure 9: The curve may have more critical points than the shadow and still be valid.

An Interface for Sketching 3D Curves [Cohen et al. 1999]

Ambiguity

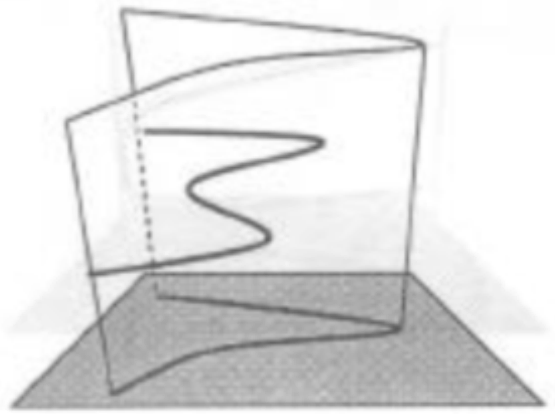
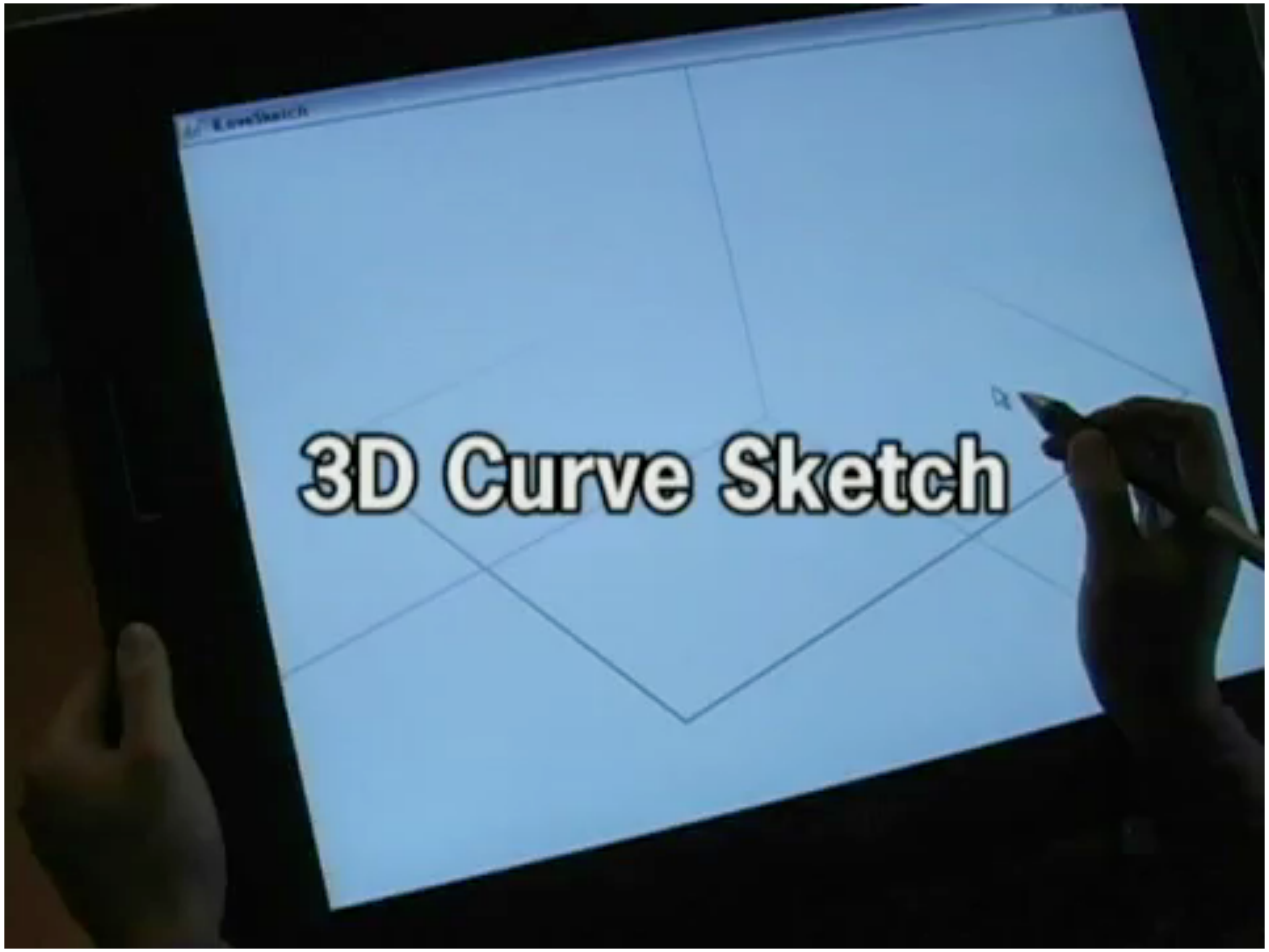


Figure 10: There is no way to project this curve onto the surface to get a continuous 3D curve.

An Interface for Sketching 3D Curves [Cohen et al. 1999]

iLoveSketch



iLoveSketch: As-natural-as-possible sketching system for creating 3D curve models [Bae et al 2008]

Takeaways

- We can remove the ambiguity in depth in several ways:
  - with initial assumptions (rotund surfaces)
  - by projecting onto other surfaces
  - by sketching from multiple points of view

References

[Zeleznik et al. 1996] Robert C. Zeleznik, Kenneth P. Herndon, John F. Hughes: SKETCH: an interface for sketching 3D scenes. SIGGRAPH Courses 2007: 19

[Igarashi et al. 1999] Takeo Igarashi, Satoshi Matsuoka, Hidehiko Tanaka: Teddy: A Sketching Interface for 3D Freeform Design. SIGGRAPH 1999: 409-416

[Bernhardt et al. 2008] Adrien Bernhardt, Adeline Pihuit, Marie-Paule Cani, Loic Barthe: Matisse: Painting 2D regions for Modeling Free-Form Shapes. SBM 2008: 57-64

[Schmidt et al. 2005-8] Ryan Schmidt, Brian Wyvill, Mario Costa Sousa, Joaquim A. Jorge: ShapeShop: Sketch-Based Solid Modeling with BlobTrees. SBM 2005: 53-62

References

[Nealen et al. 2007] Andrew Nealen, Takeo Igarashi, Olga Sorkine, Marc Alexa: FiberMesh: designing freeform surfaces with 3D curves. ACM Trans. Graph. 26(3): 41 (2007)

[Cohen et al. 1999] Jonathan M. Cohen, Lee Markosian, Robert C. Zeleznik, John F. Hughes, Ronen Barzel: An interface for sketching 3D curves. SI3D 1999: 17-21

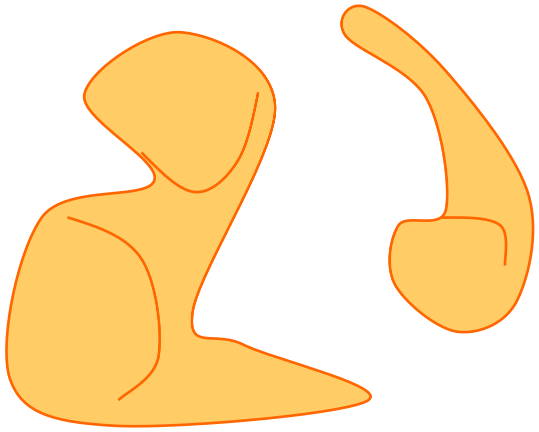
[Bae et al 2008] Seok-Hyung Bae, Ravin Balakrishnan, Karan Singh: ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. UIST 2008: 151-160

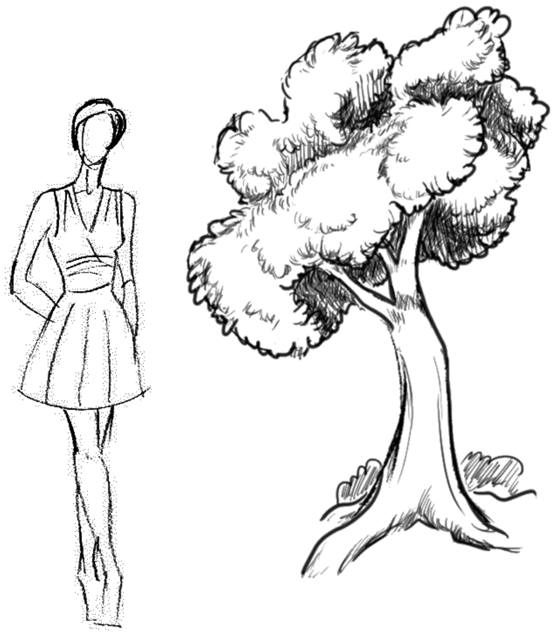
Sketch-based modelling using prior knowledge

Even Entem & Marie-Paule Cani

Grenoble University & Inria

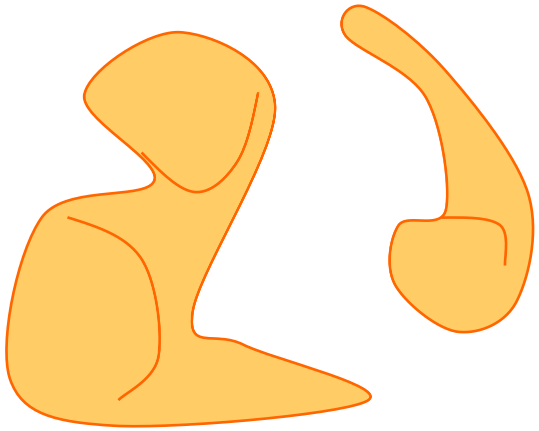
Use of prior knowledge : Motivation





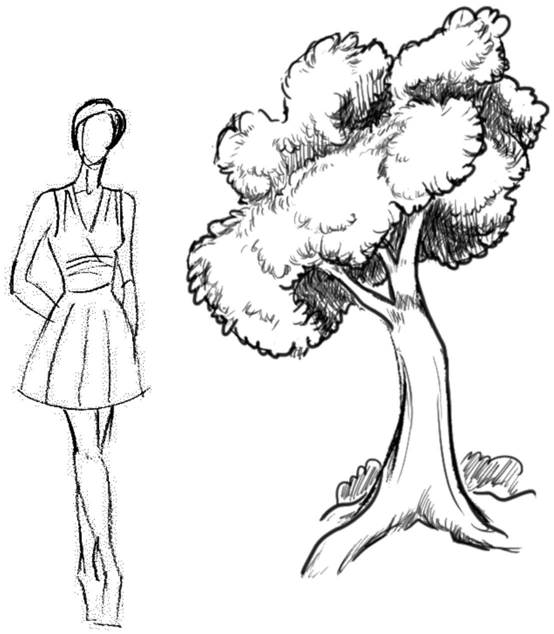
Why do we "see" 3D shapes when we look at a sketch?

Use of prior knowledge : Motivation



Unknown shape

- We infer the simplest one



Well known shapes

- We use prior knowledge

Use of prior knowledge : Motivation

Well known shapes

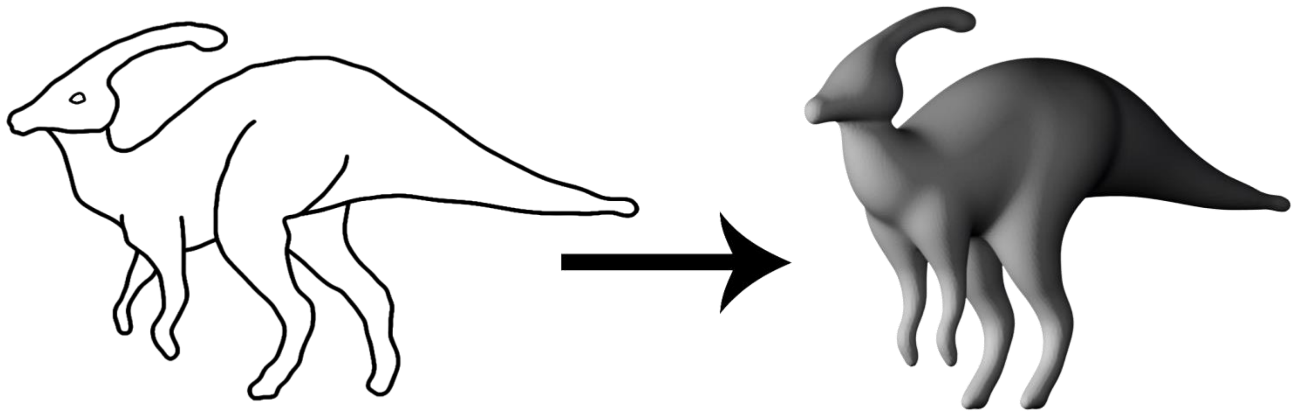
- Model prior knowledge
- It can help us to infer 3D from a single sketch!

Two examples in this talk

- Animals
- Garments

Sketching specific shapes
Example 1: Animals from a side-view sketch

Goal: Modeling animals and other creatures from a single sketch



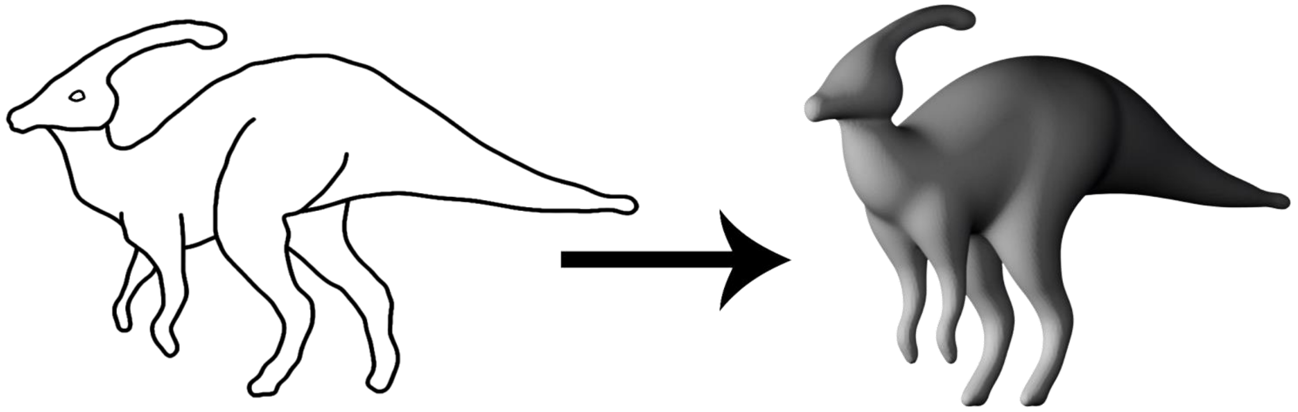
Prior knowledge:

- Organic shapes: Rather smooth, volumetric shapes
- Structural symmetries (limbs in arbitrary postures)
- Anatomic principles

Animals from a side-view sketch

Strategy:

- What kind of drawing gives the best trade-off between user-friendliness and explicitness? In terms of:
  - View: Side
  - Style: Processed sketch (beautified and regularized)



Animals from a side-view sketch

What kind of 3D shape representation to use ?

– Inflated polygon meshes:

~ Enable flat areas and full control over the geometry. But ambiguities inherent to the sketch make it unnecessary

- Post-editing is not user-friendly

– Convolution surfaces:

+ Ensures the smoothness of the shape

+ 3D line skeleton suited for user-friendly post-editing

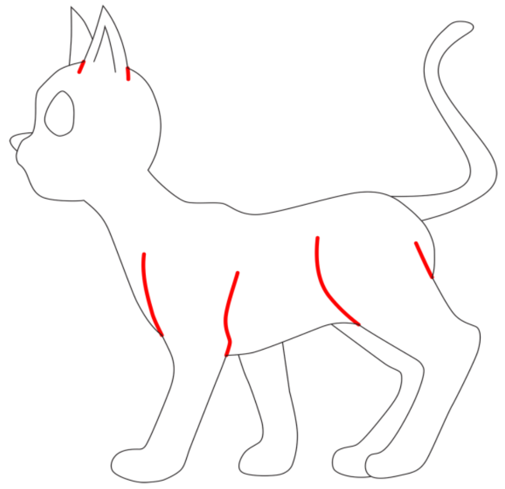
Animals from a side-view sketch

How to infer a 3D line skeleton from the sketch ?

- Perceptual process: segmentation into subparts
  - Subparts are partially/fully bounded by curves and may be partially occluded.
  - Depth ordering from cues ("T-junctions" and inclusions)

-> Identify curves in terms of meaning

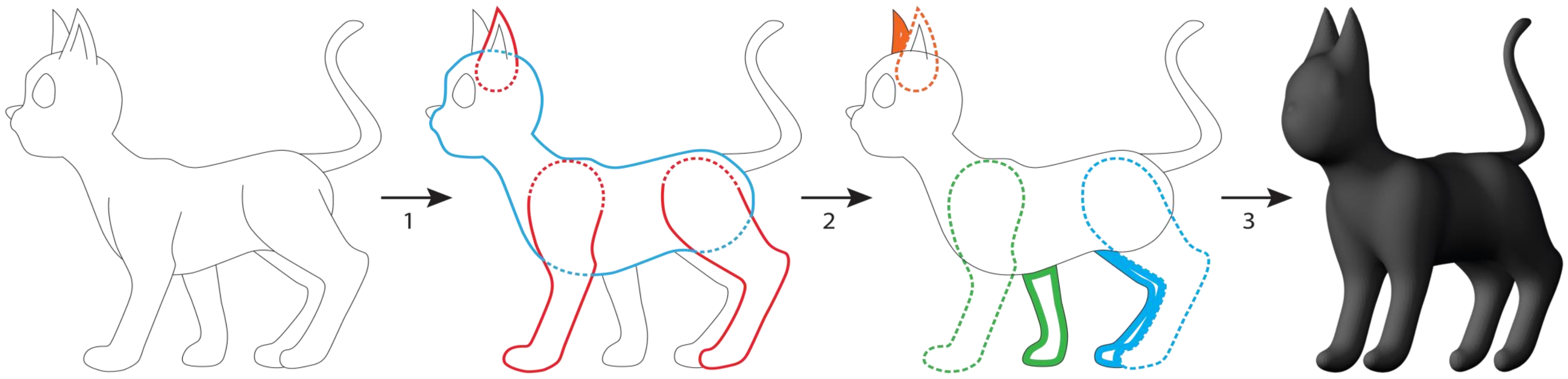
- Silhouette contours
- Suggestive contours
  - Silhouette in most nearby views



Animals from a side-view sketch

Identify the ambiguities and tasks

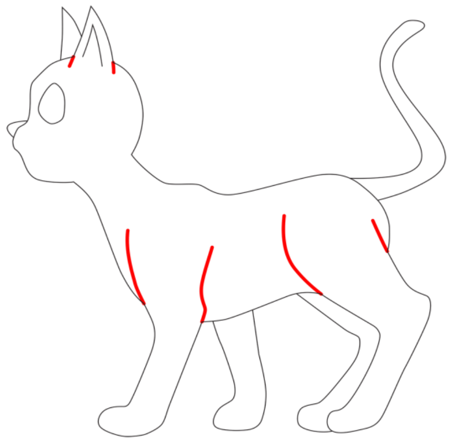
- Suggestive contours pairing (and closures)
- Structural symmetries in the background
- Depth positioning

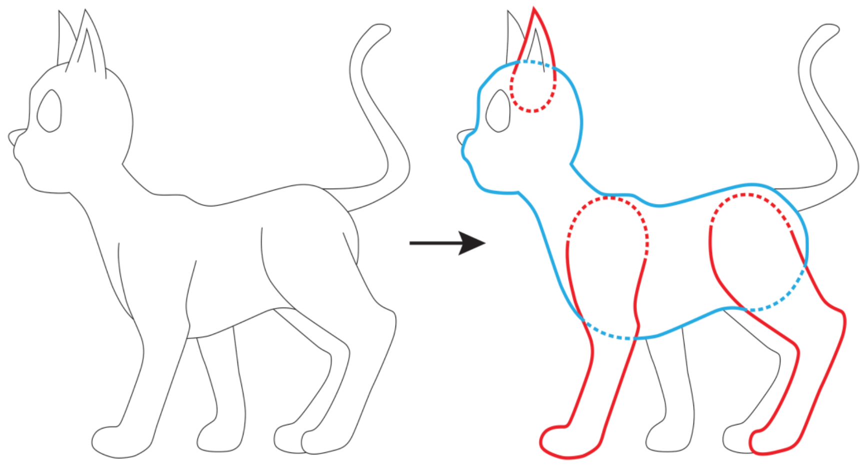


Animals from a side-view sketch

Identify the ambiguities and tasks

- Suggestive contours pairing (and closures)
- Structural symmetries in the background
- Depth positioning

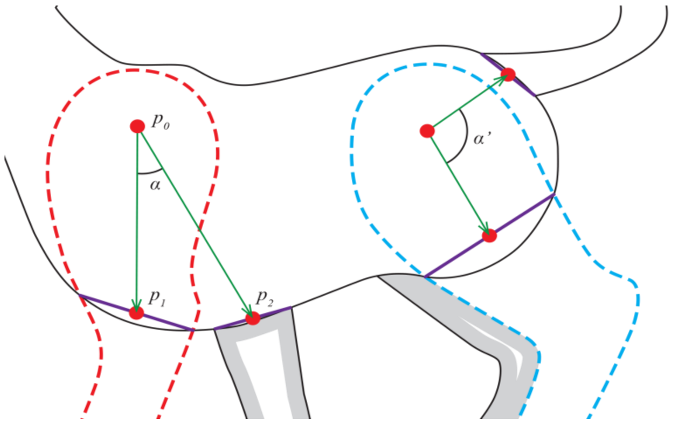


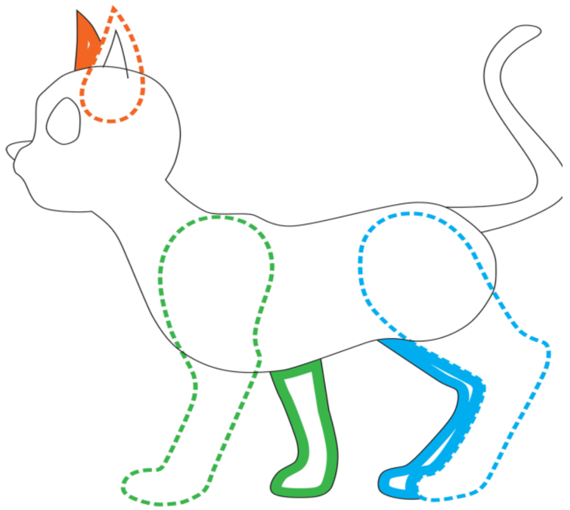


Animals from a side-view sketch

Identify the ambiguities and tasks

- Suggestive contours pairing (and closures)
- Structural symmetries in the background
- Depth positioning

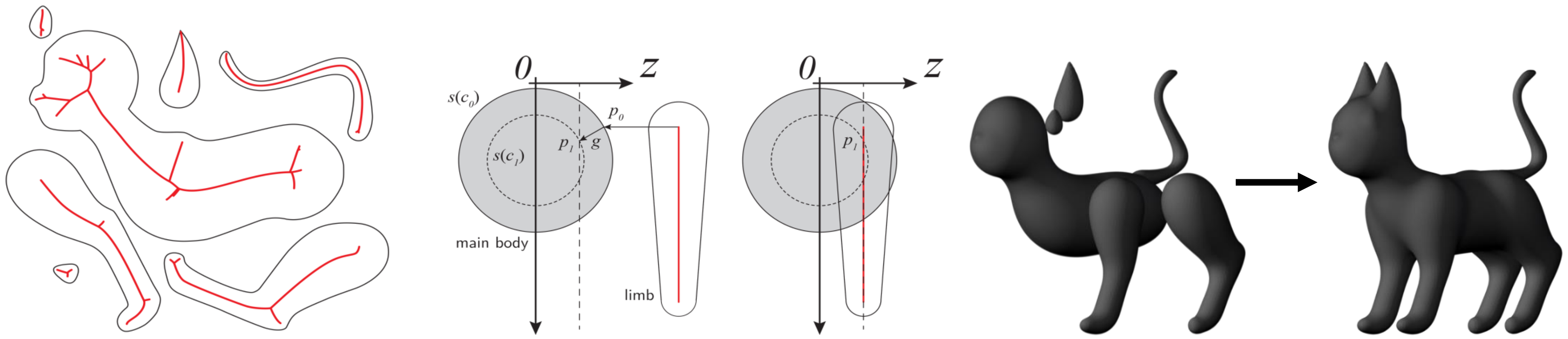




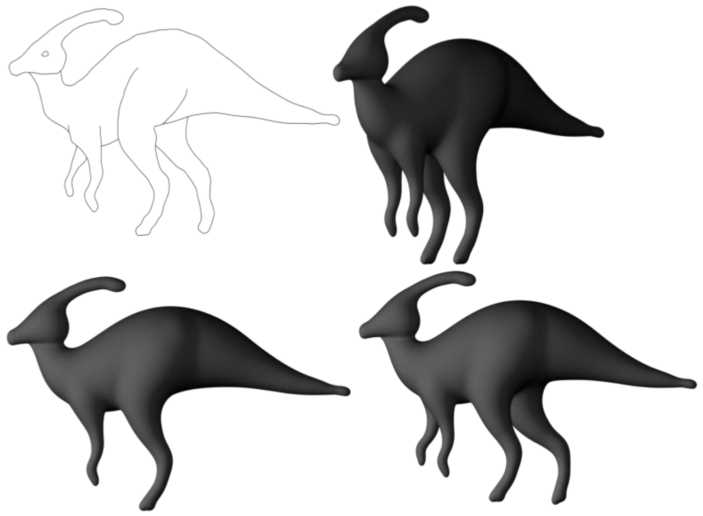
Animals from a side-view sketch

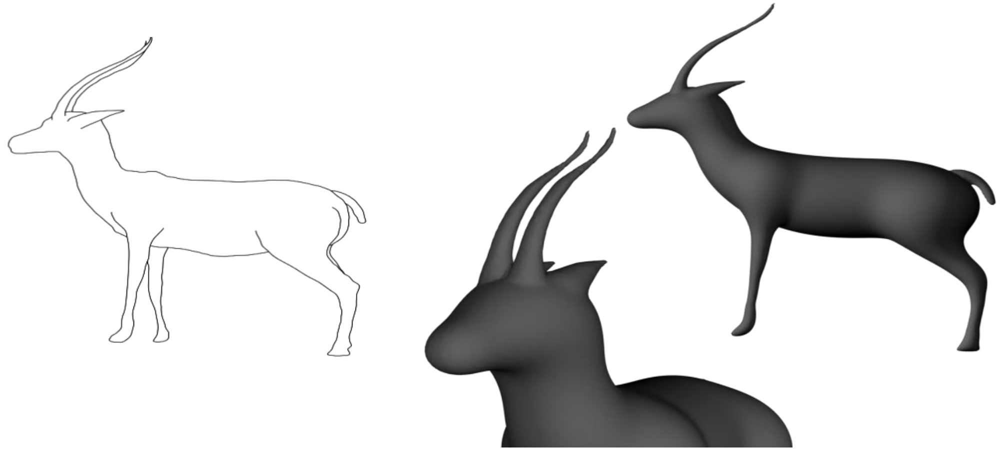
Generation of the 3D model

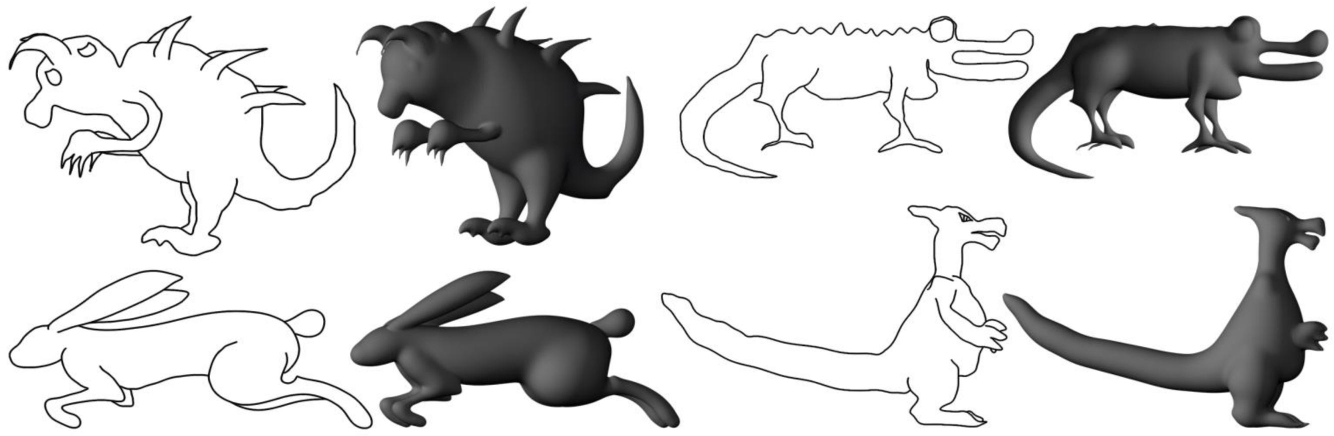
- Medial-Axis to get skeleton lines
- Prior knowledge let us define relative depths
  - "flesh around bones" considering lateral agonist and antagonist muscles equally developed.



Animals from a side-view sketch





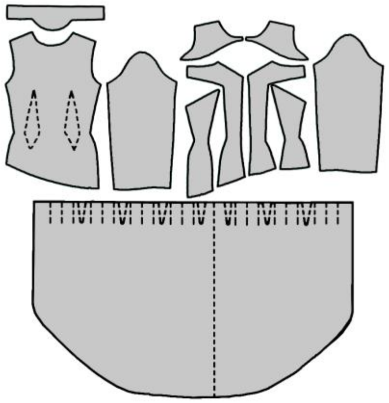


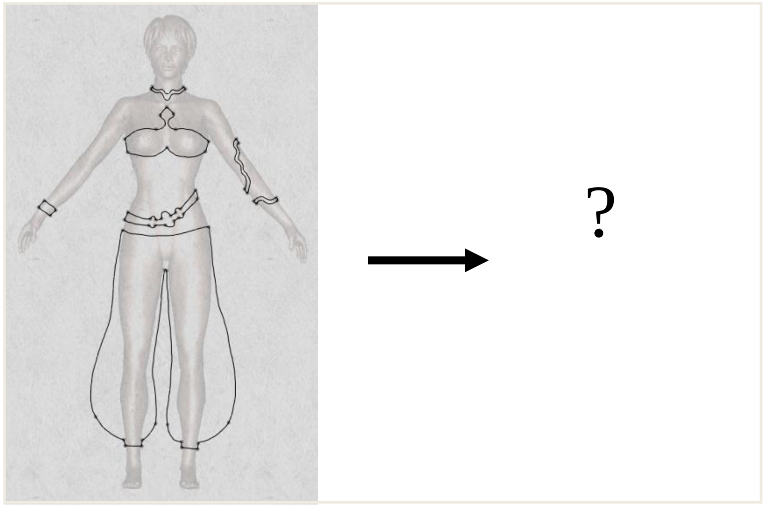
[Entem, Barthe, Cordier, Cani, Van de Panne, SMI'2014]

Sketching specific shapes
Example 2: Clothing design

Standard virtual clothing in Computer Graphics

- Design & place patterns
- Run a simulation!





3D model from 2D fashion sketch?

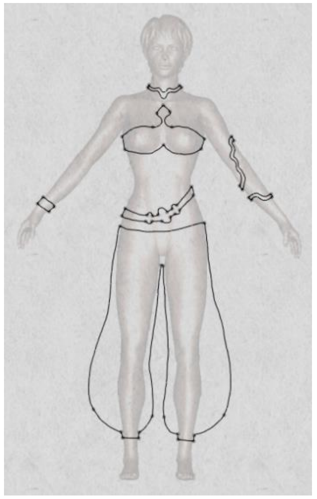
→ would compute the patterns!

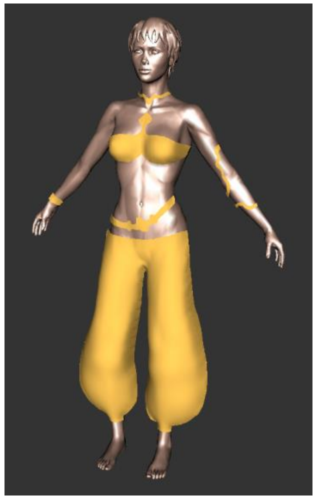
Clothing design

Using silhouette information only

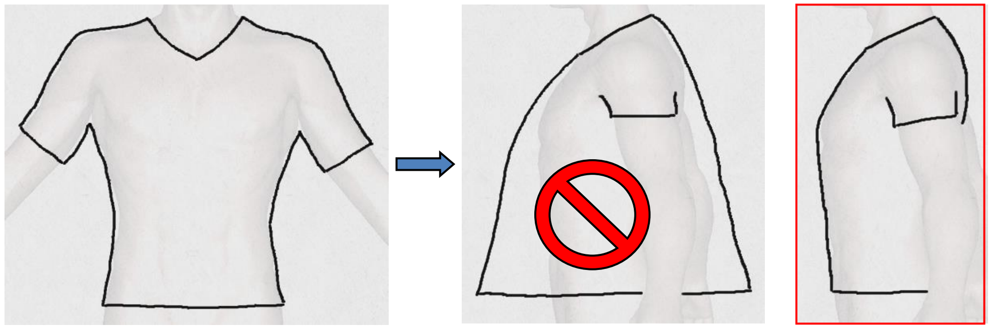
Virtual clothing from a sketch?

- Sketch on a view of a 3D model

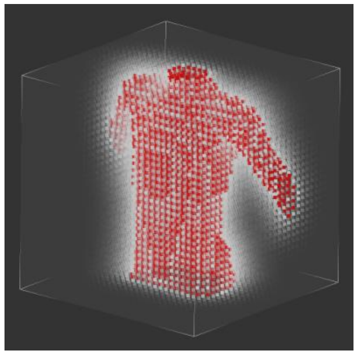




- Knowledge? Rule of thumb:
  - – Fitting is the same in all directions!



Sketch in a distance field!

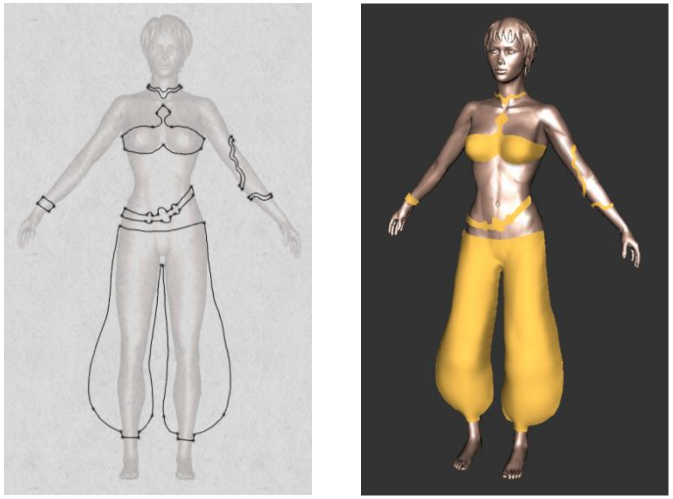


Clothing design

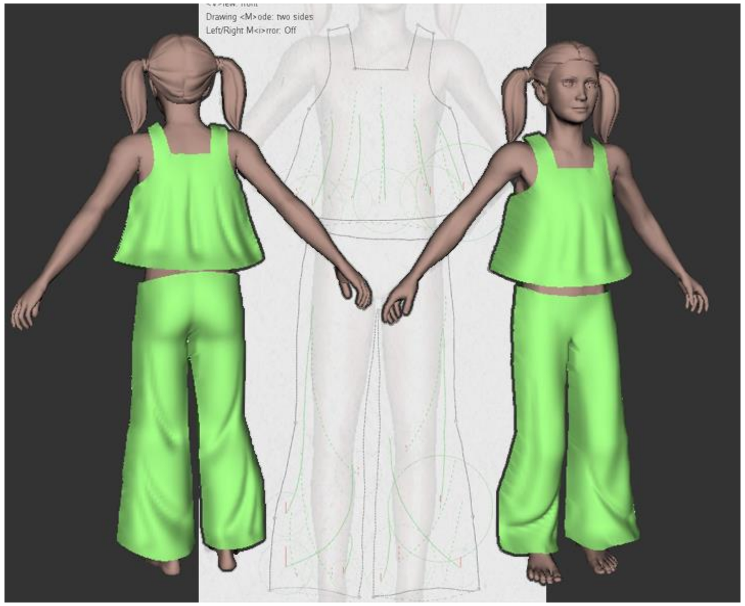
Using silhouette information only

Results lack folds!

- Allow the designer to sketch them?



[Turquin, Cani, Hughes 2004]



Drawing <M>ode: two sides
Left/Right M<i>rror: Off

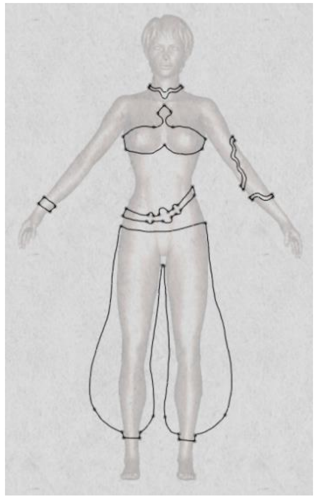
[Turquin, Cani, Hughes 2007]

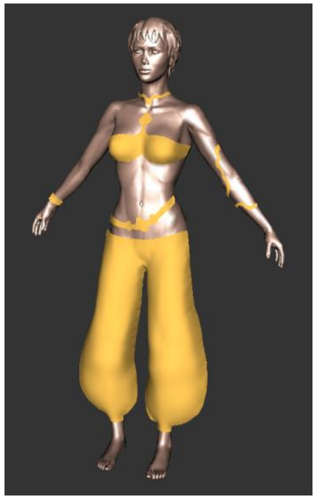
Nice if the designer is good!

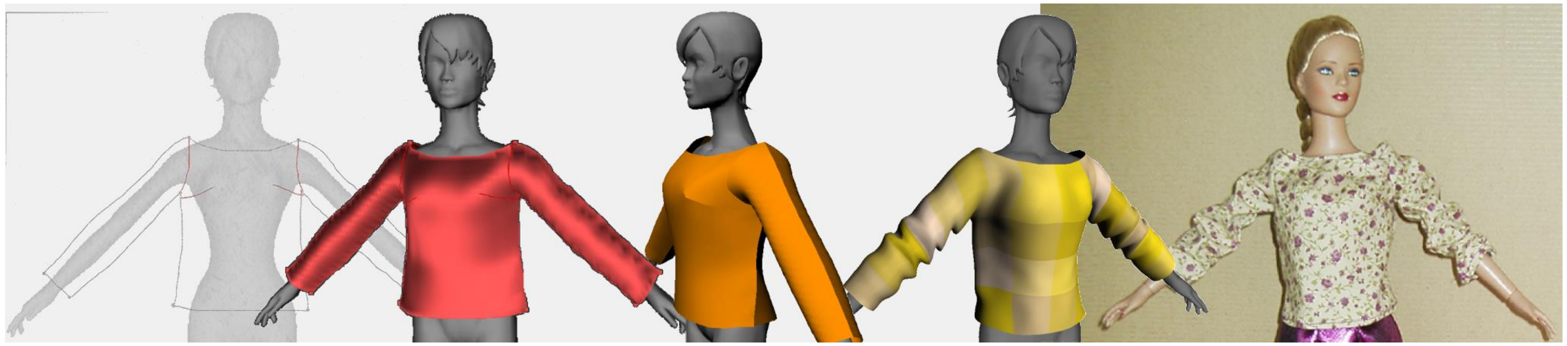
Clothing design
Using silhouette information only

Results lack folds!

- Ask the designer to sketch them…
- Or use more a priori knowledge?
  - Garment is piece-wise developable
  - Folds can be computed







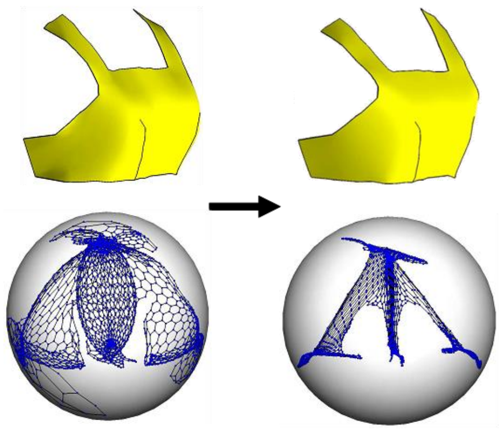
EUROGRAPHICS 2016

Clothing design

Developable surfaces from a sketch

Developable surface from sketch?

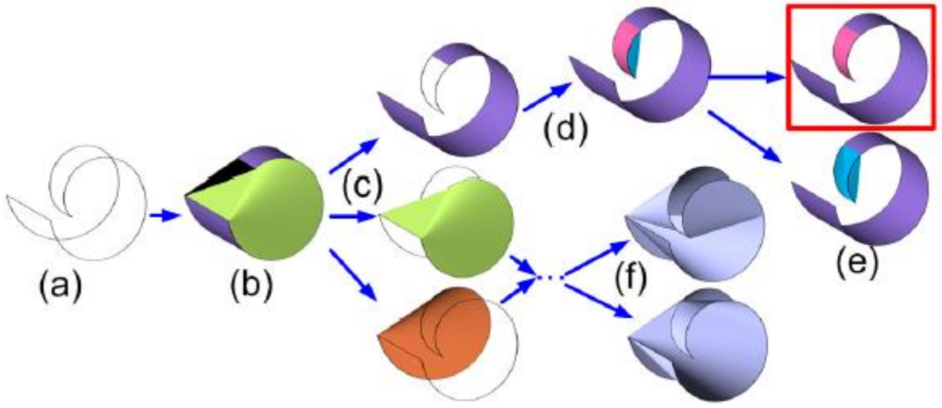
- Solution 1: increase developability
  - Start with the rough surface
  - Locally optimize the shape (1D normal map)

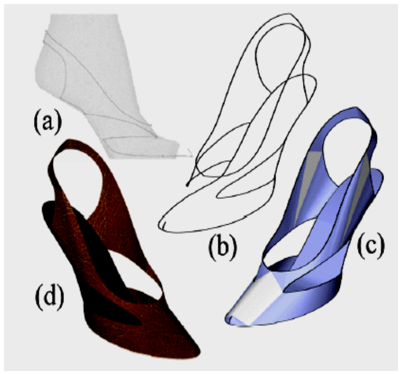


- Solution 2: smooth developable surface from contours

Convex edges?

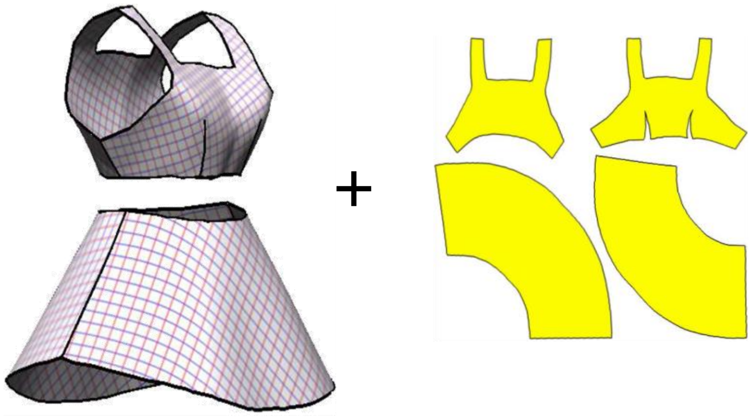
Recursively split & triangulate the convex hull





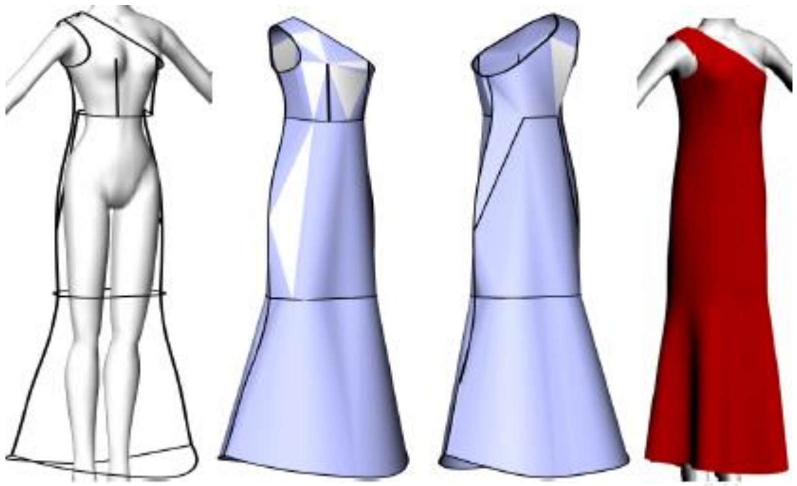
Clothing design
Developable surfaces from a sketch

Results still lack folds!



Run a simulation?

- Physically-based parameters to set up
- Stiff system for un-extensible cloth

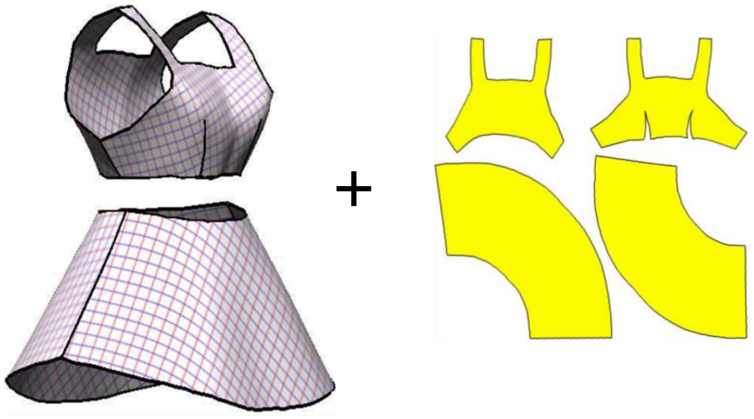


EUROGRAPHICS 2016

Clothing design

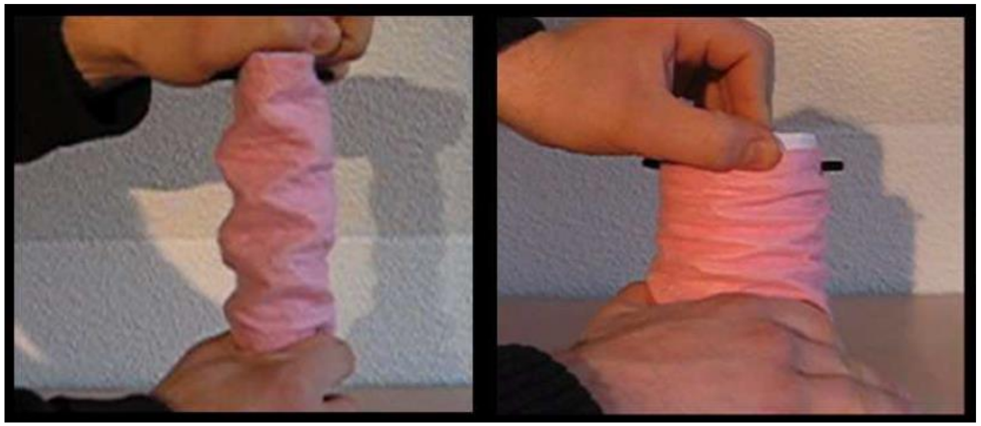
Developable surfaces from a sketch

Results still lack folds!



Or use more knowledge…

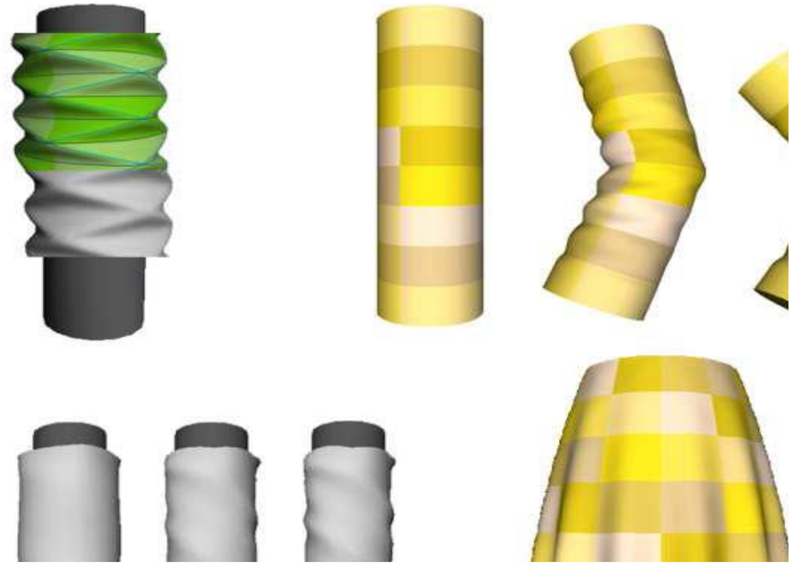
- Cloth wrapped on cylinders always folds the same way!

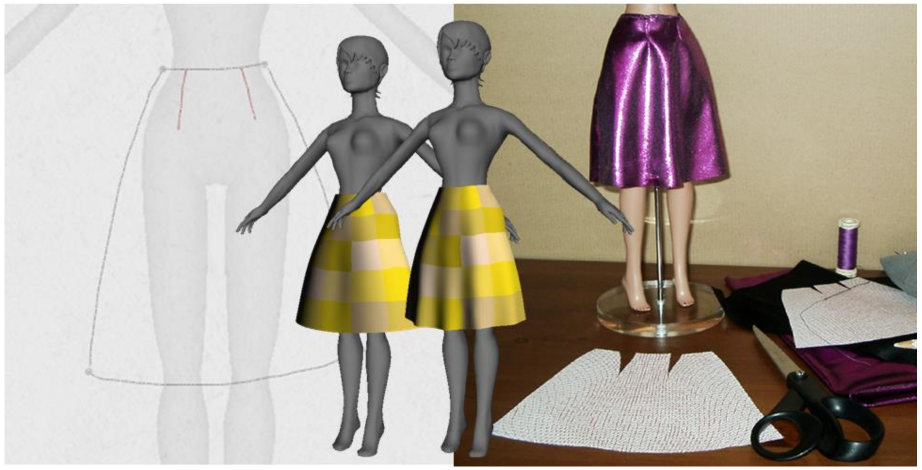


EUROGRAPHICS 2016

Clothing design

Developable surfaces from a sketch





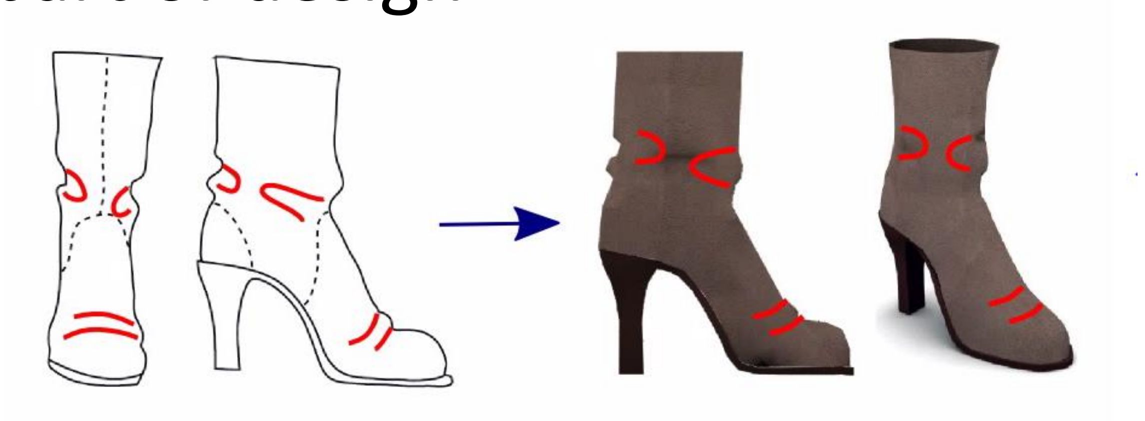
[Decaudin & al 2006]

[Julius et al 2007]

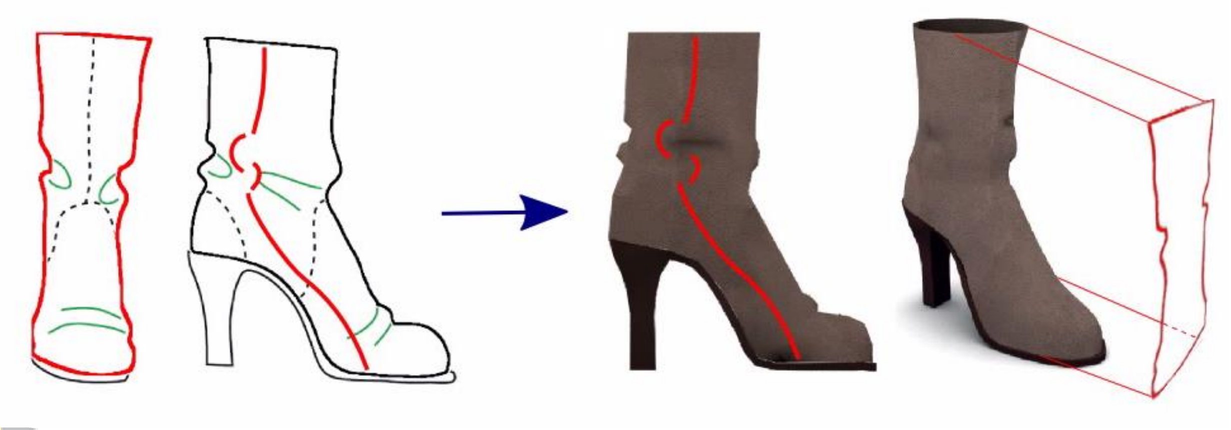
Clothing design

Sketching folds?

Folds are part of design



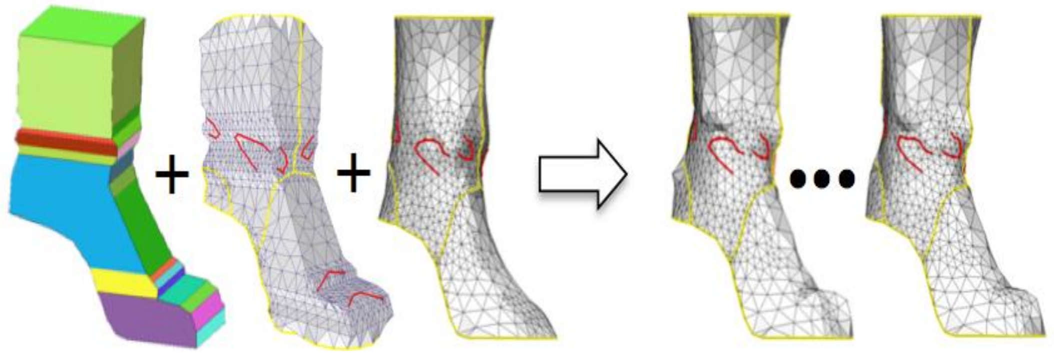
Challenge: Non-flat silhouettes !



EUROGRAPHICS 2016

Clothing design

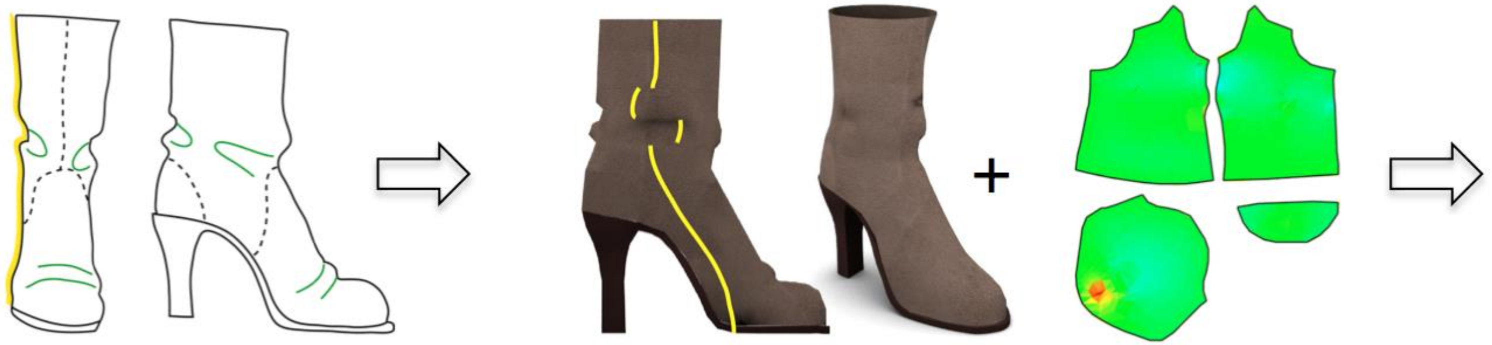
Sketching a folded surface



Iterate :

- Optimize developability
- Match the sketch

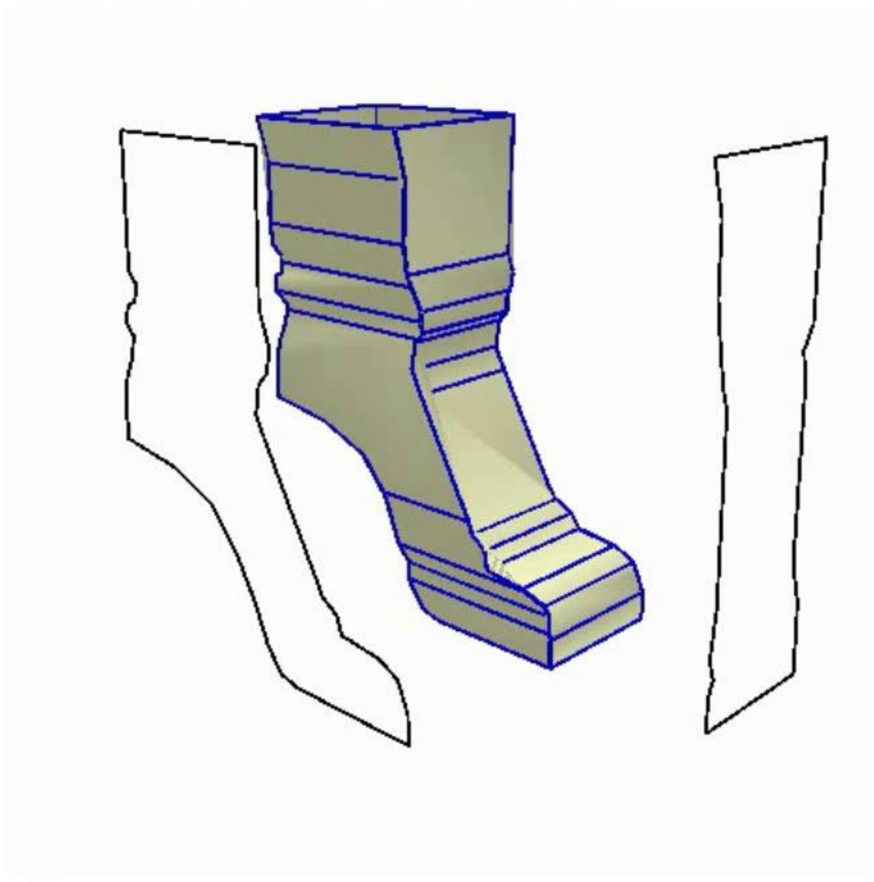
[Jung et al. TOG 2015]



EUROGRAPHICS 2016

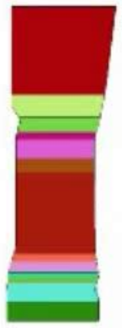
Initialization

Visual hull from silhouettes



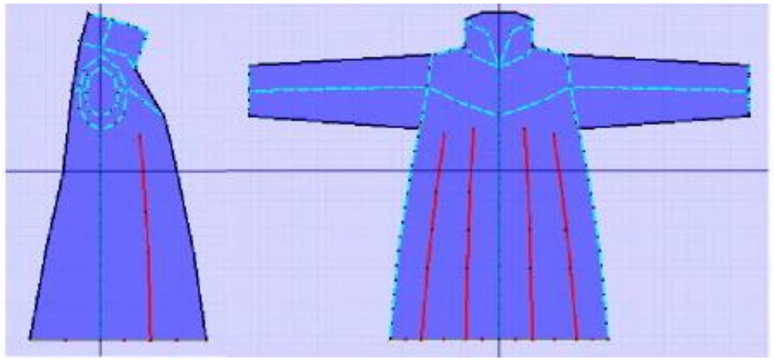
view y

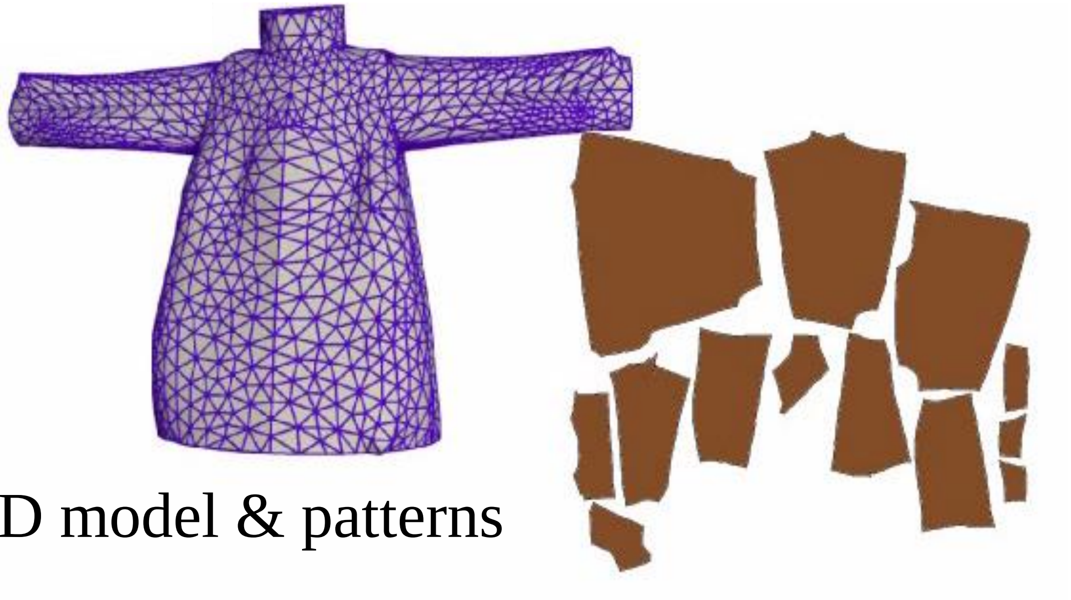
view x



Clothing design
Sketching a folded surface: Results

User input



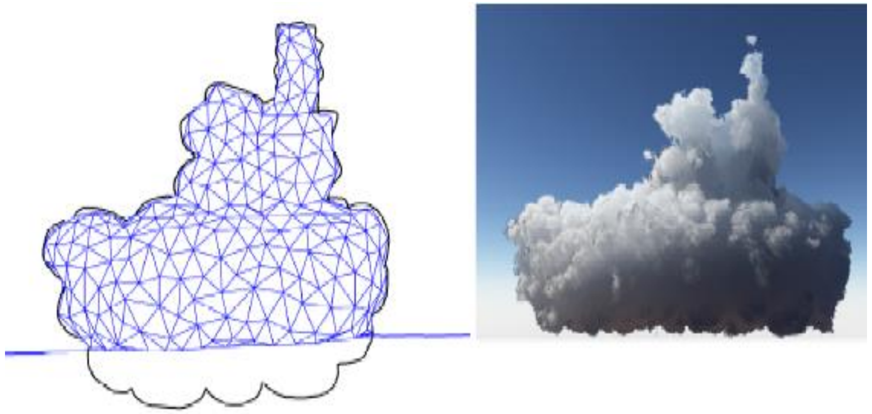


3D model & patterns

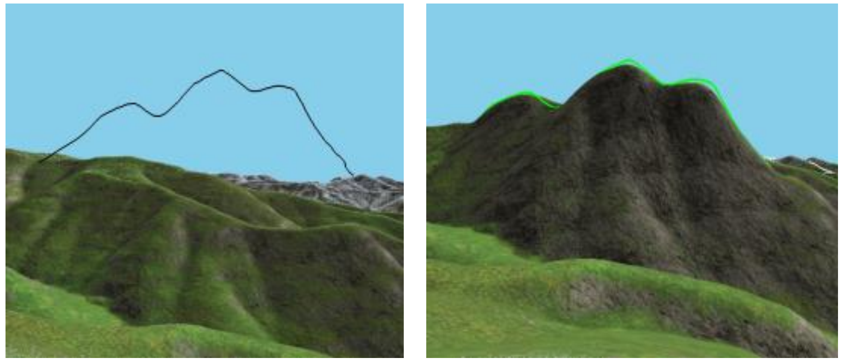
Sketch-based modelling using prior knowledge

Many other examples!

[Wither Bertails Cani 2007]



[Wither Bouthors Cani 2008]



[Tasse,Emilien, Cani, Hahmann, Dogson, GI'2014]

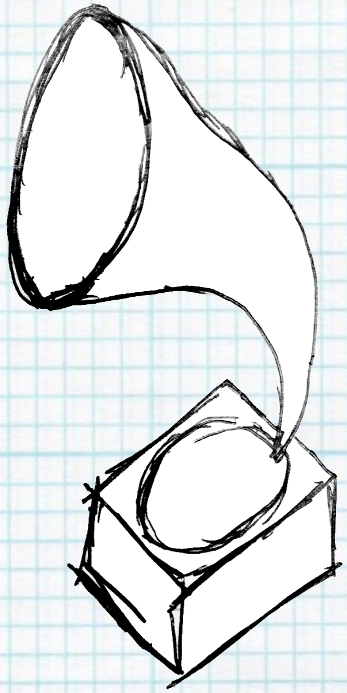
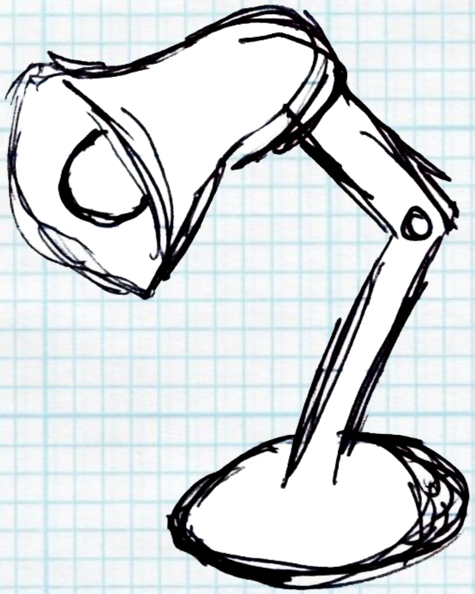
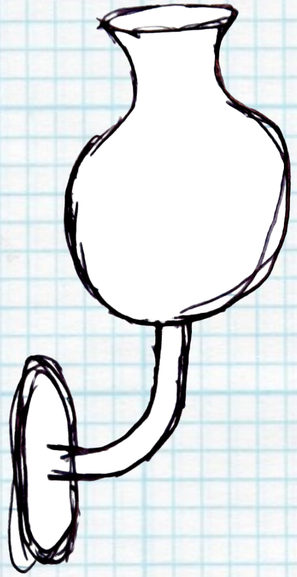
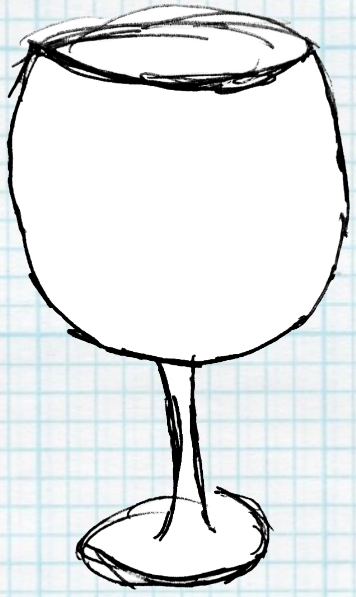
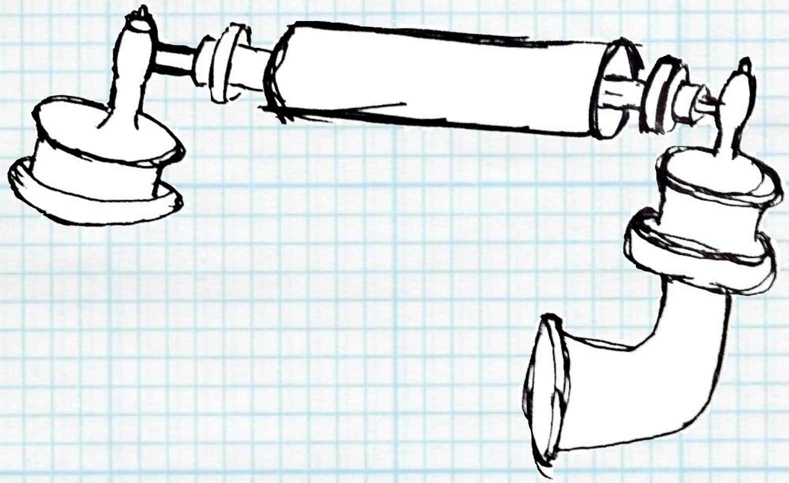
EUROGRAPHICS 2016

References

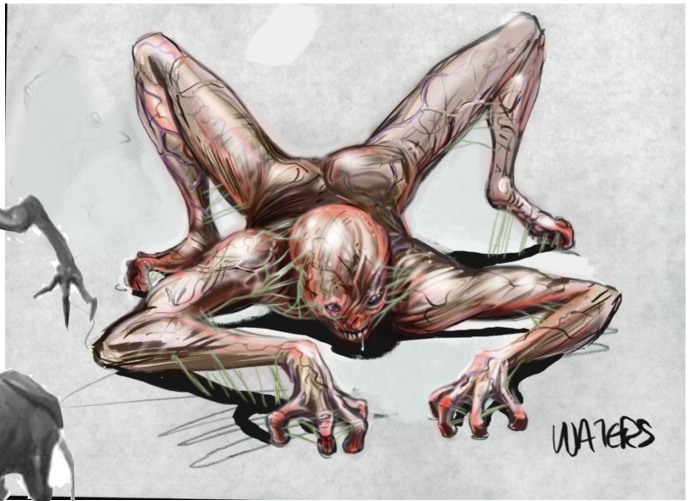
- ENTEM ET AL. - Modeling 3D animals from a side-view sketch
  Shape Modeling International (SMI), 2014
- TURQUIN, CANI, HUGHES - Sketching Garments for Virtual Characters
  Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBM), 2004
- DECAUDIN ET AL. - Virtual Garments: A Fully Geometric Approach for Clothing
  Computer Graphics Forum (CGF), 2006
- JULIUS, SHEFFER, CANI - Developable surfaces from arbitrary sketched boundaries
  Symposium on Geometry Processing (SGP), 2007
- JUNG, HAHMANN, ROHMER, CANI - Sketching Folds
  Transactions On Graphics (TOG), 2015
- WITHER, BOUTHORS, CANI - Rapid sketch modeling of clouds
  Eurographics *Workshop on Sketch-Based Interfaces and Modeling (SBM),* 2004
- WITHER, BERTAILS, CANI - Realistic Hair from a Sketch
  Shape Modeling International (SMI), 2007
- TASSE ET AL. - Feature-based terrain editing from complex sketches
  Computers and Graphics, 2014

Single-view sketch-based modeling of 3D curves and surfaces Part I

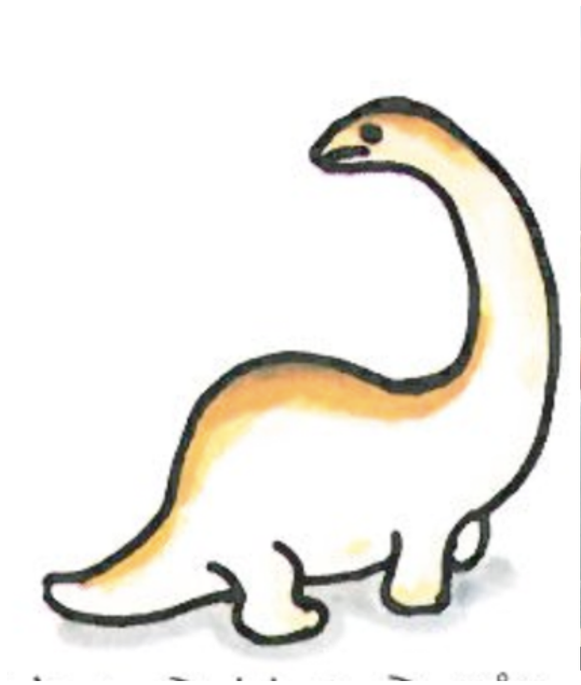
Yotam Gingold

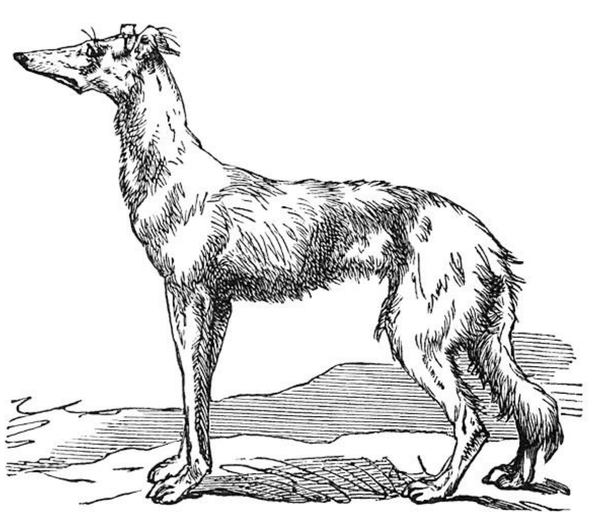






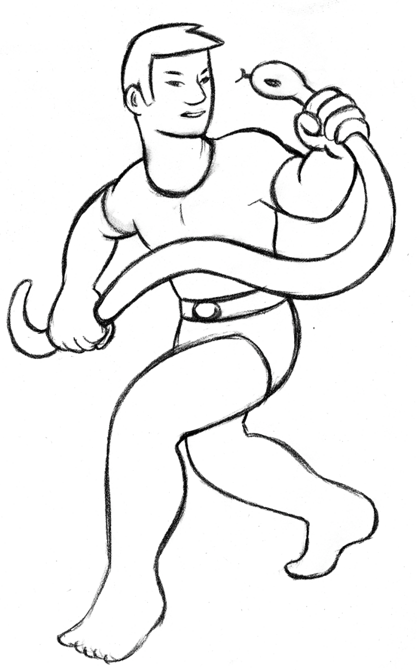
FEB. 11 2008

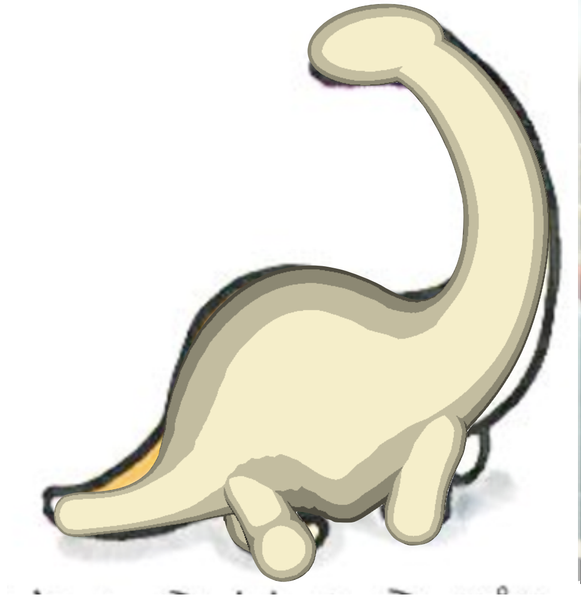

WATERS

©BradFitzpatrick.com



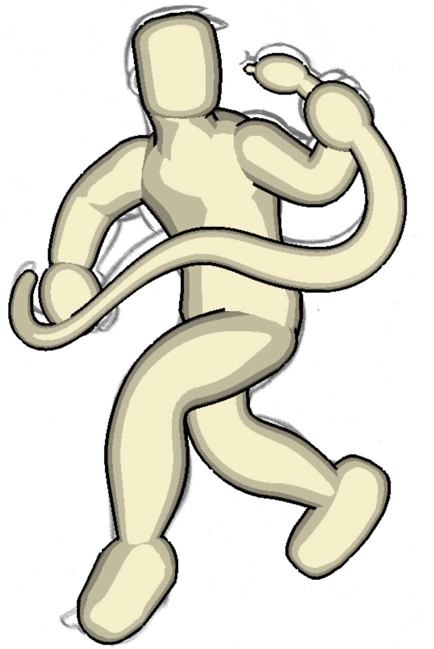
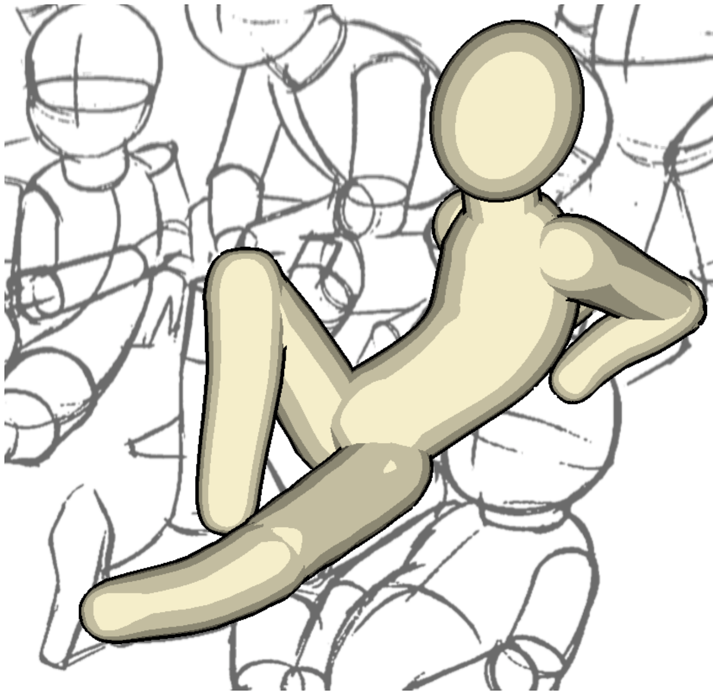

Lévrier persan.





CGSociety.org

© 2009 Chris Onstad :: achewood.com
SALE
9-4

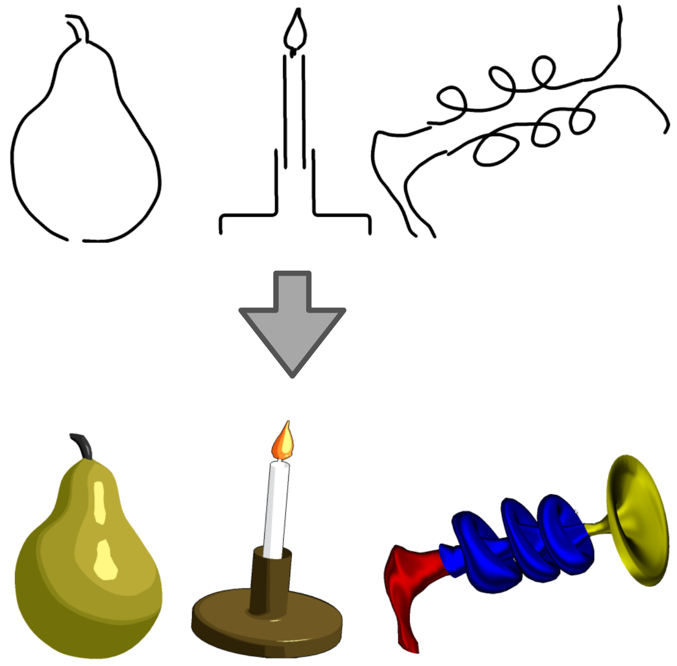
Lévrier persan.
oldbookillustrations.com



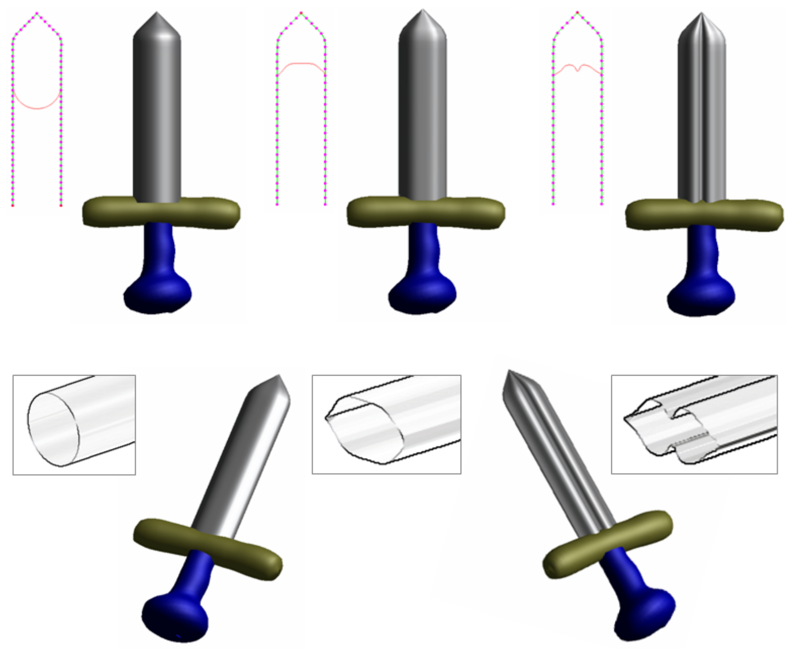
Goals

- Model by “describing” an existing 2D image with primitives and annotations.
- Usable by novices, including those with poor drawing skills.
- Create structured models.

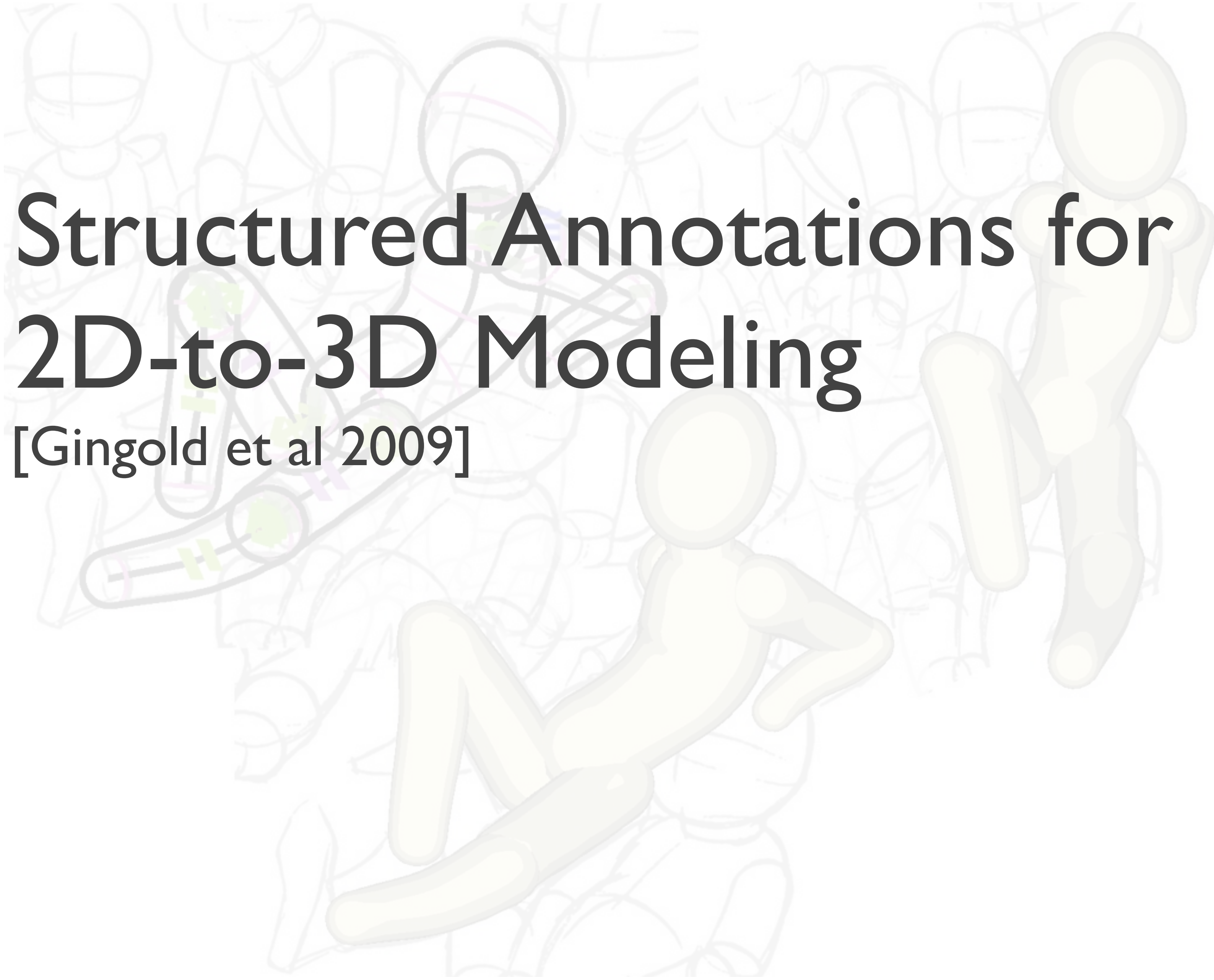
Sketch-based modeling with few strokes [Cherlin et al. 2005]

- Generalized cylinders with varying cross sections and “spines”





EUROGRAPHICS 2016

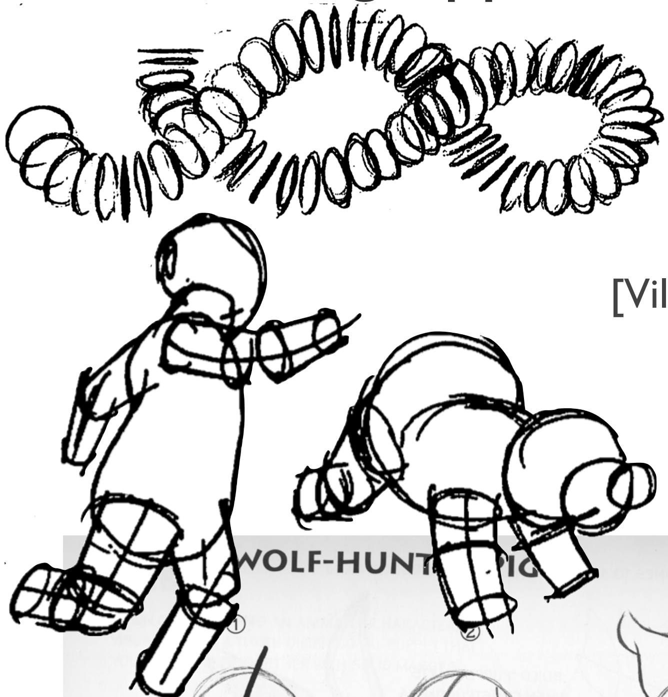


Structured Annotations for 2D-to-3D Modeling

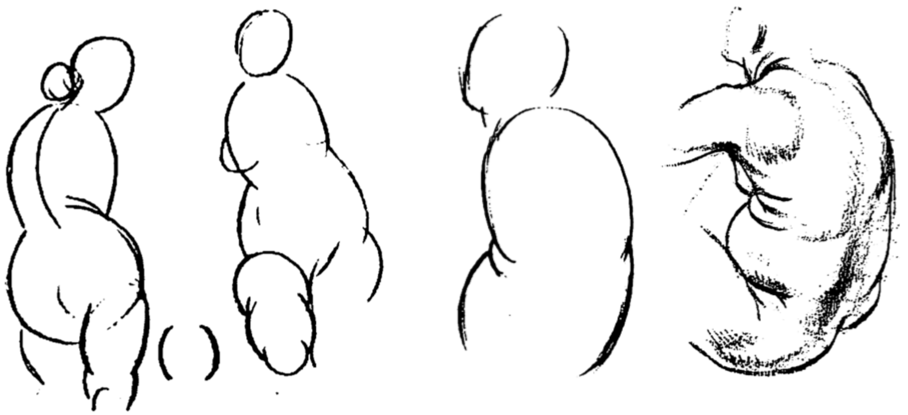
[Gingold et al 2009]

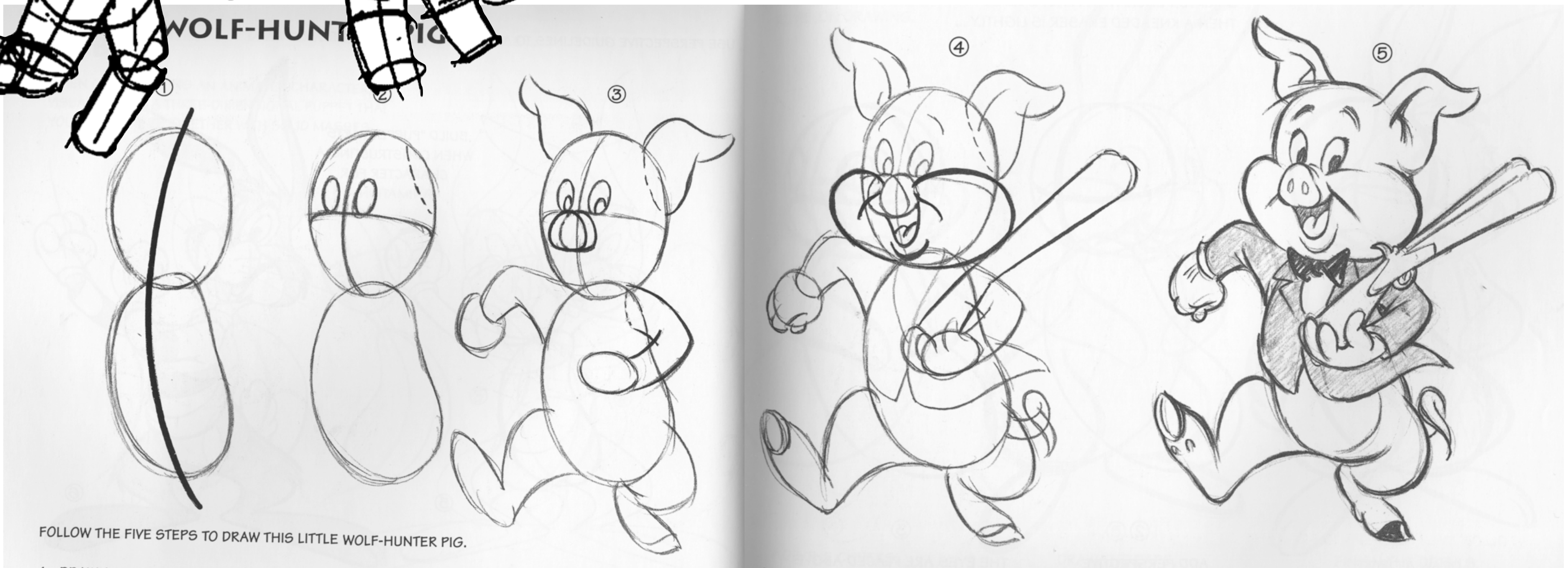
Inspiration

2D Drawing Approaches



[Vilppu 1997]

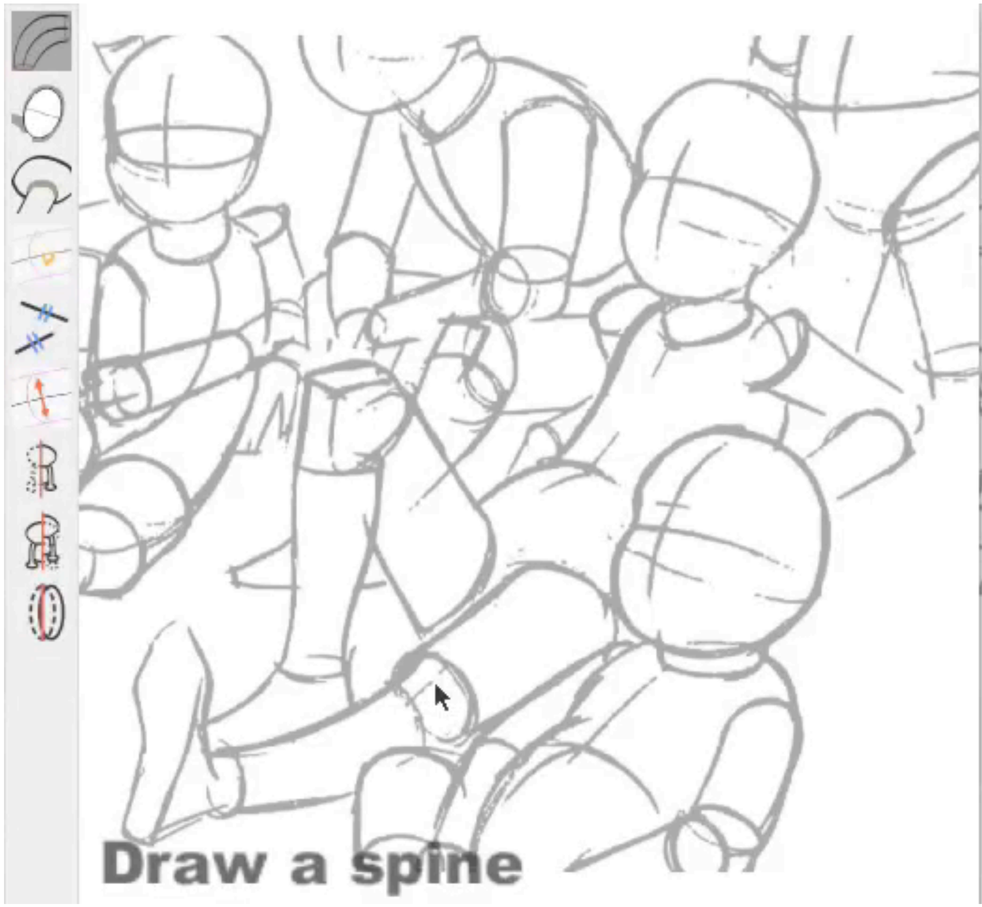




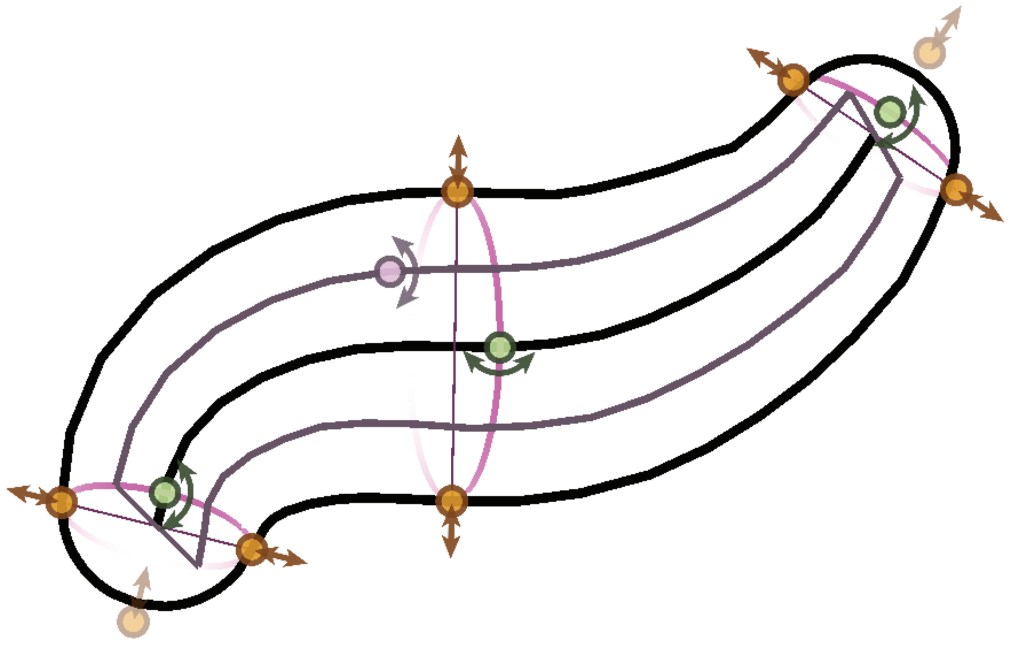
[Blair 1994]

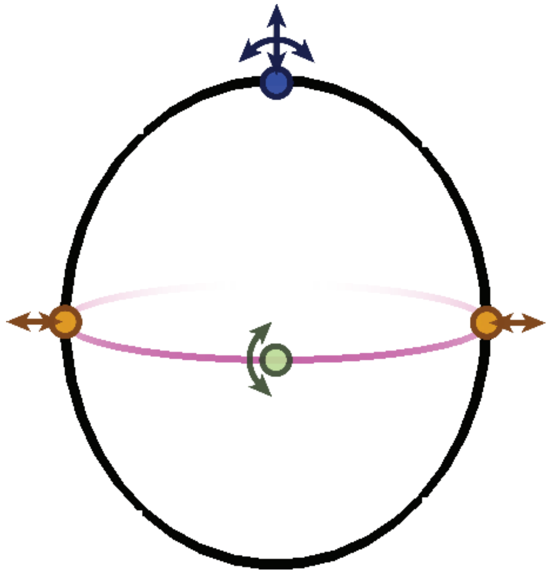
Primitives

Generalized Cylinders & Ellipsoids



Primitives


Generalized Cylinder


Ellipsoid

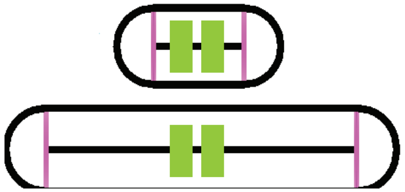
Primitive: Generalized Cylinder

Generalized Cylinder

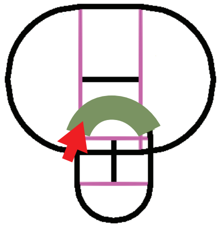
Primitive: Ellipsoid

Annotations

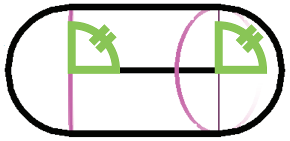
Same-length



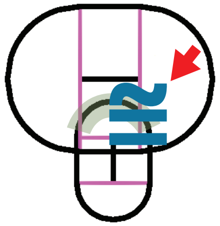
Connection curve



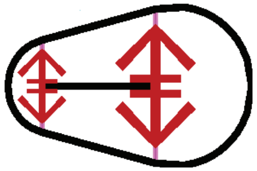
Same-tilt



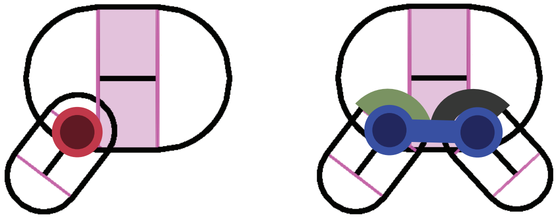
Mirror



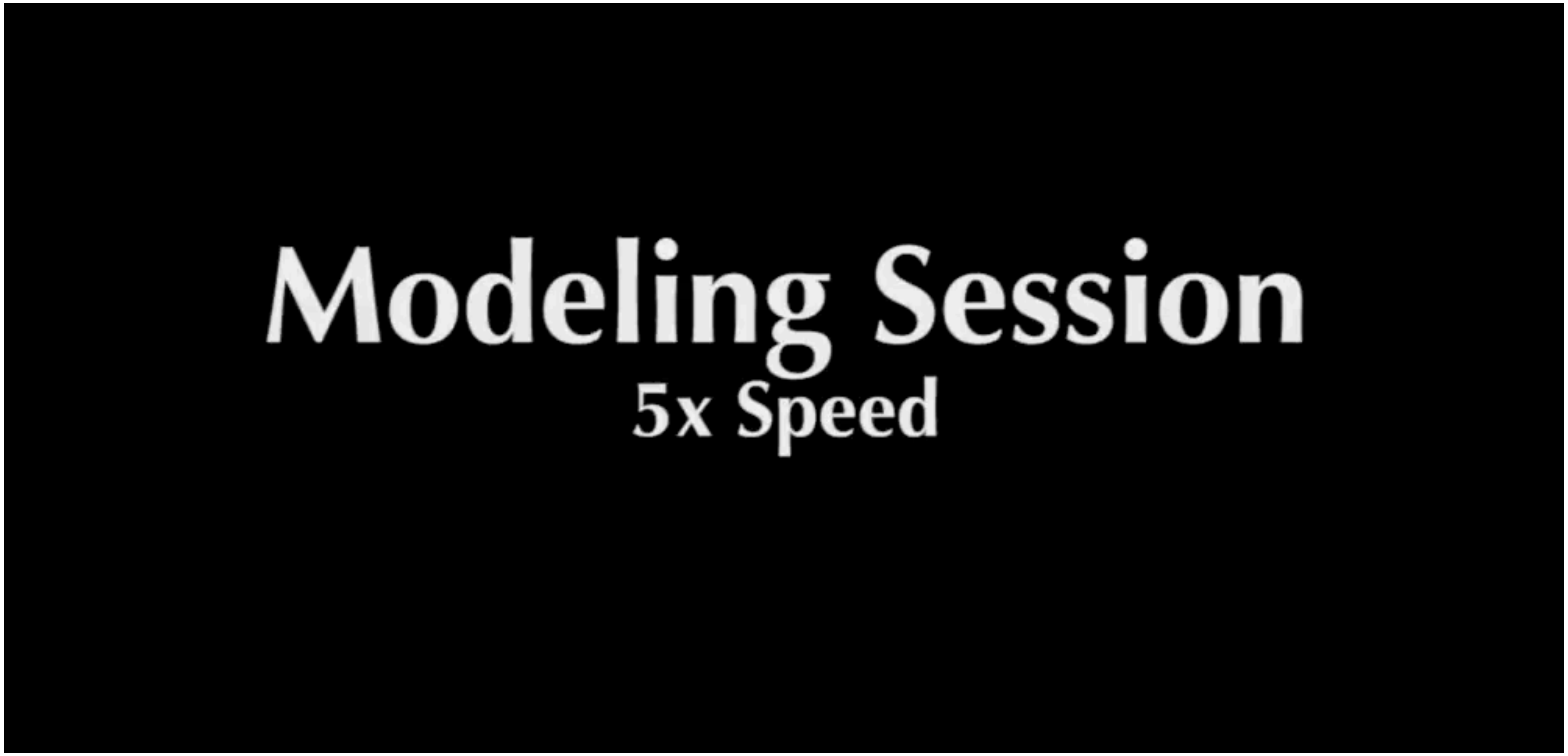
Same-scale



Alignment

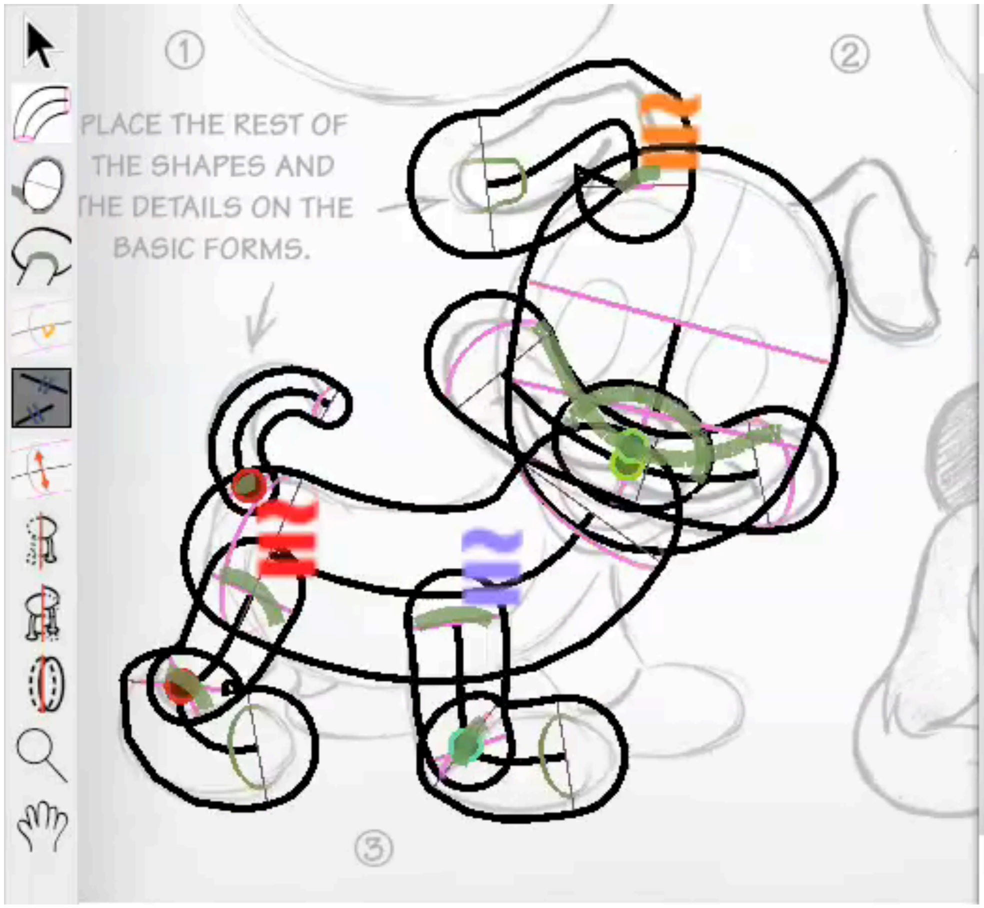


Demo

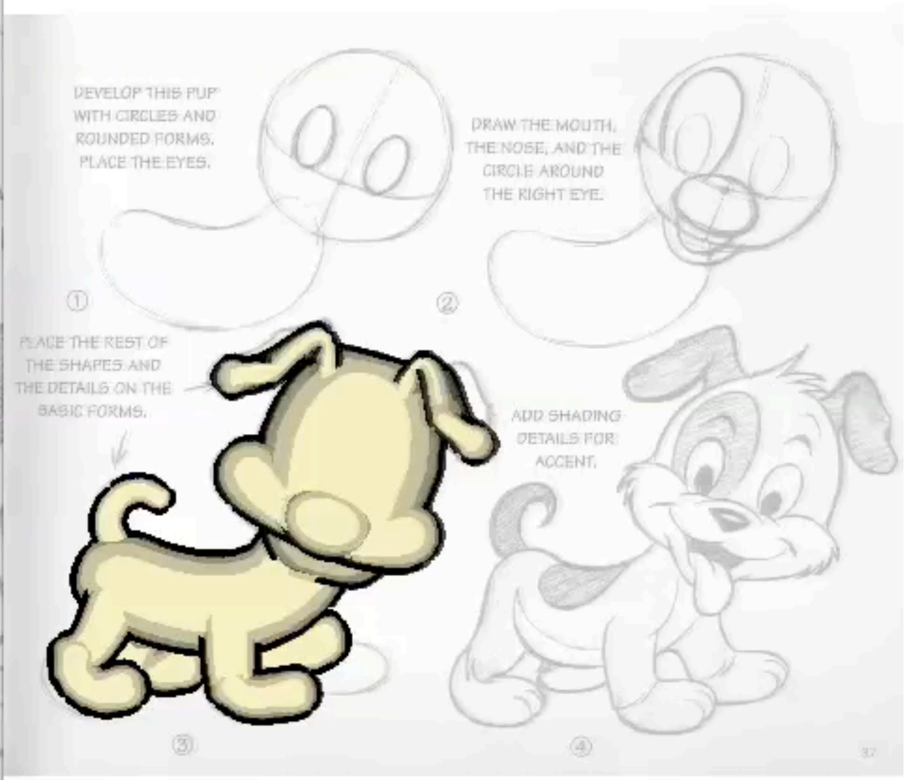


Guide image [Vilppu 1997]

Results



①
②
③
PLACE THE REST OF THE SHAPES AND THE DETAILS ON THE BASIC FORMS.



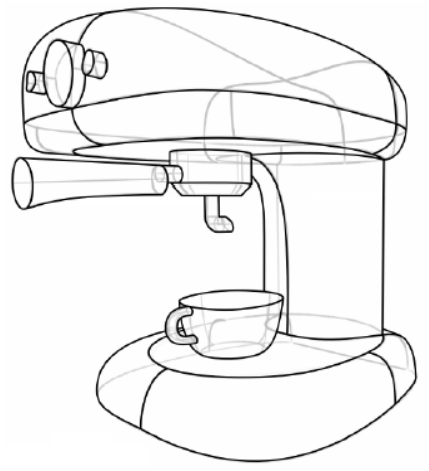
DEVELOP THIS PUP WITH CIRCLES AND ROUNDED FORMS. PLACE THE EYES.
DRAW THE MOUTH, THE NOSE, AND THE CIRCLE AROUND THE RIGHT EYE.
PLACE THE REST OF THE SHAPES AND THE DETAILS ON THE BASIC FORMS.
ADD SHADING DETAILS FOR ACCENT.
①
②
③
④

Guide images: [Blair 1994]; © Alex Rosmarin; © Kei Acedera, Imaginism Studios 2008; © Björn Hurri, www.bjornhurri.com; © Alex Rosmarin; © Alex Rosmarin; [Kako 1973]; [Kako 1973]

Limitations

Limitations

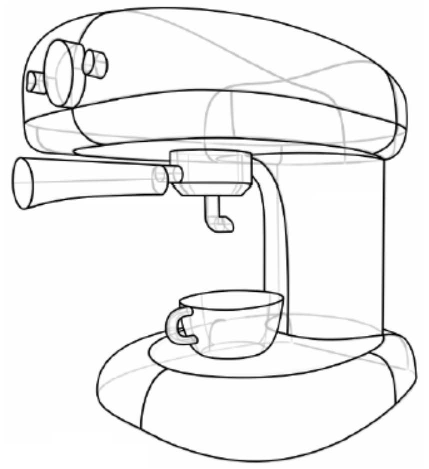
- Limited range of models

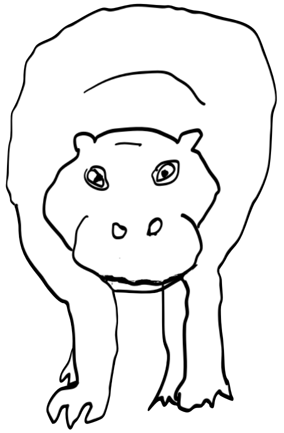


[Schmidt et al. 2009b]

Limitations

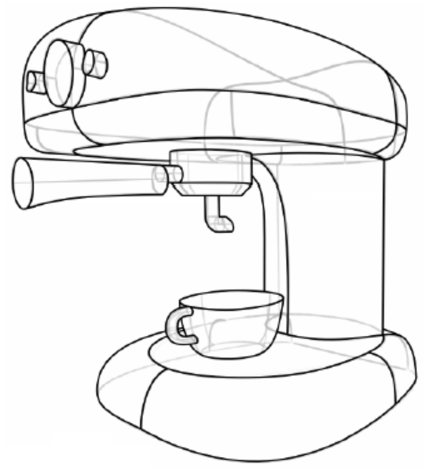
- Limited range of models
- Can't be used for certain drawings

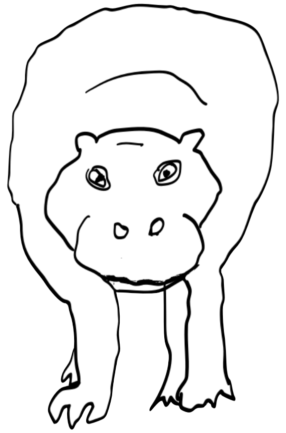

[Schmidt et al. 2009b]



Limitations

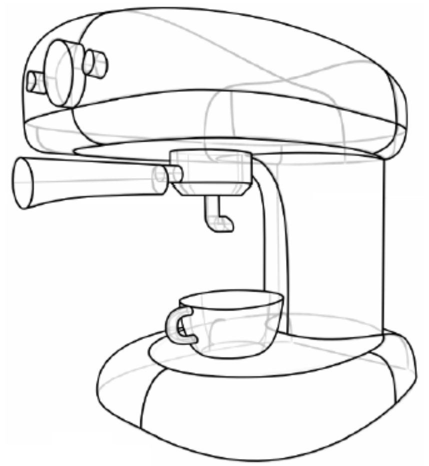
- Limited range of models
- Can’t be used for certain drawings
- No cycles of connection curves

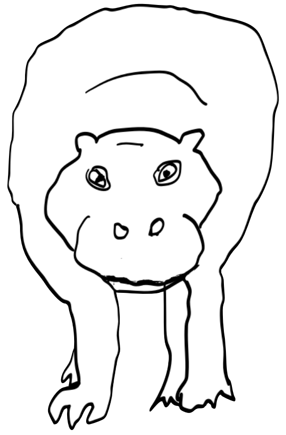

[Schmidt et al. 2009b]

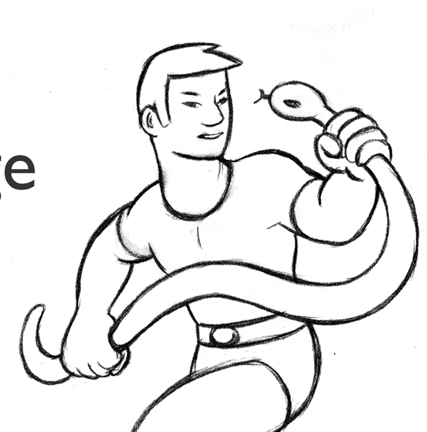


Limitations

- Limited range of models
- Can't be used for certain drawings
- No cycles of connection curves
- Doesn't actually use the guide image

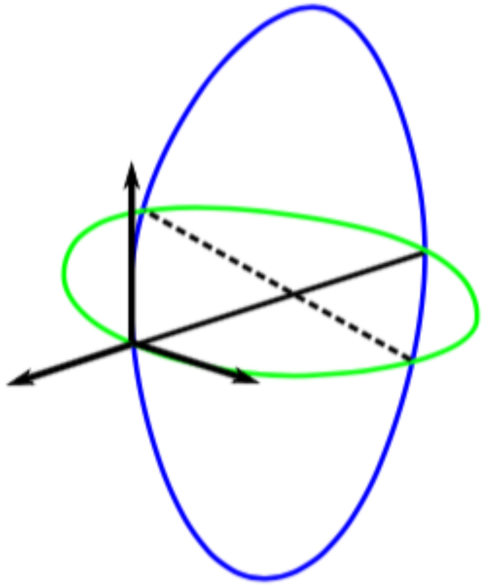

[Schmidt et al. 2009b]





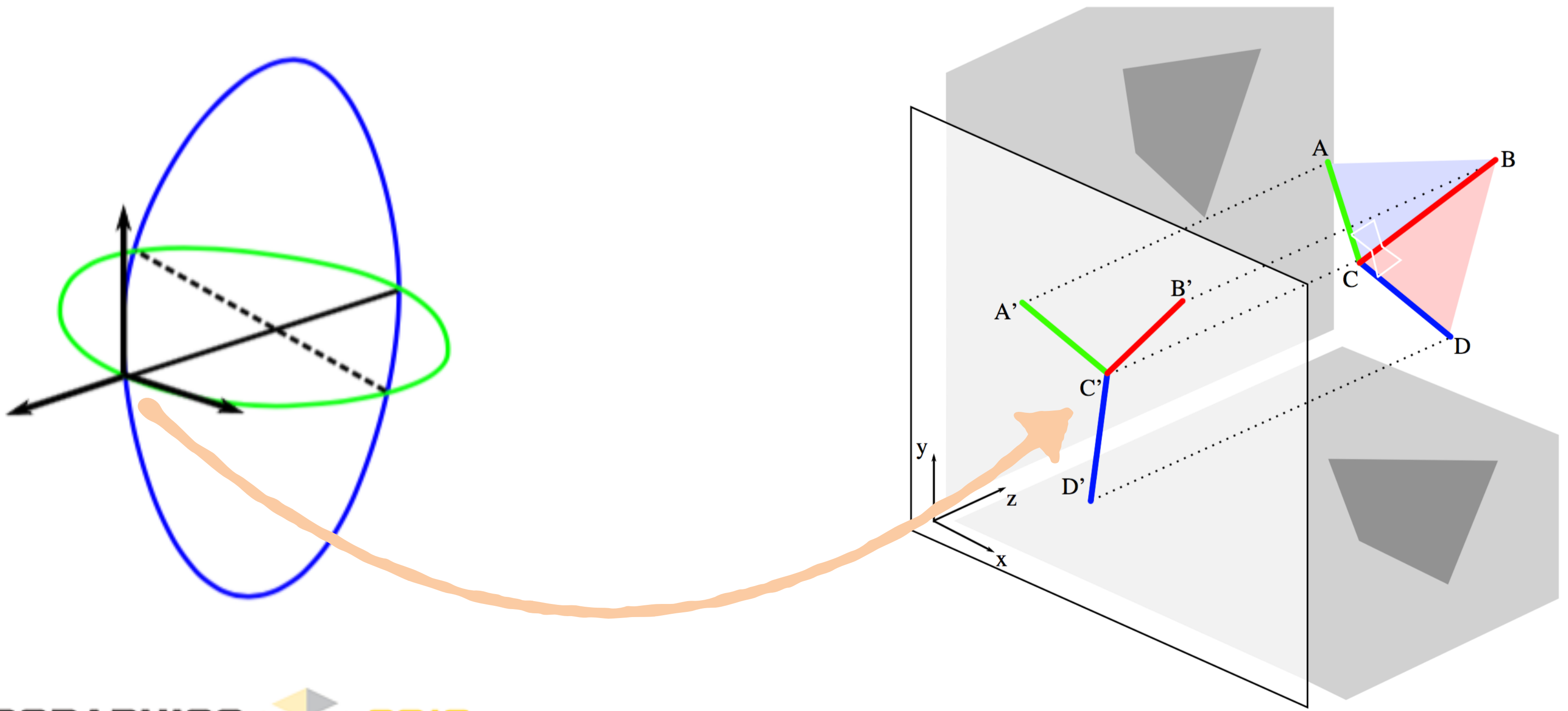
Single-View Sketch-Based Modeling [Andre and Saito 2011]

- Two perpendicular cross sections form the projection of a cubic corner (which is well-defined)



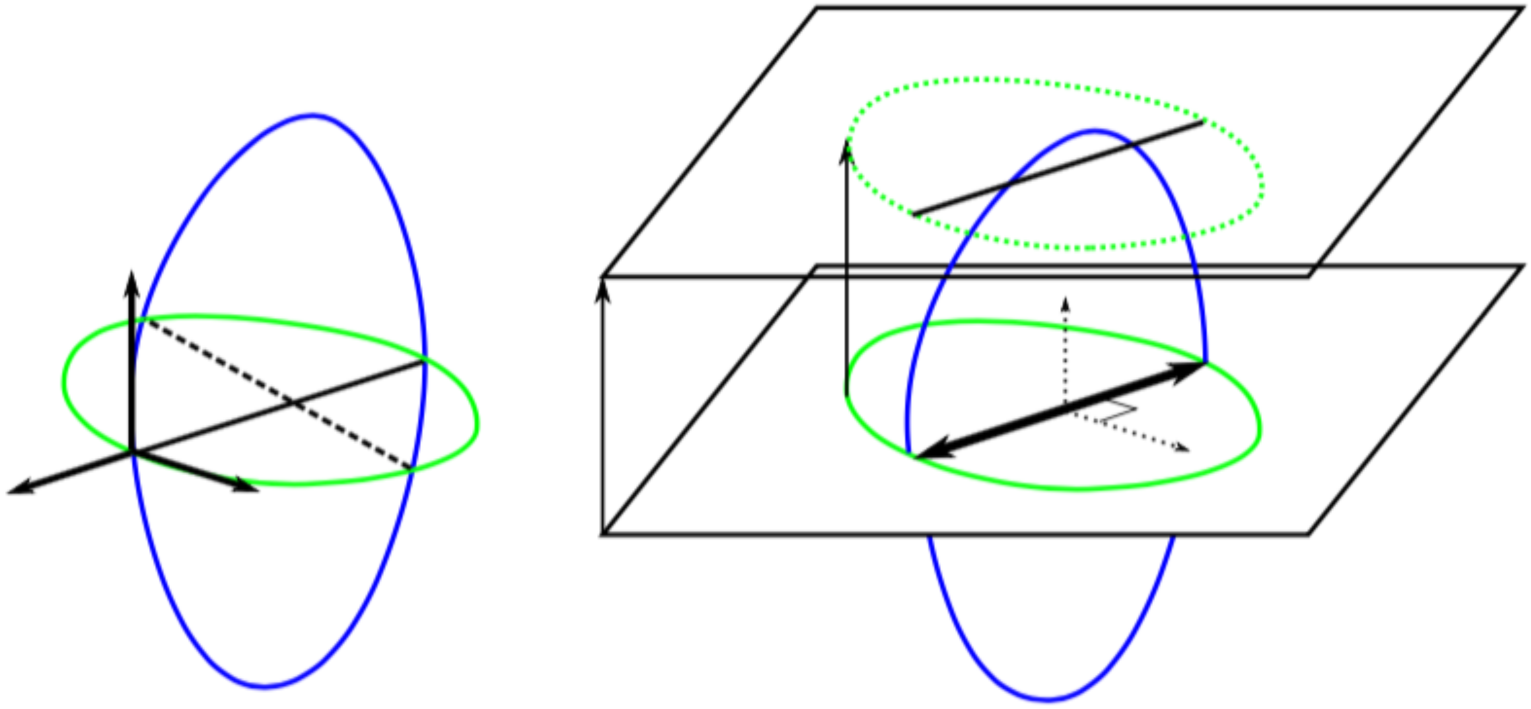
Single-View Sketch-Based Modeling [Andre and Saito 2011]

- Two perpendicular cross sections form the projection of a cubic corner (which is well-defined)



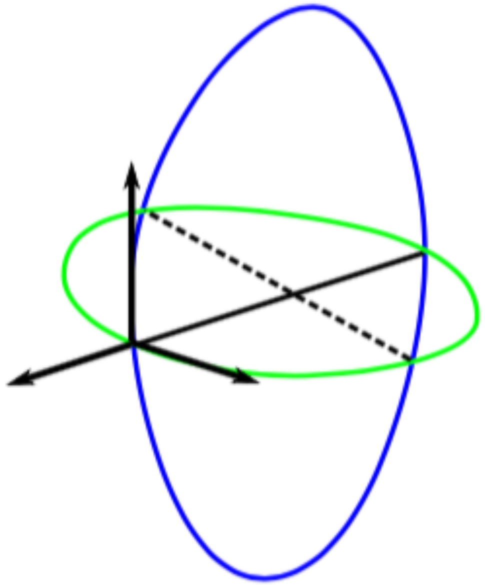
Single-View Sketch-Based Modeling [Andre and Saito 2011]

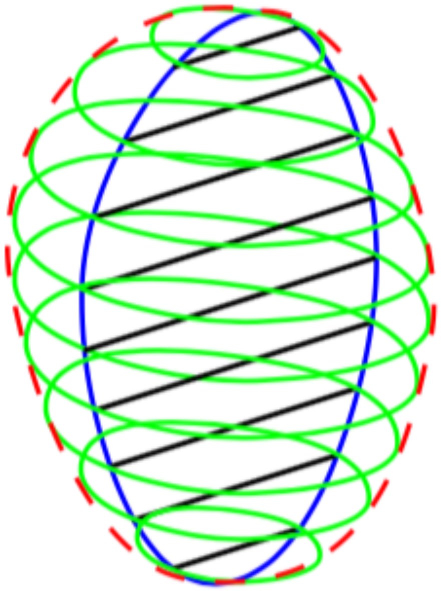
- That gives us 3D axes for the shape



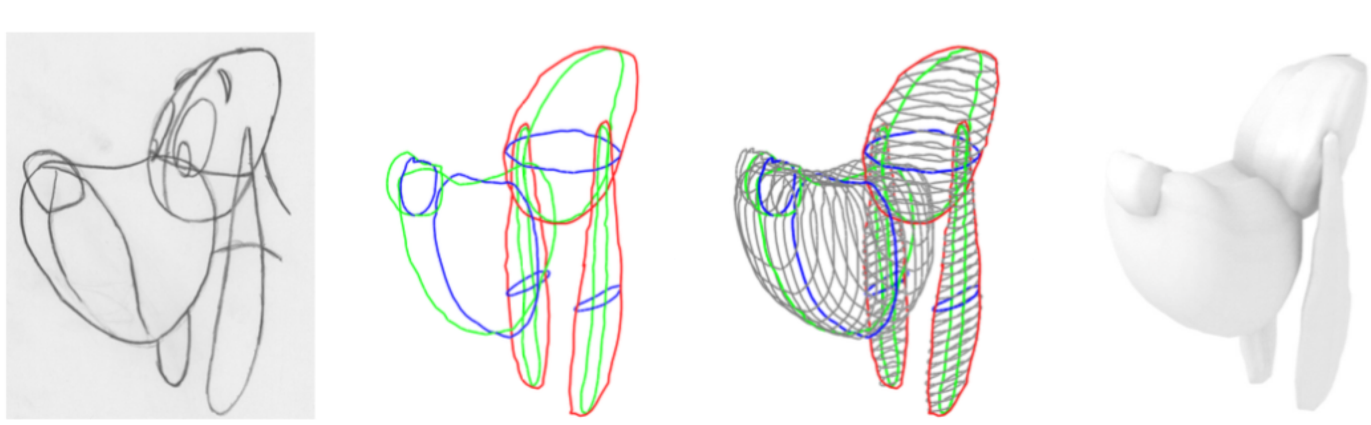
Single-View Sketch-Based Modeling [Andre and Saito 2011]

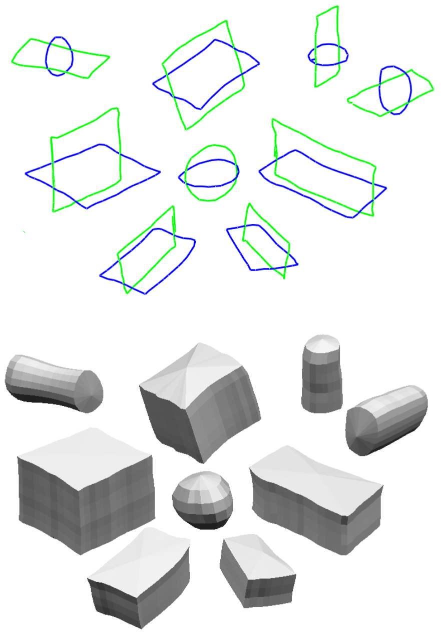
- Which we can use to sweep out a surface

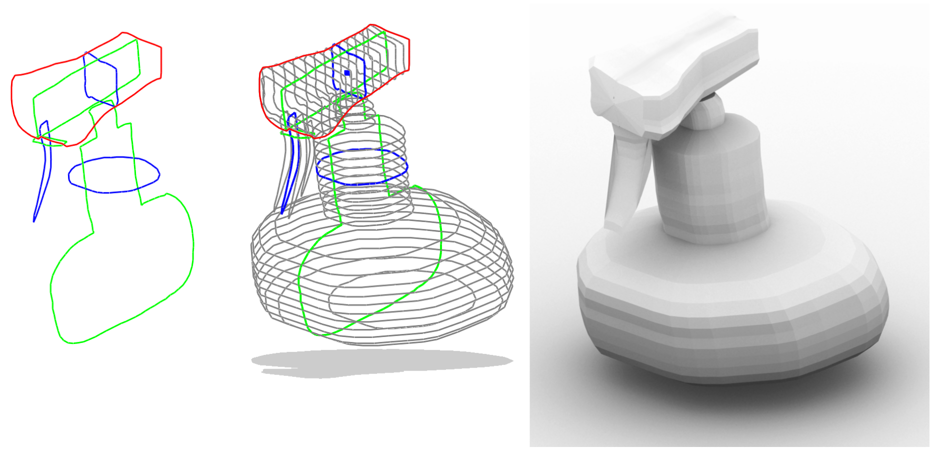


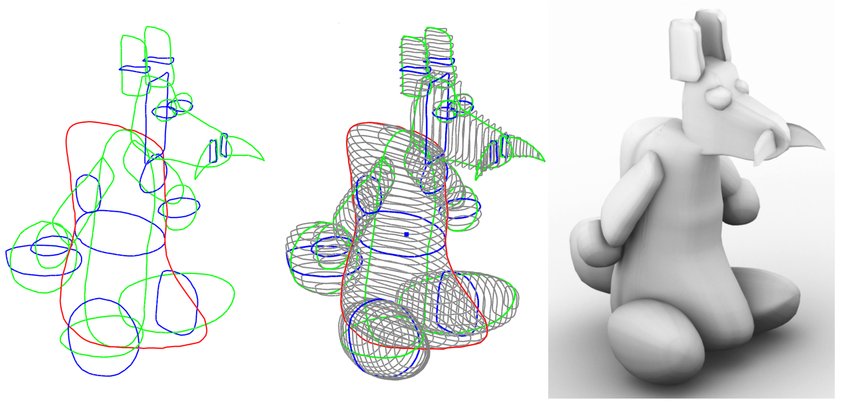


Single-View Sketch-Based Modeling [Andre and Saito 2011]



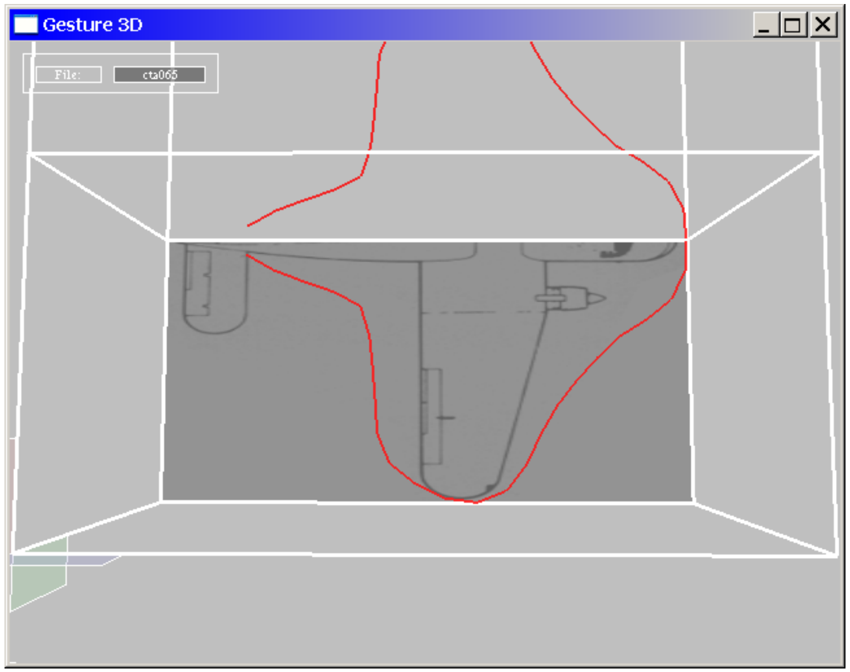




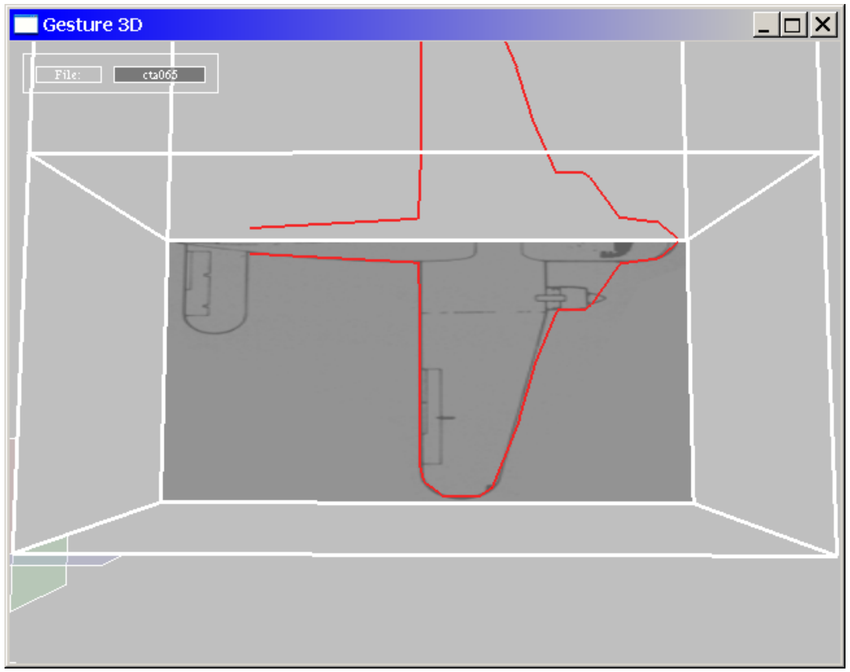


A suggestive interface for image guided 3D sketching [Tsang et al. 2004]

- Use the guide sketch to snap strokes.



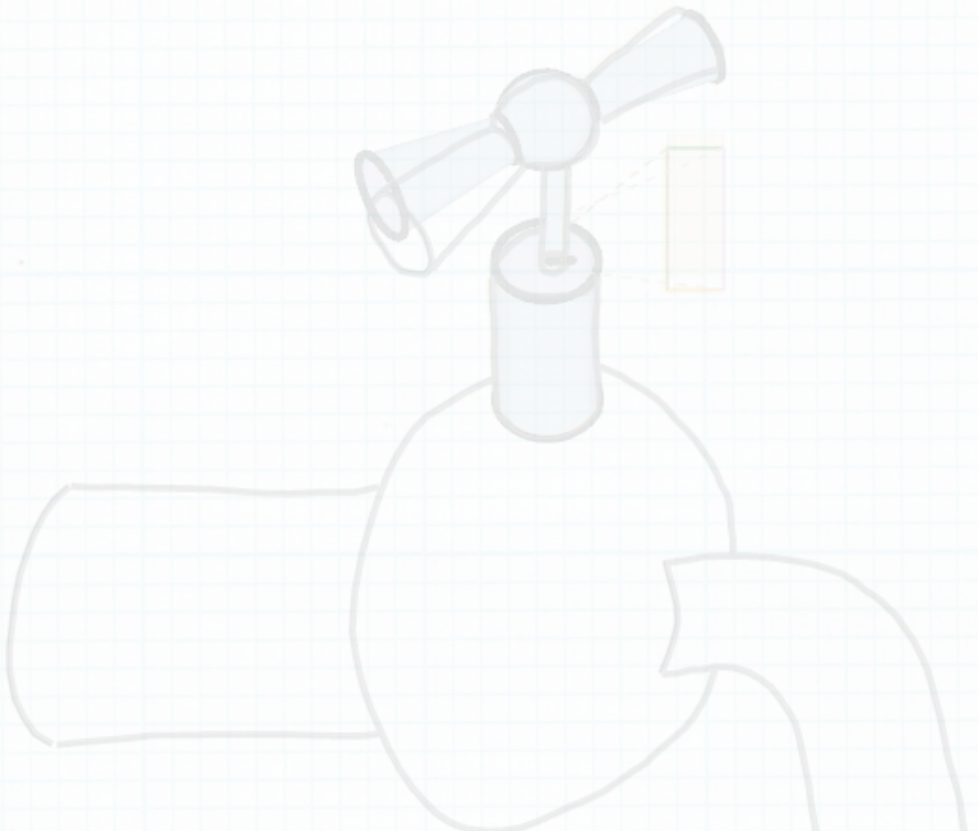
User sketch



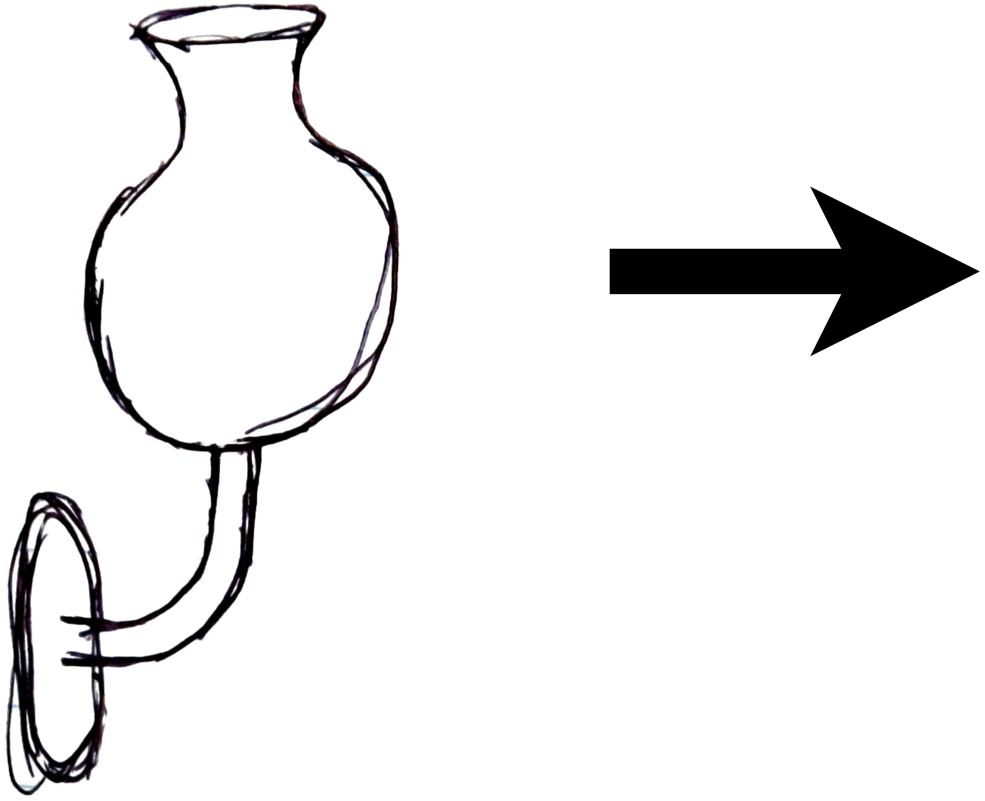
Automatically snapped to the guide image

Geosemantic Snapping for Sketch-Based Modeling

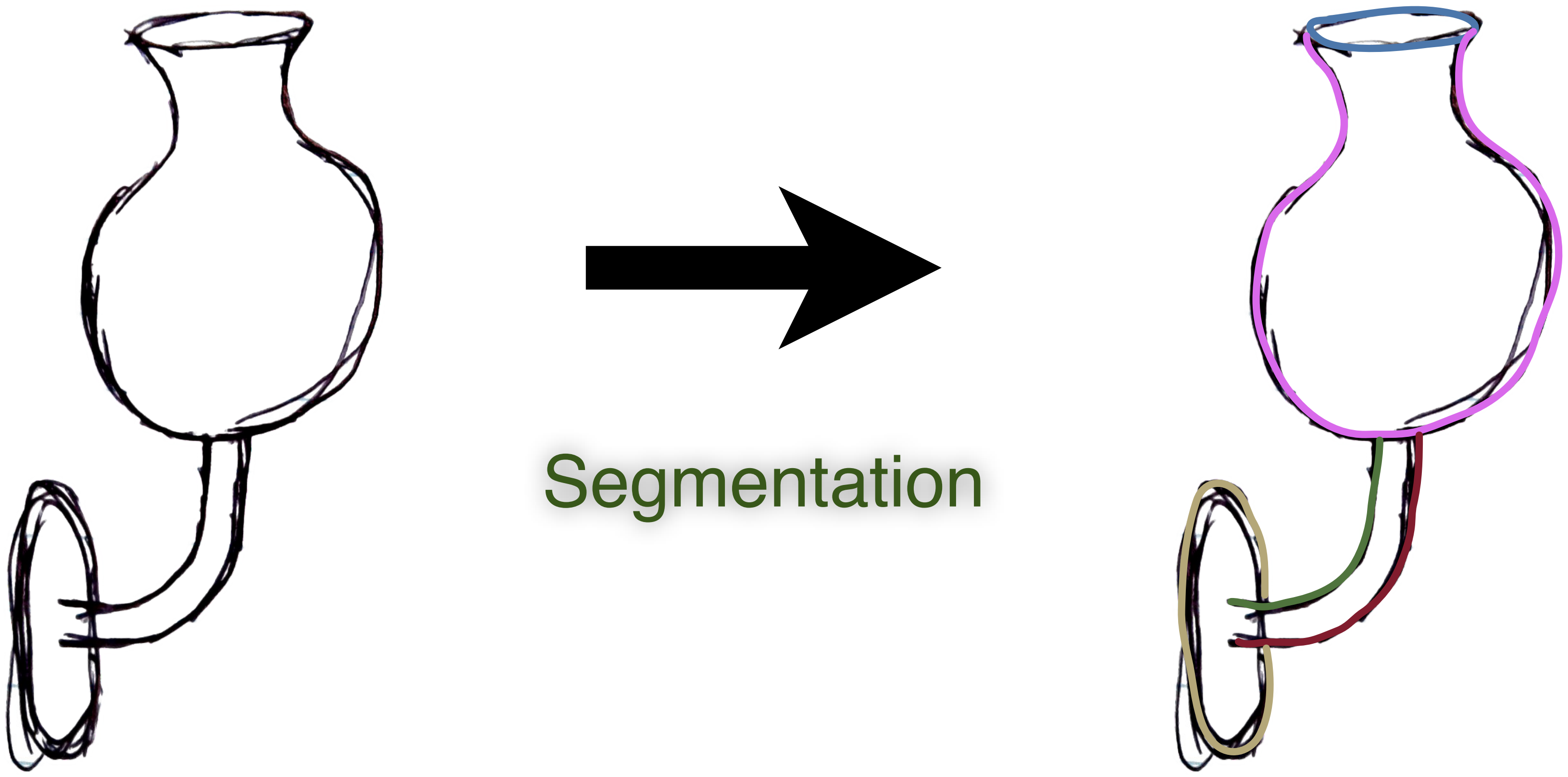
[Shtof et al. 2013]



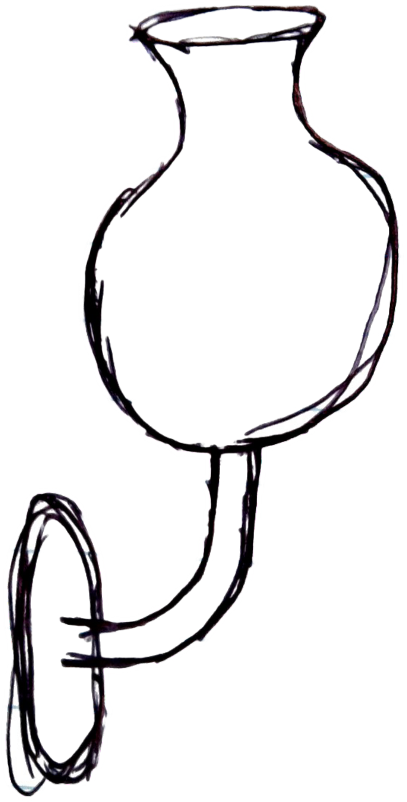
Challenges



Challenges

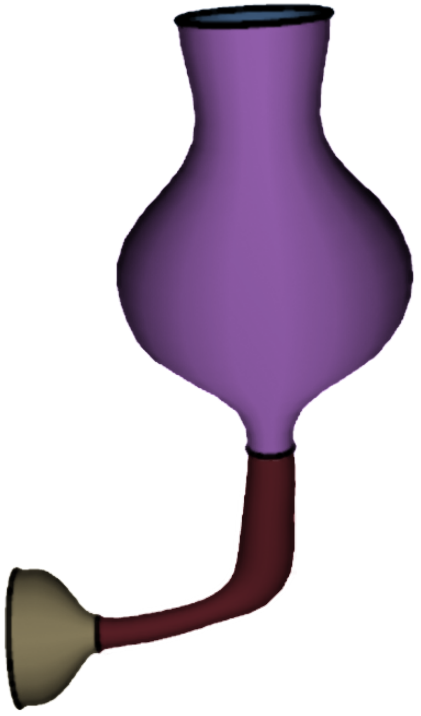


Challenges

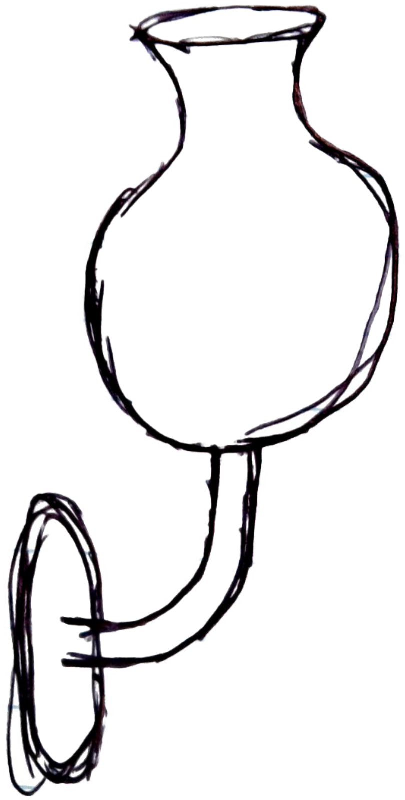


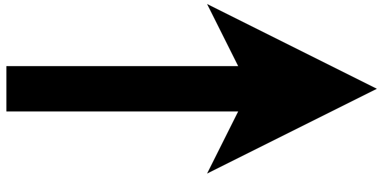
Segmentation

Recognition

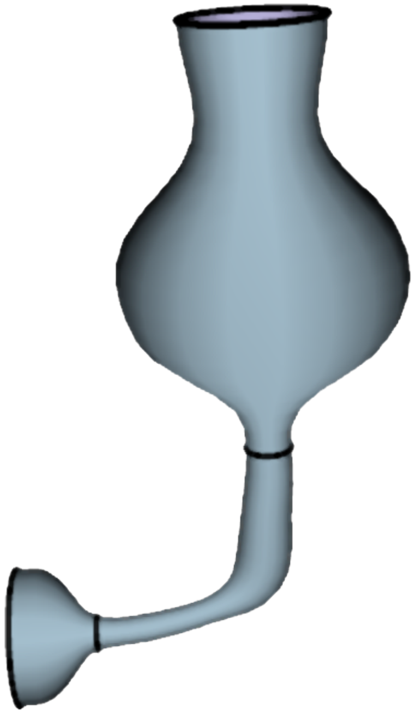


Challenges

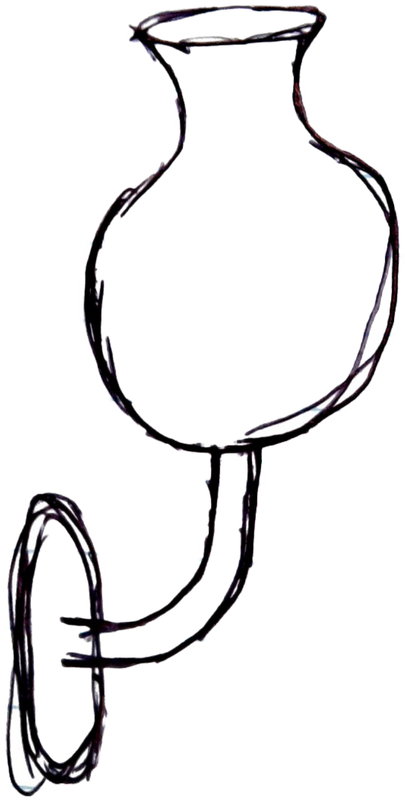


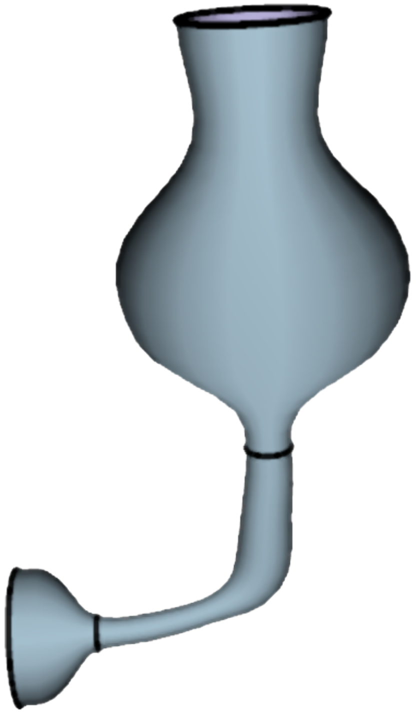


Segmentation
Recognition
Positioning



Challenges

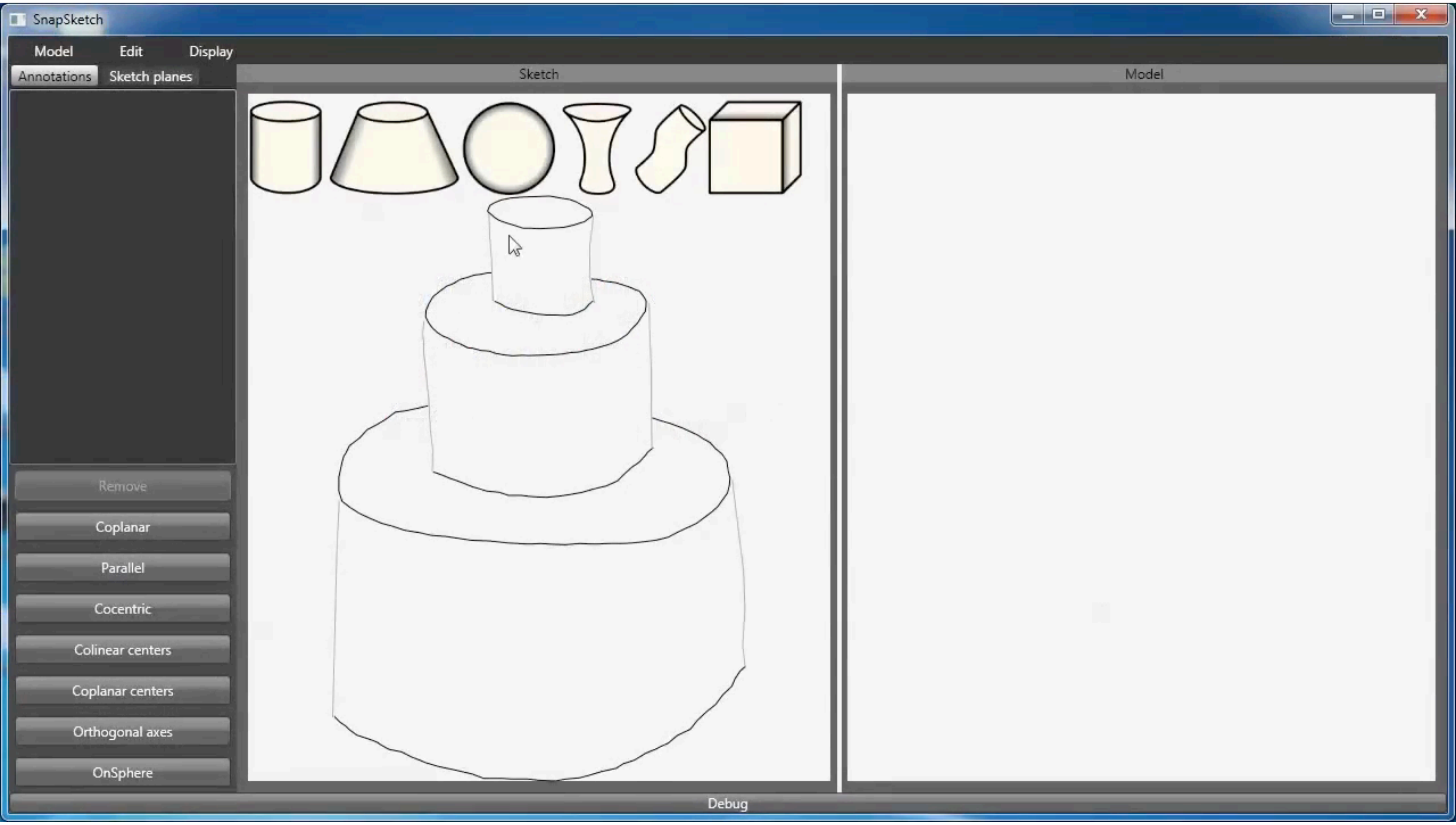




Segmentation
Recognition
Positioning

An automatic solution entails solving a **complex, non-convex** optimization problem with **many local minima**.

Interactive Approach

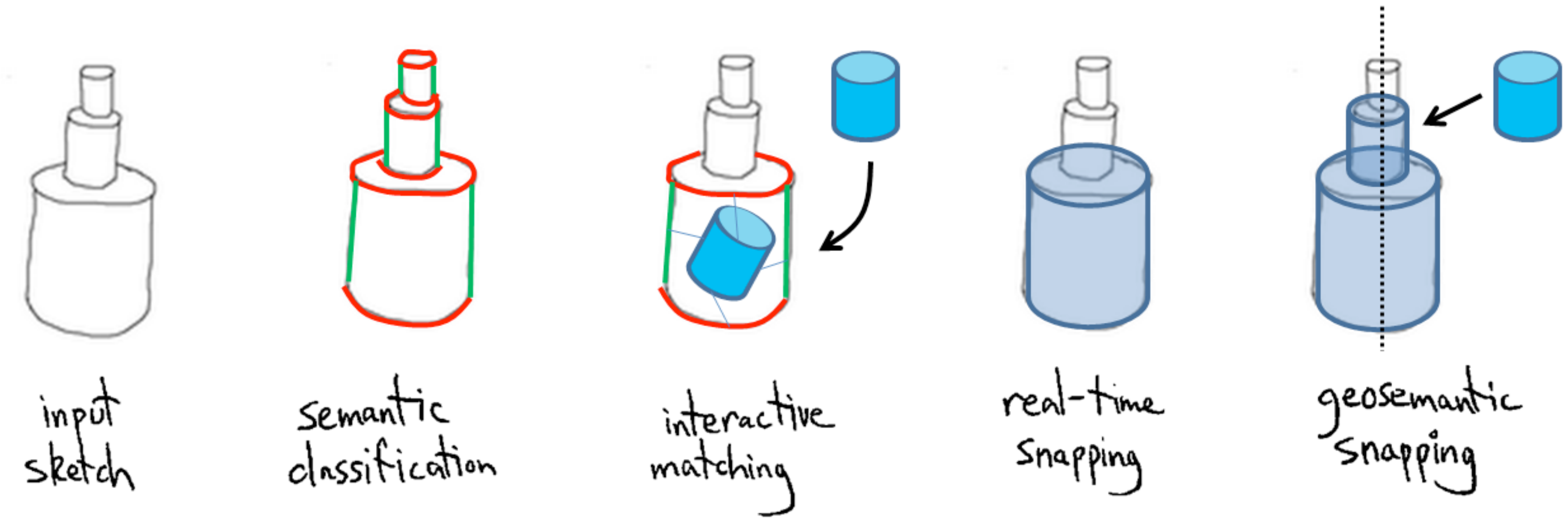


Separate the problem into **semantic** and **geometric** tasks

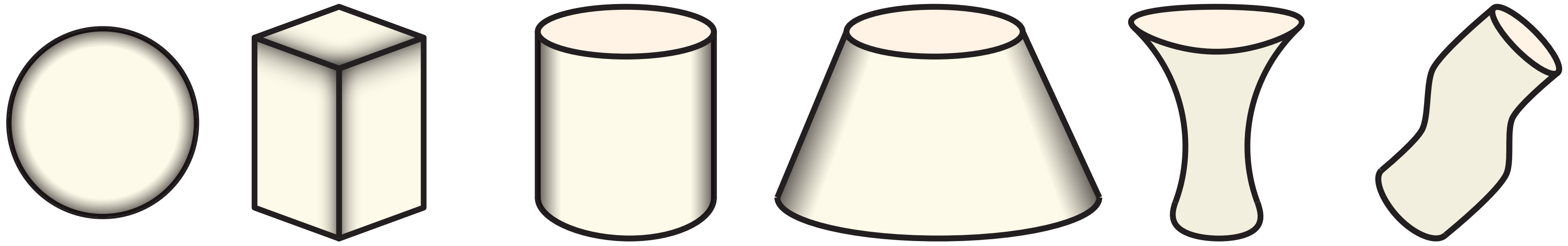
semantic: interpreting the sketch’s individual strokes and parts

geometric: fitting and reconstructing precise geometry

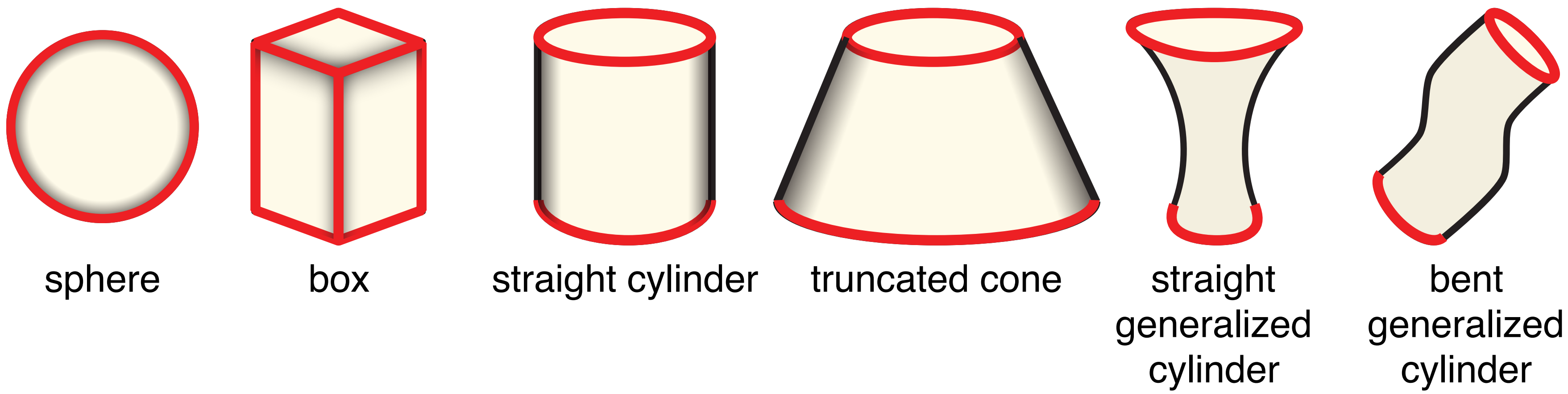
Overview


input sketch
semantic classification
interactive matching
real-time snapping
geosemantic snapping

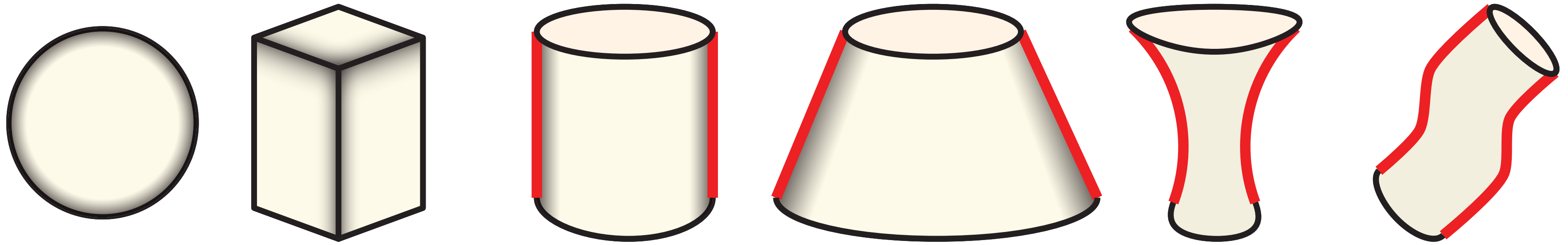
Primitives


sphere
box
straight cylinder
truncated cone
straight generalized cylinder
bent generalized cylinder

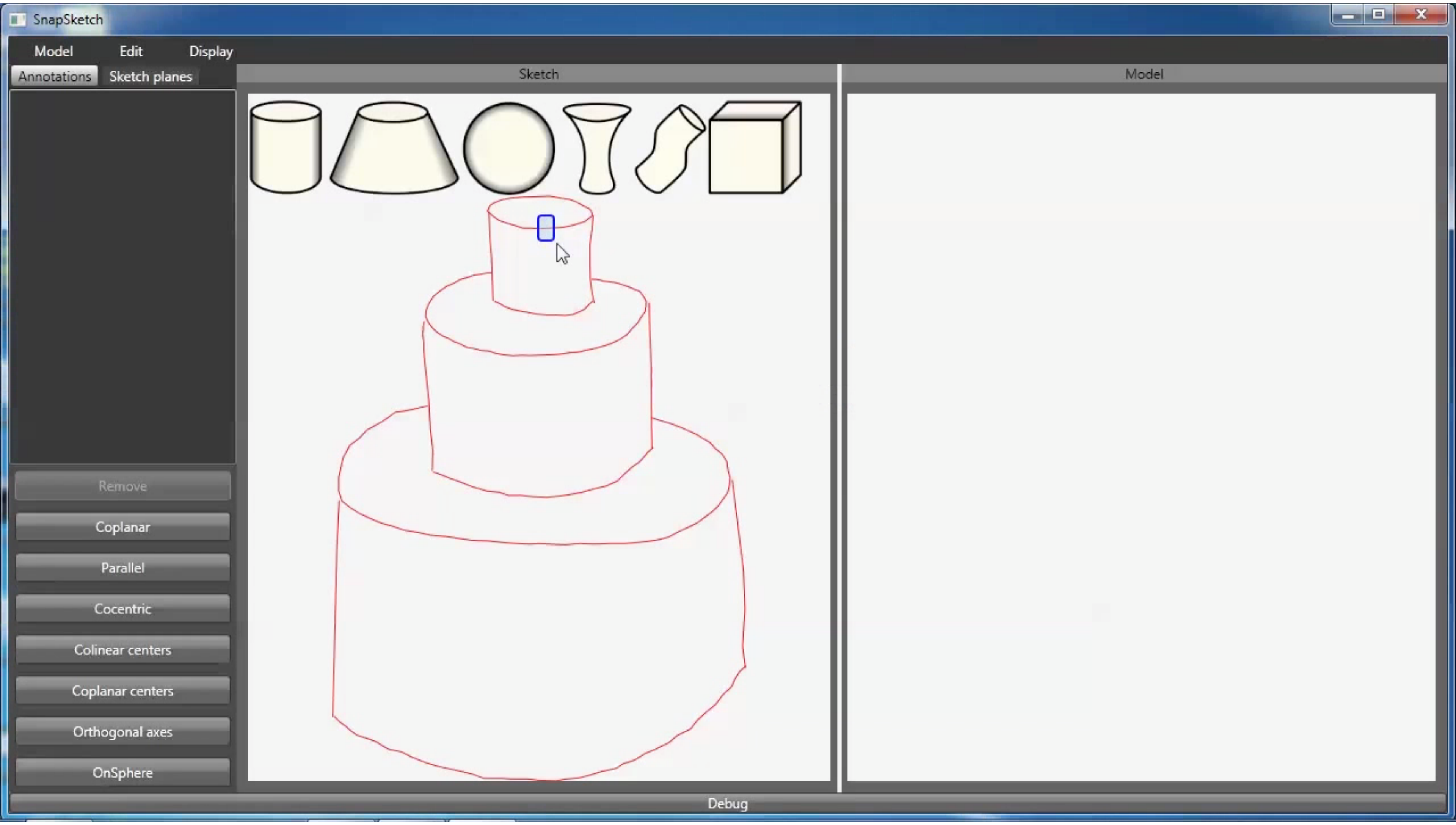
Primitives: Feature Curves



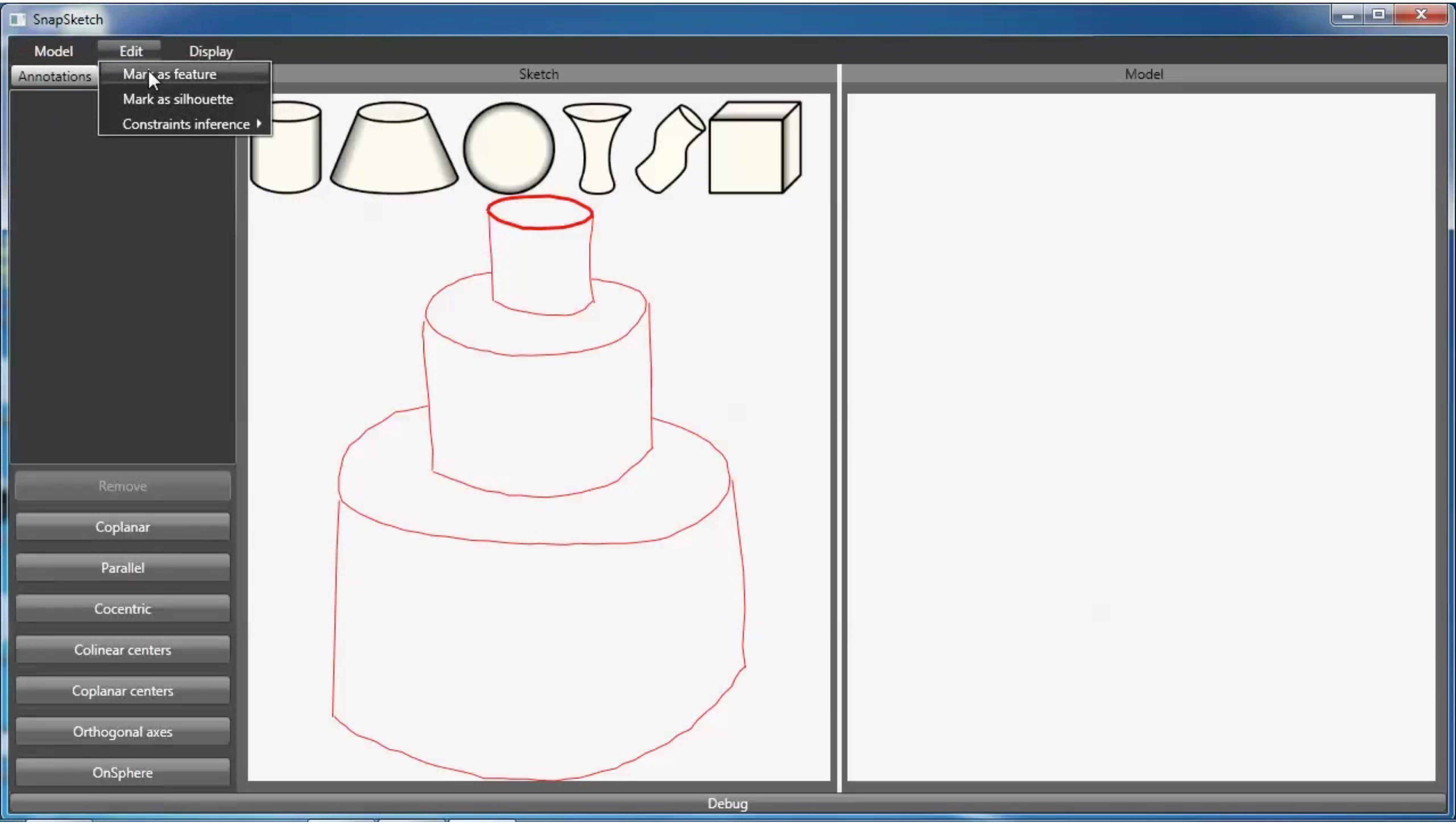
Primitives: Silhouette Curves



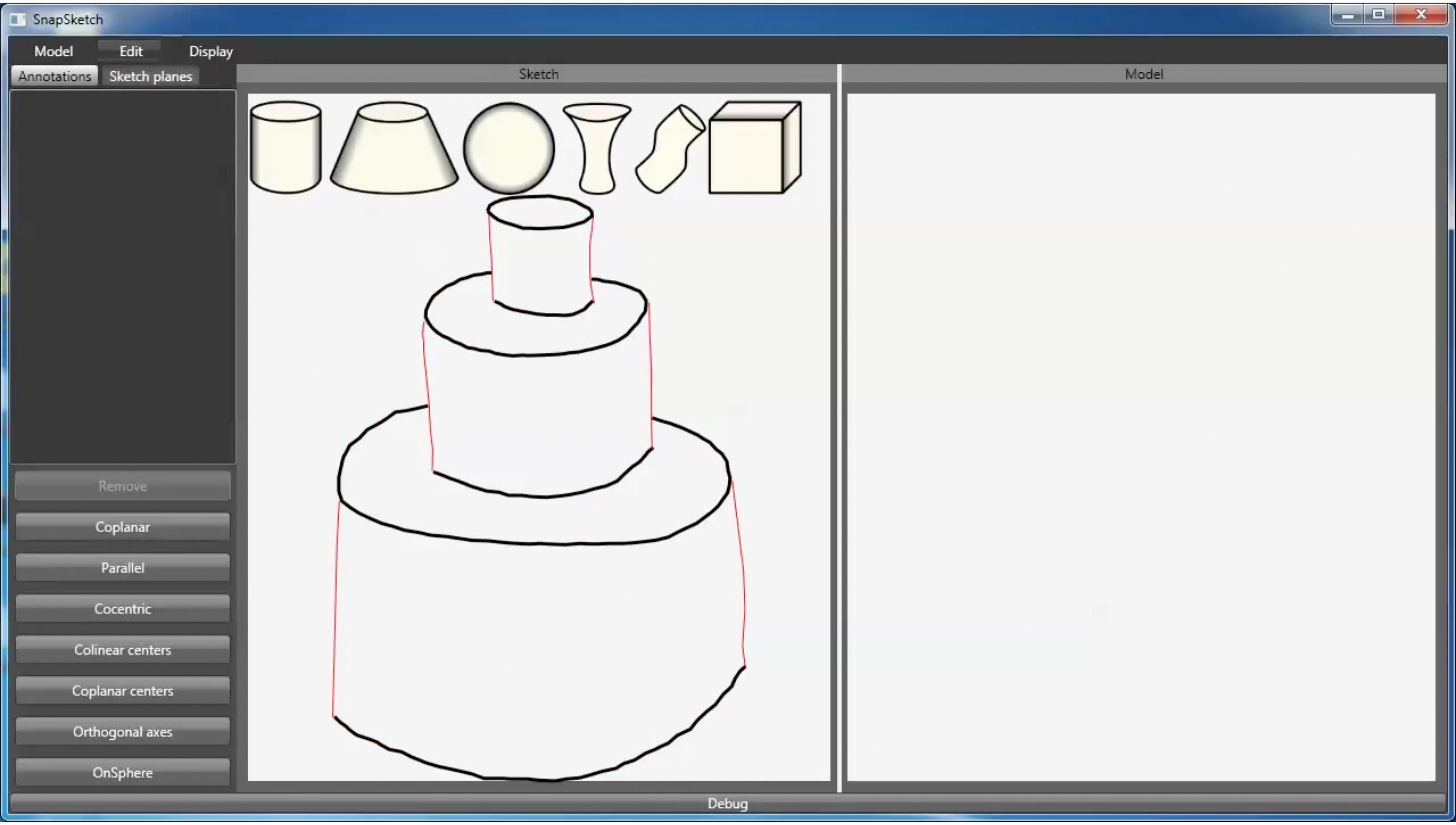
Tagging


SnapSketch
Model
Edit
Display
Annotations
Sketch planes
Sketch
Model
Remove
Coplanar
Parallel
Cocentric
Colinear centers
Coplanar centers
Orthogonal axes
OnSphere
Debug

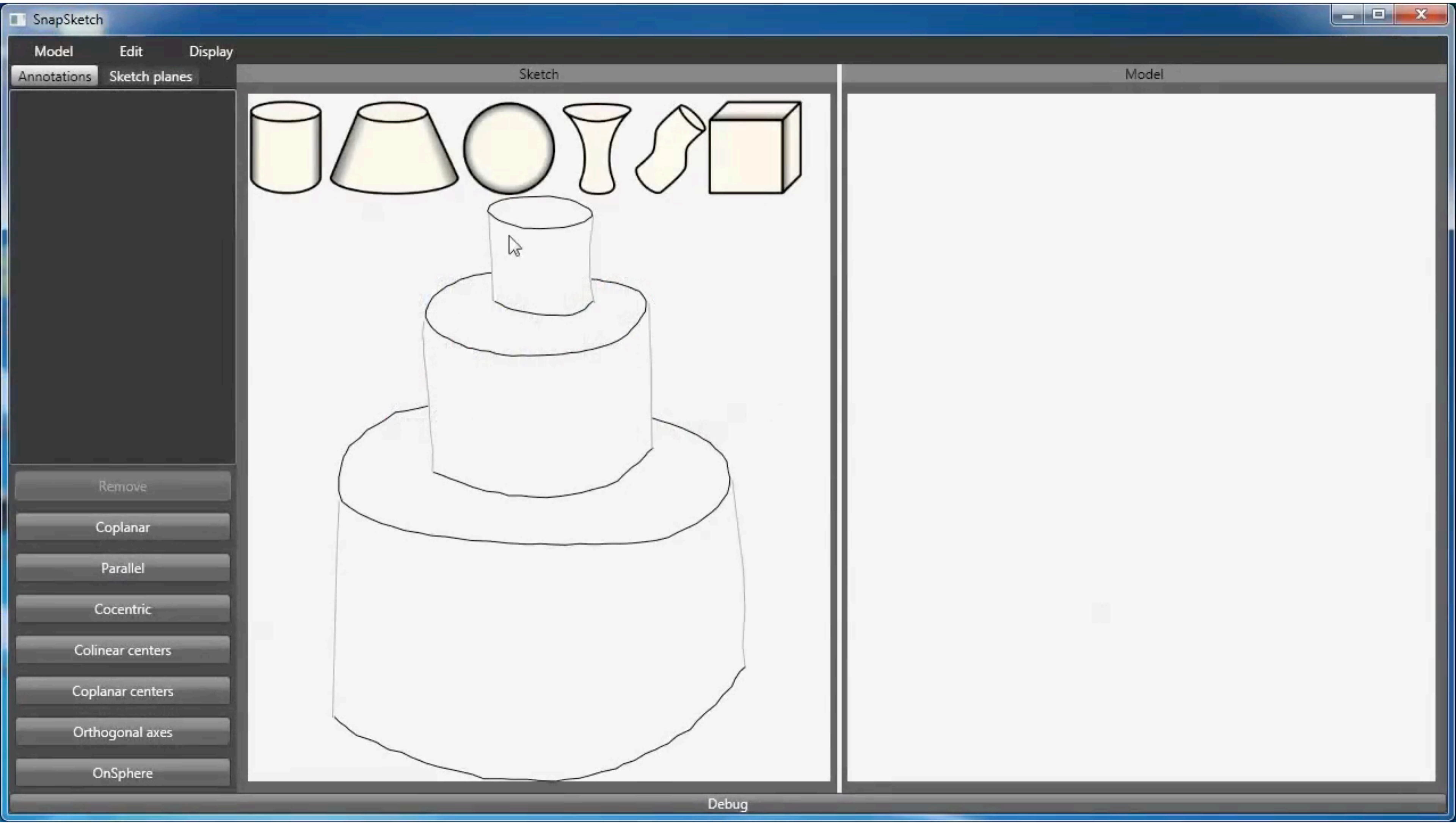
Tagging


SnapSketch
Model
Edit
Display
Annotations
Mark as feature
Mark as silhouette
Constraints inference
Sketch
Model
Remove
Coplanar
Parallel
Cocentric
Colinear centers
Coplanar centers
Orthogonal axes
OnSphere
Debug

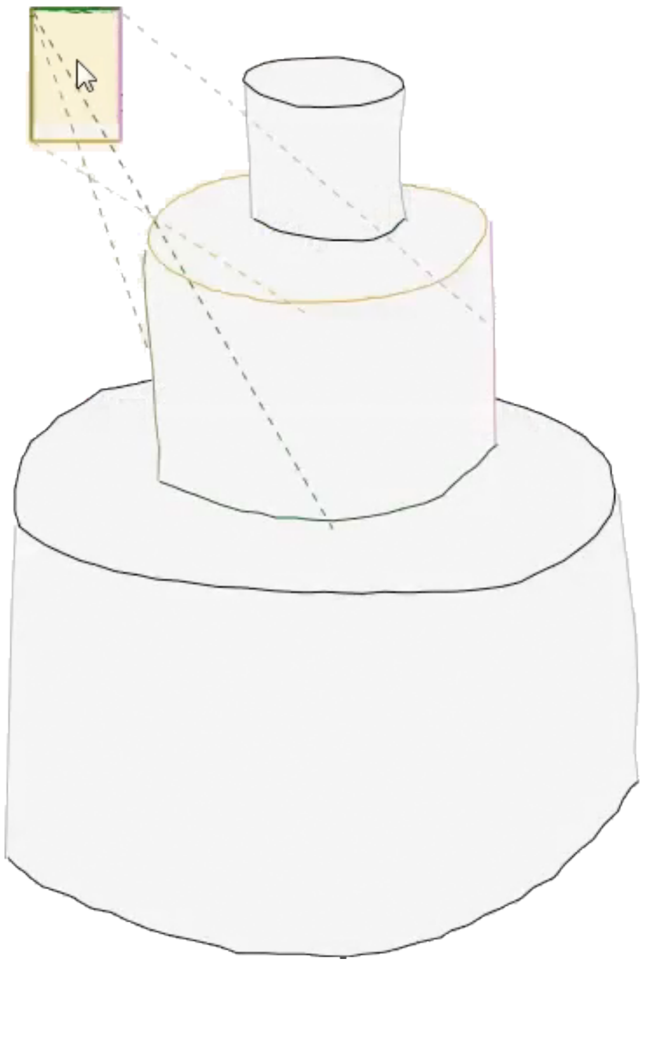
Tagging


SnapSketch
Model
Edit
Display
Annotations
Sketch planes
Sketch
Model
Remove
Coplanar
Parallel
Cocentric
Colinear centers
Coplanar centers
Orthogonal axes
OnSphere
Debug

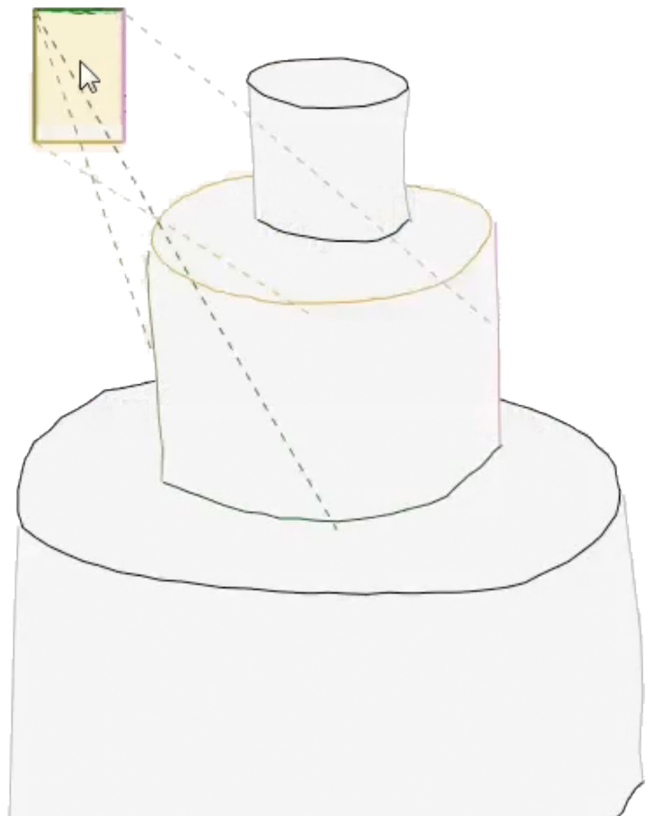
Drag-and-Drop



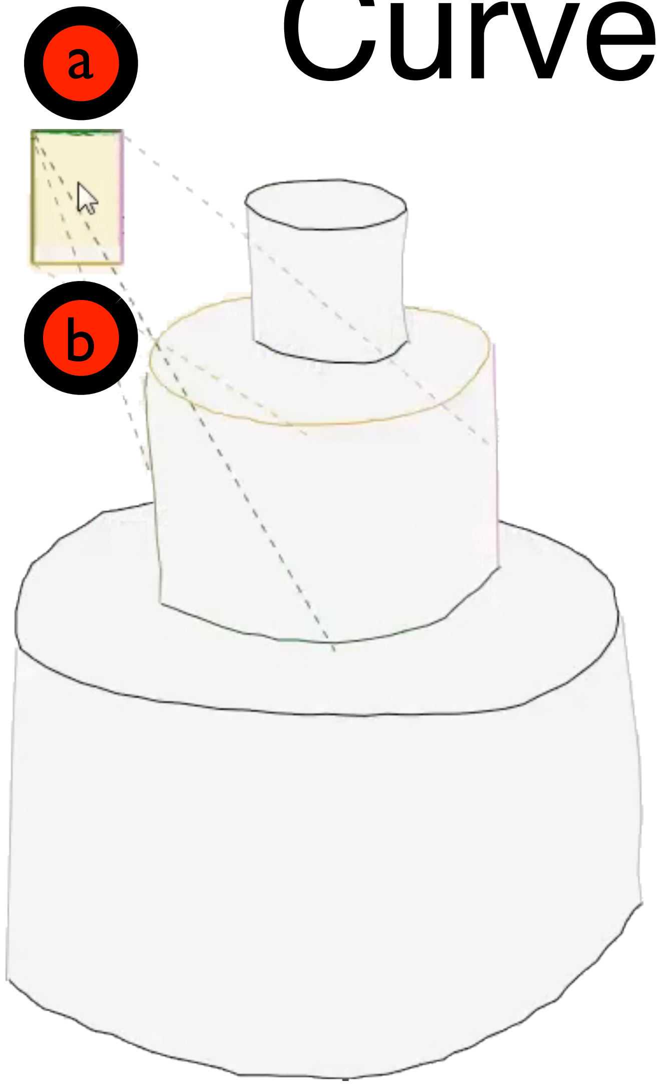
Anatomy of a Drag: Curve Matching

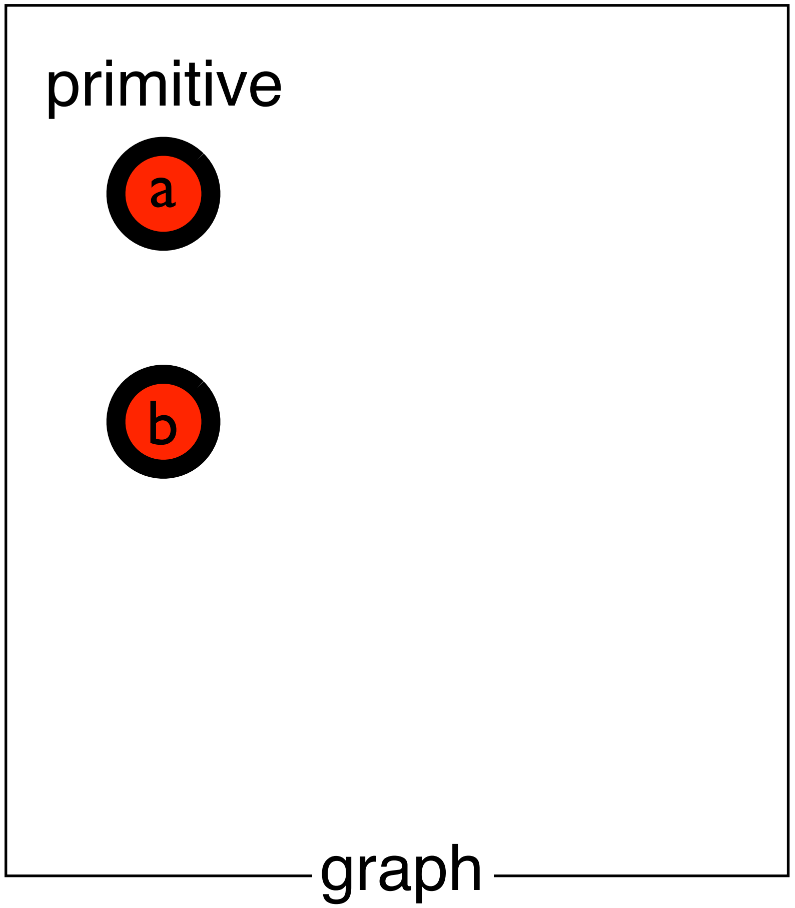


Anatomy of a Drag: Curve Matching

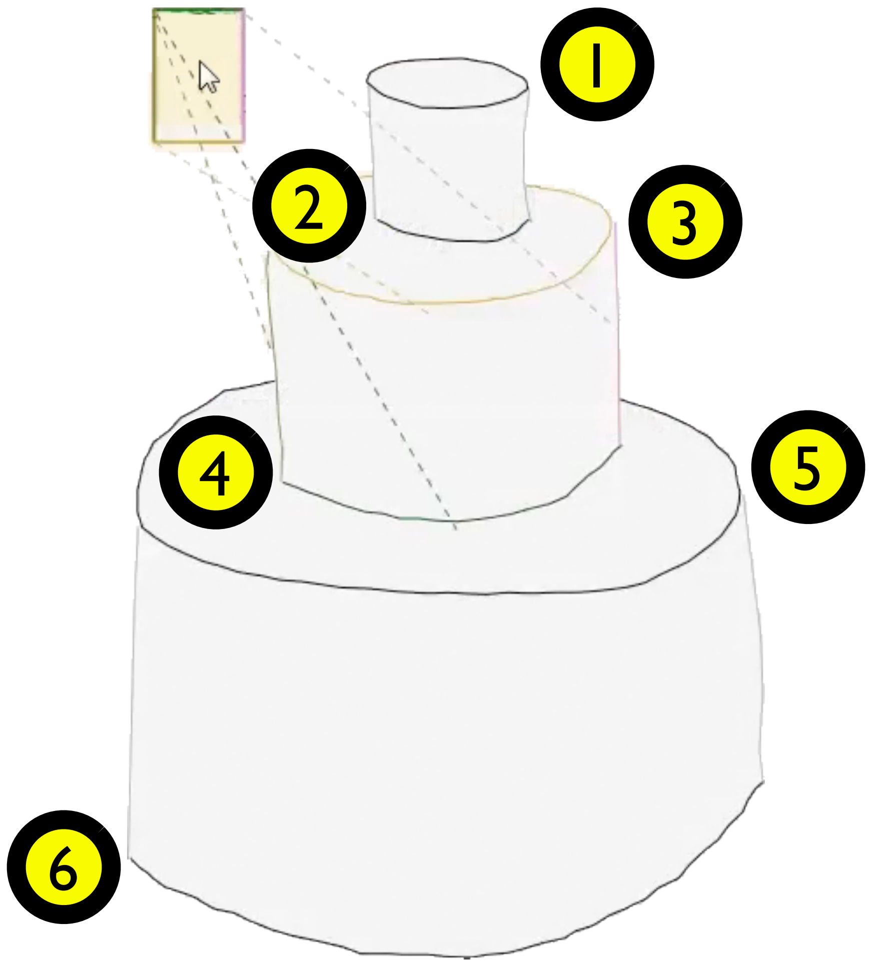


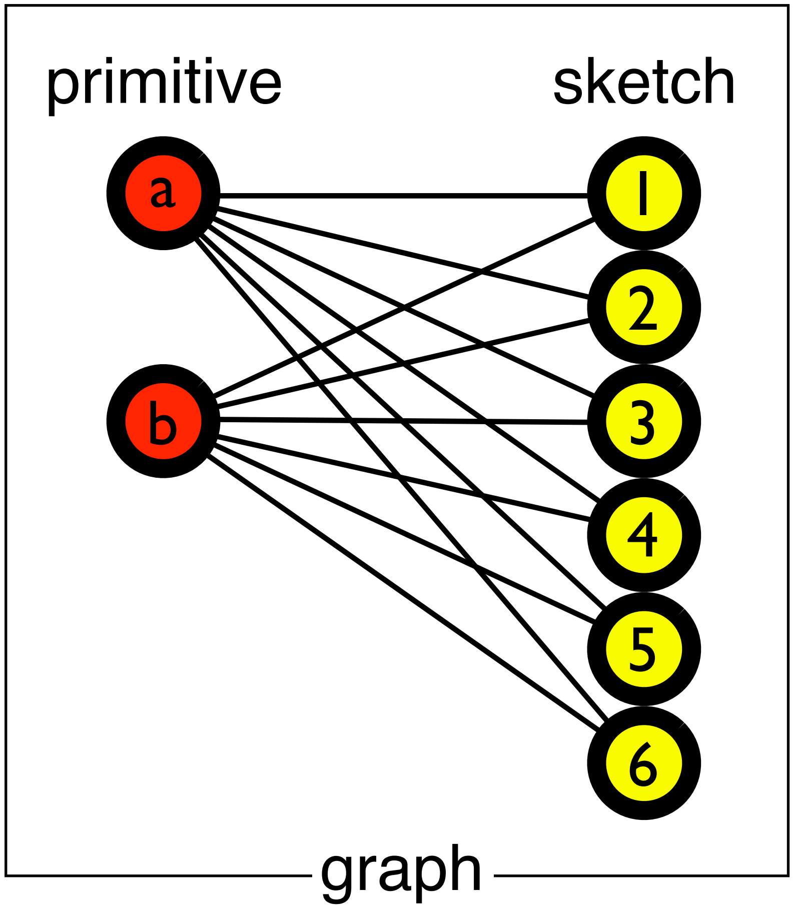
Anatomy of a Drag: Curve Matching



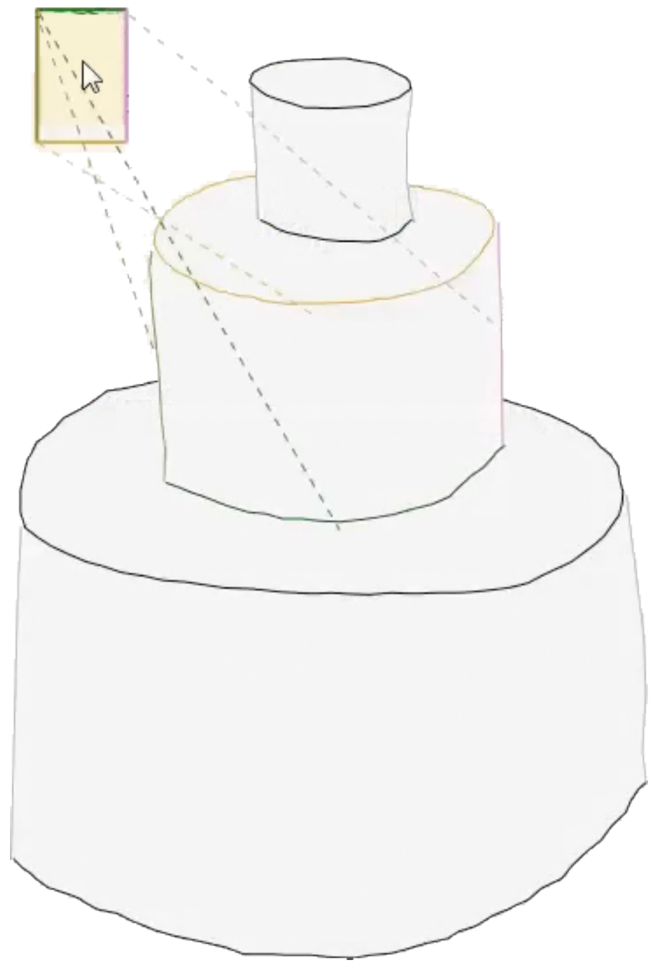


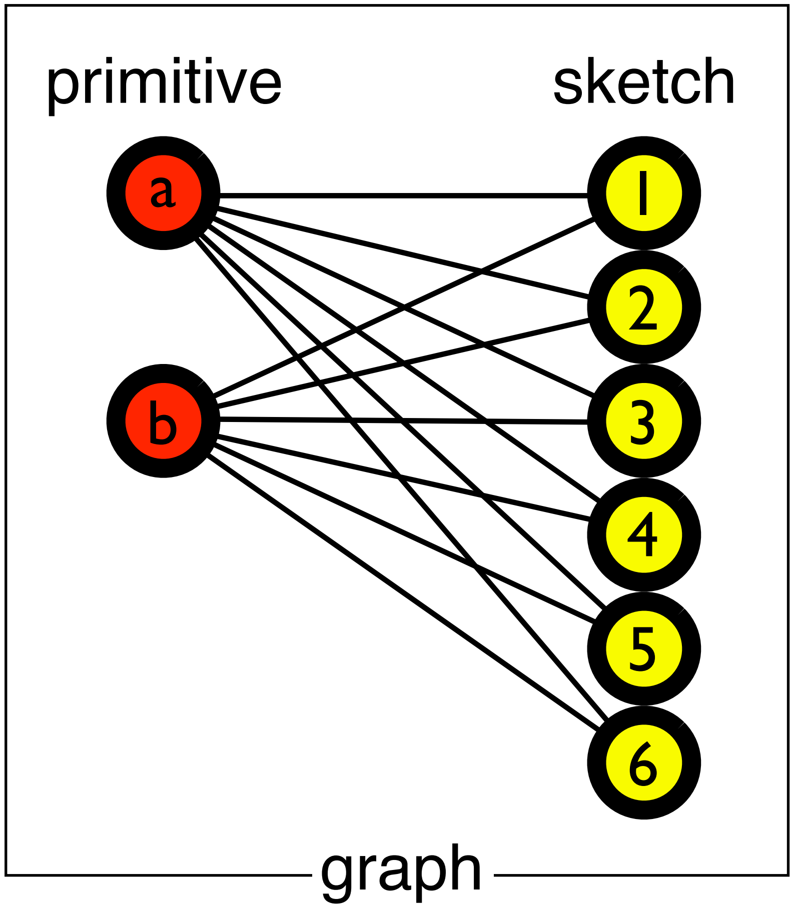
Anatomy of a Drag: Curve Matching



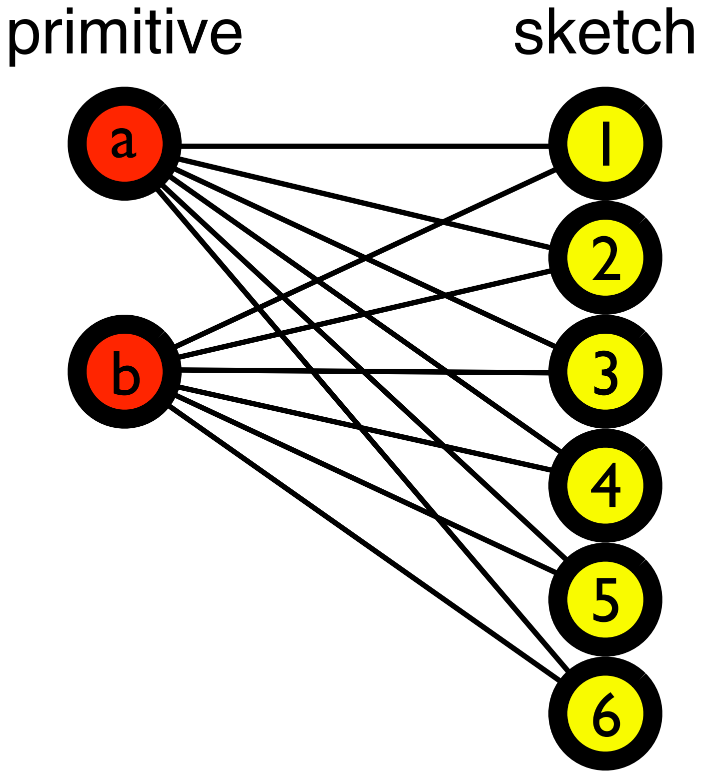


Anatomy of a Drag: Curve Matching

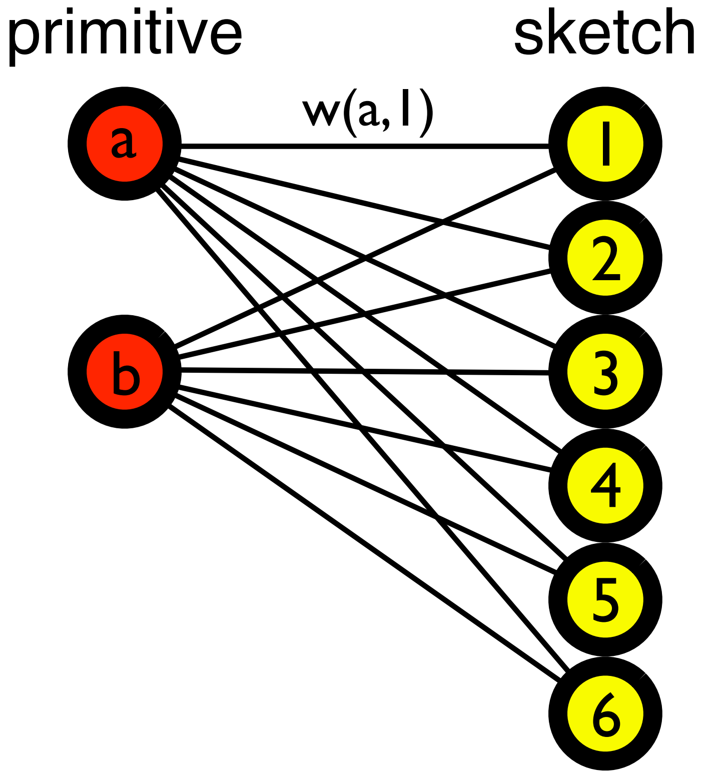




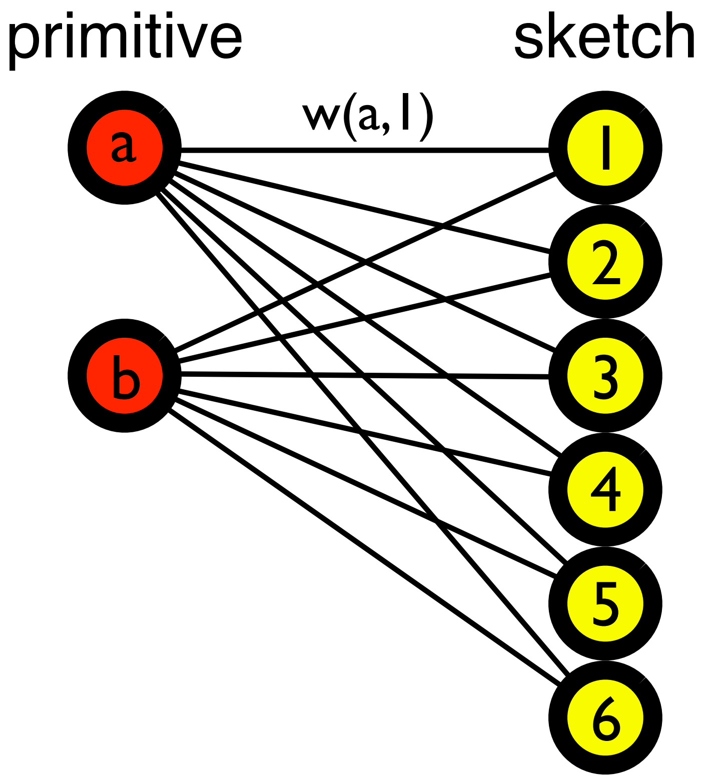
Anatomy of a Drag: Curve Matching

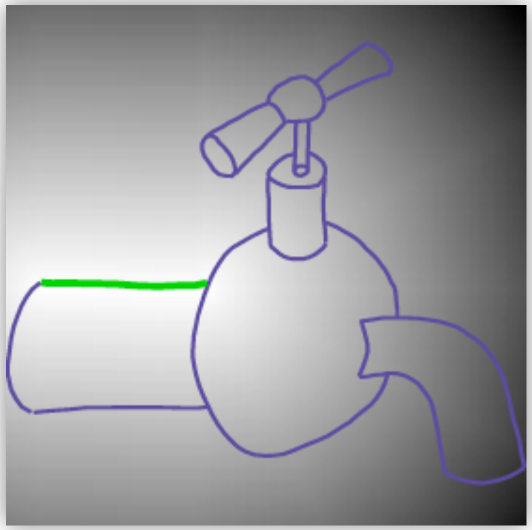


Anatomy of a Drag: Curve Matching

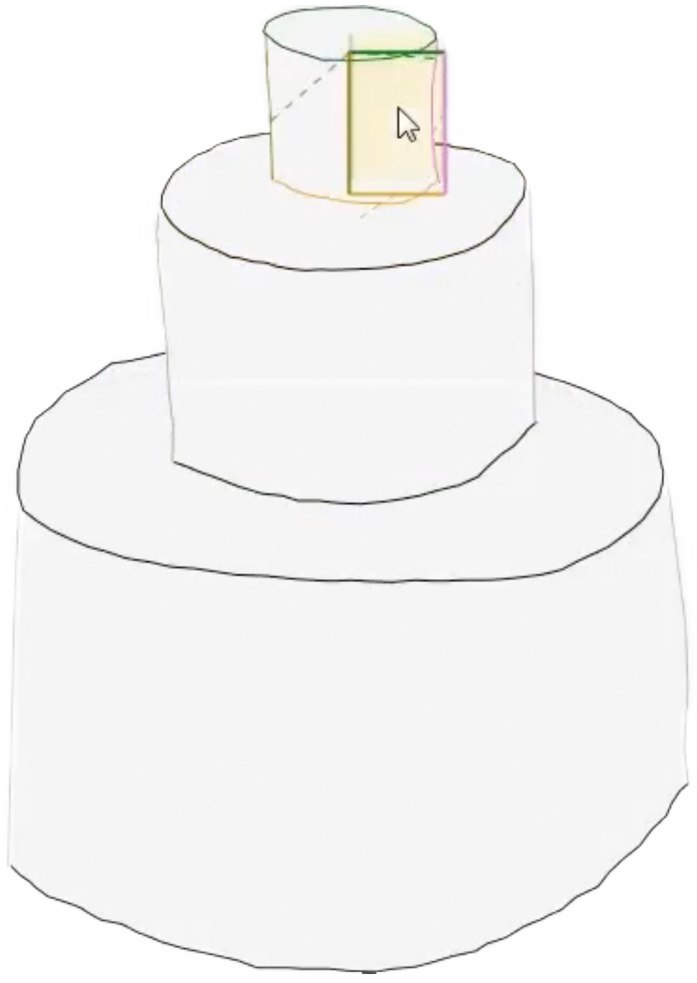


Anatomy of a Drag: Curve Matching

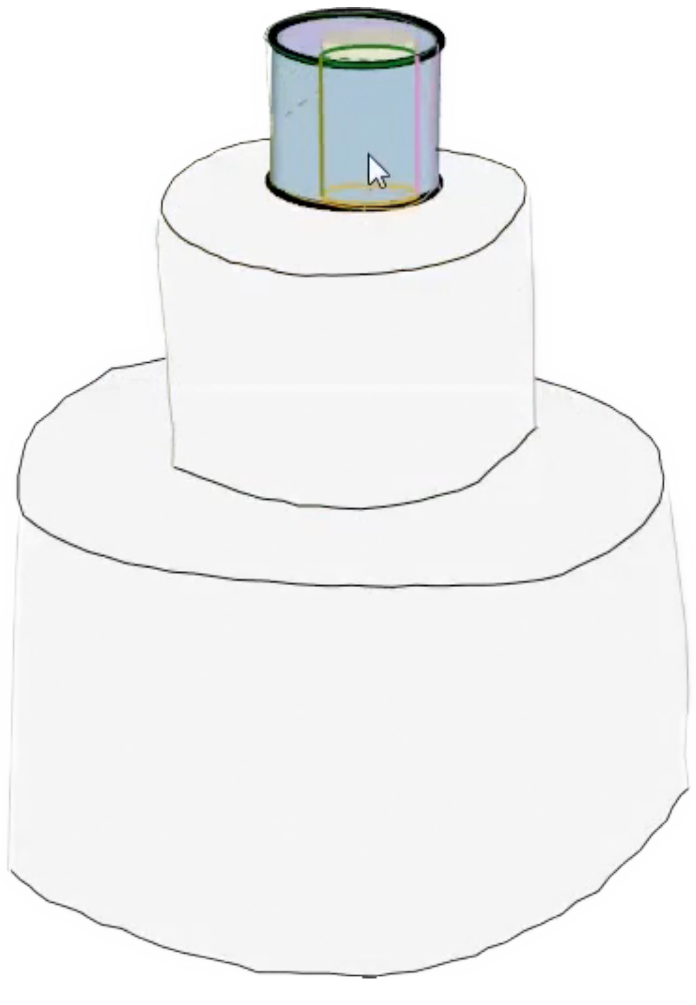




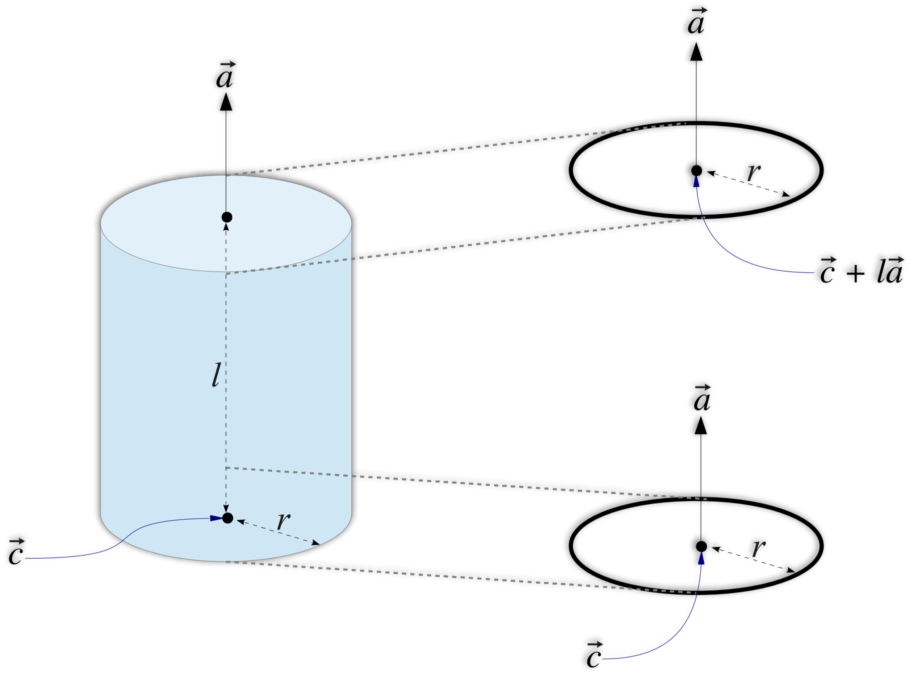
Anatomy of a Drag: Curve Matching



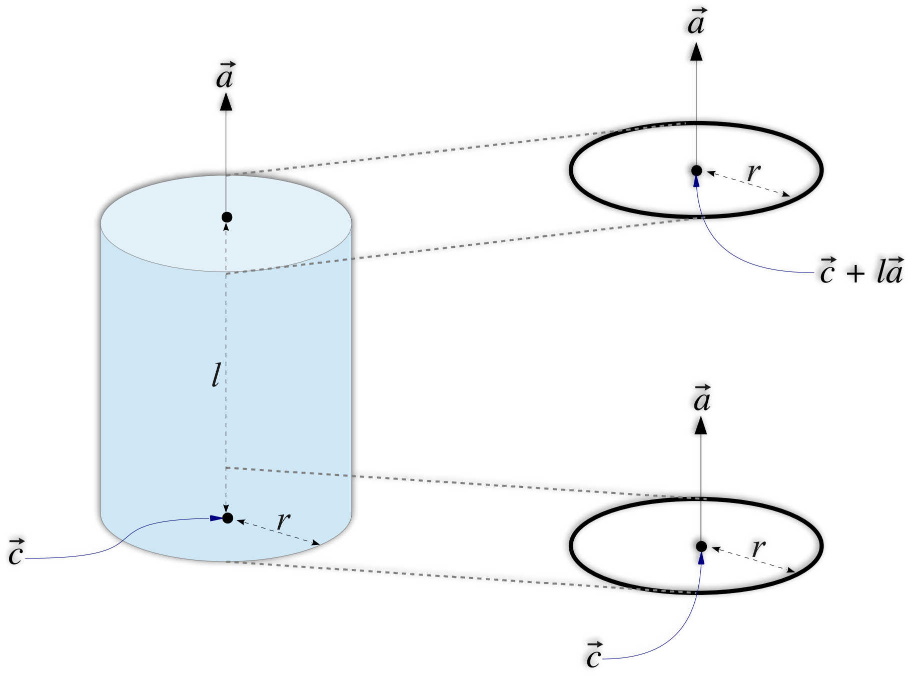
Anatomy of a Drag: Primitive Fitting



Anatomy of a Drag: Primitive Fitting

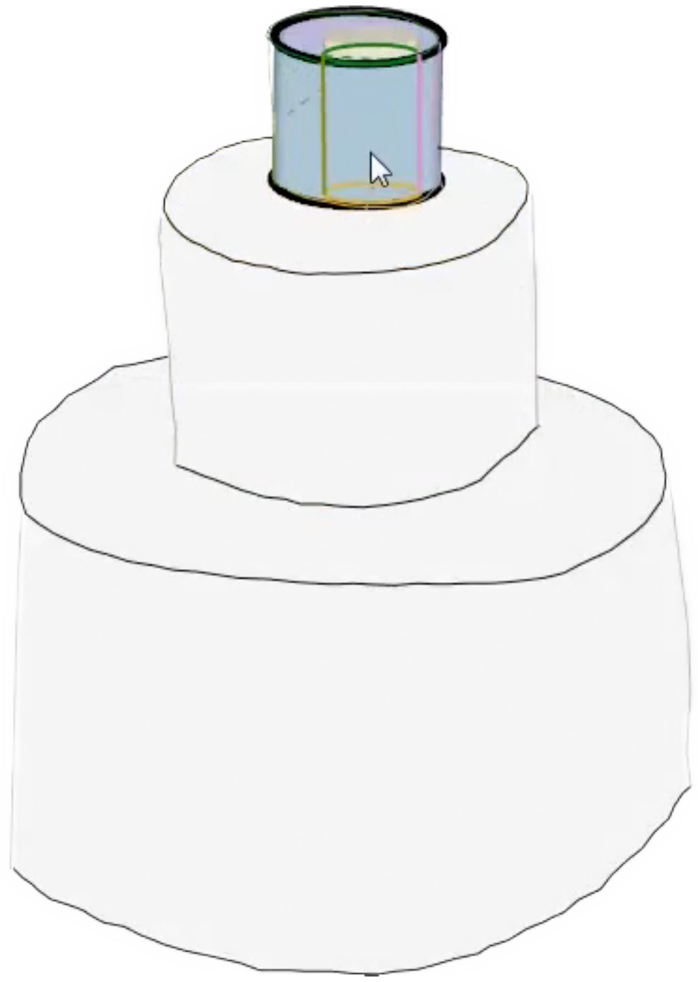


Anatomy of a Drag: Primitive Fitting

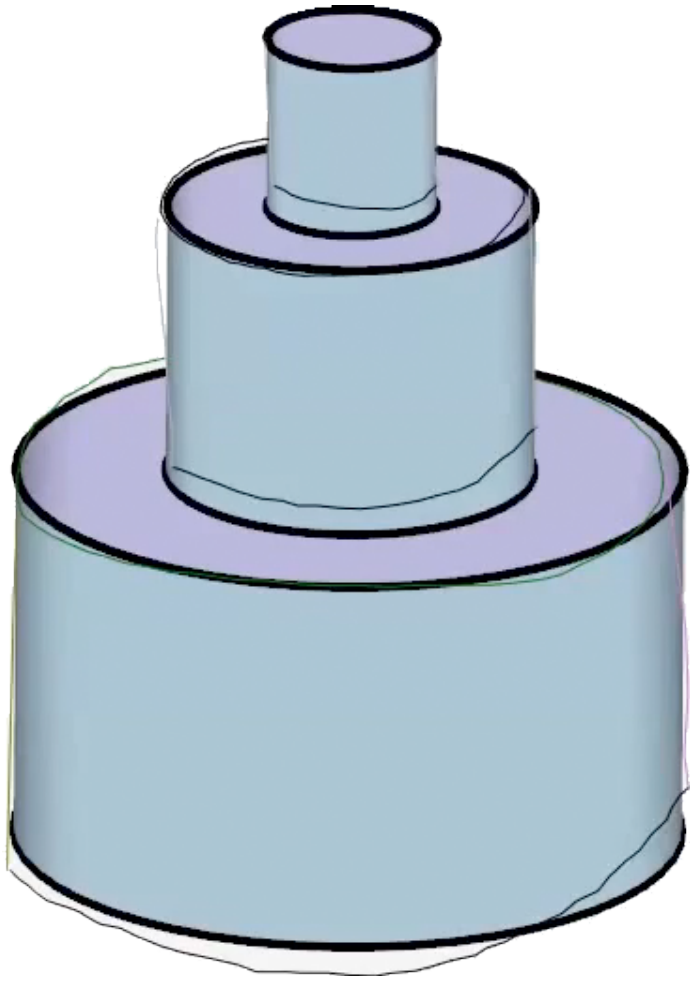


$$\begin{aligned} min: &\quad \phi_p(x_p) \\ s.t.: &\quad C_p(x_p) = 0 \end{aligned}$$

Anatomy of a Drag: Primitive Fitting

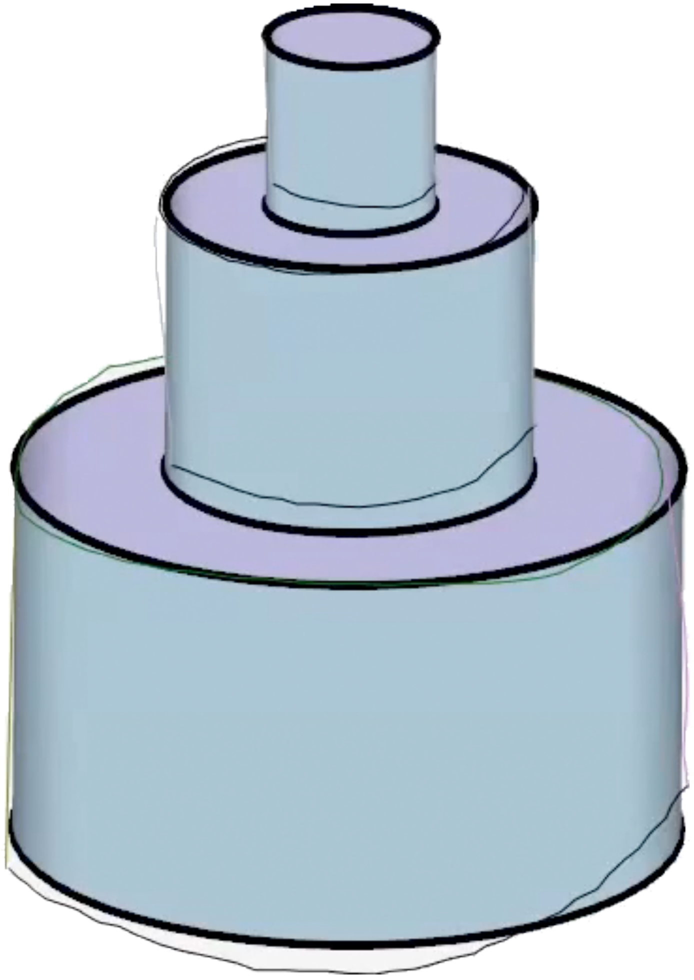


Anatomy of a Drag: Geosemantic Relations



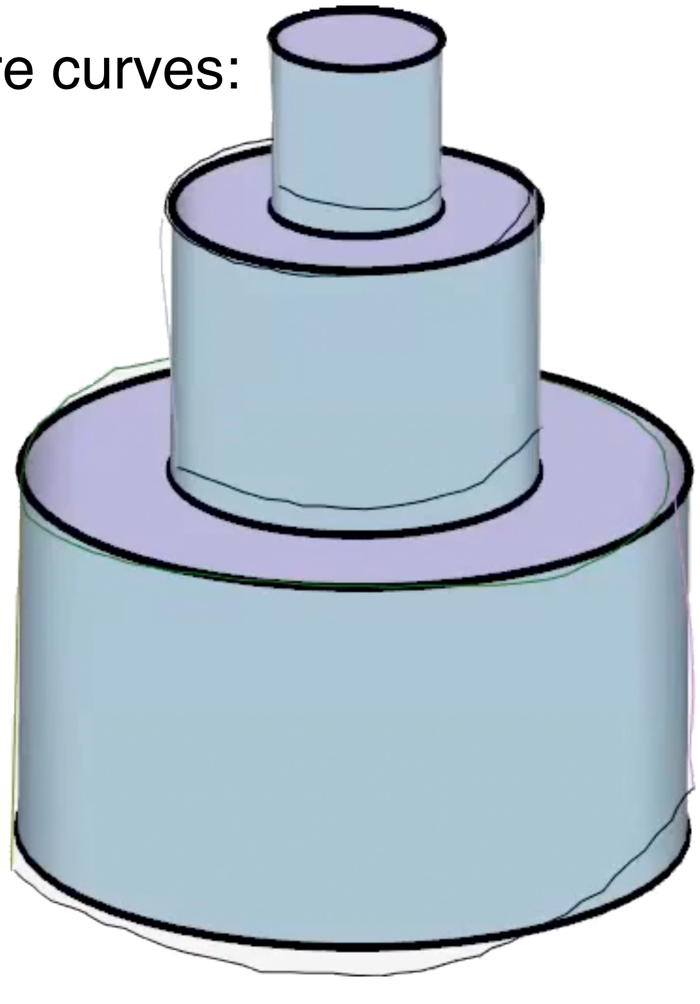
Anatomy of a Drag: Geosemantic Relations

$$
\begin{array}{lll}
\min\limits_{x} : & \sum_{p \in P} \phi_p(x_p) & \\
s.t. : & C_p(x_p) = 0 & \forall p \in P \\
 & \psi_g(x_g) = 0 & \forall g \in G
\end{array}
$$



Geosemantic Relations

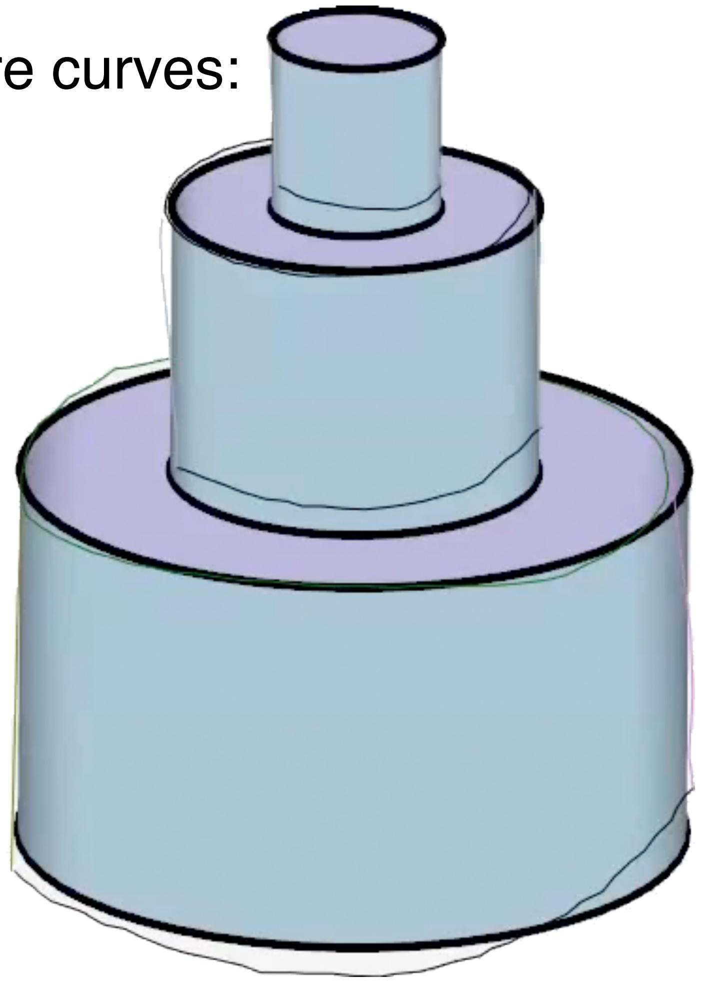
Constraints linking two or more feature curves:



Geosemantic Relations

Constraints linking two or more feature curves:

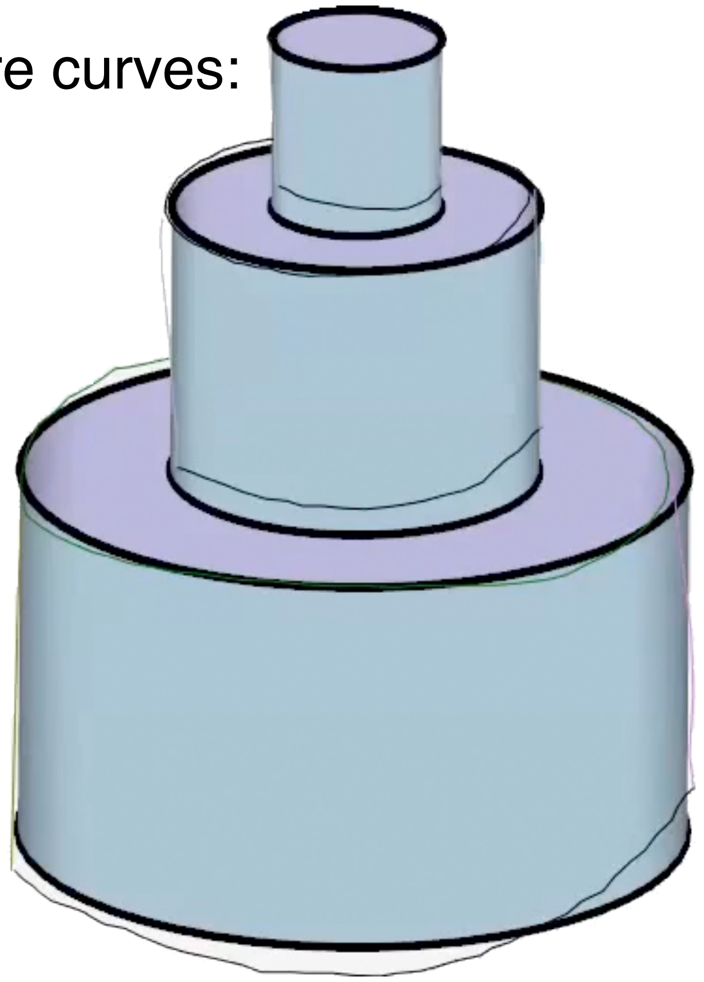
- Parallelism



Geosemantic Relations

Constraints linking two or more feature curves:

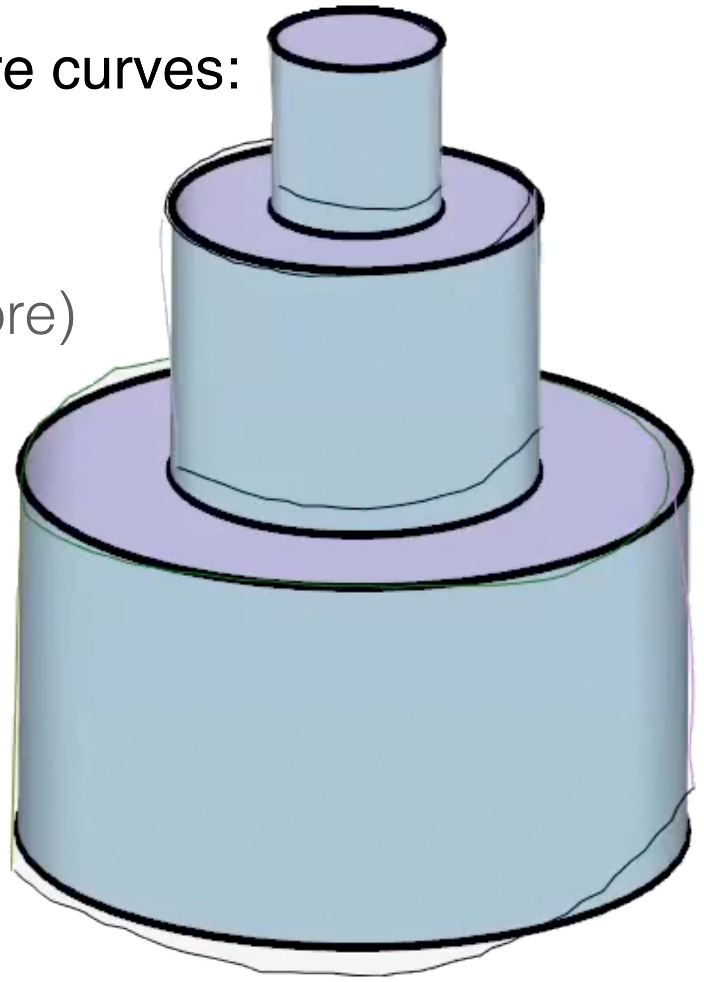
- Parallelism
- Orthogonality



Geosemantic Relations

Constraints linking two or more feature curves:

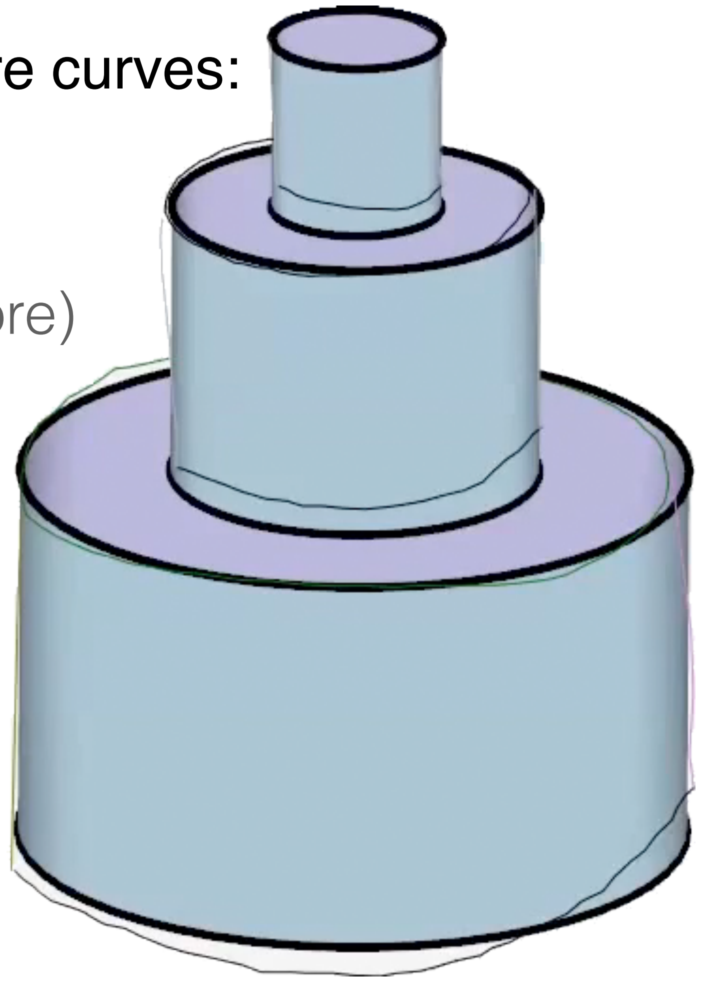
- Parallelism
- Orthogonality
- Collinear centers (three or more)



Geosemantic Relations

Constraints linking two or more feature curves:

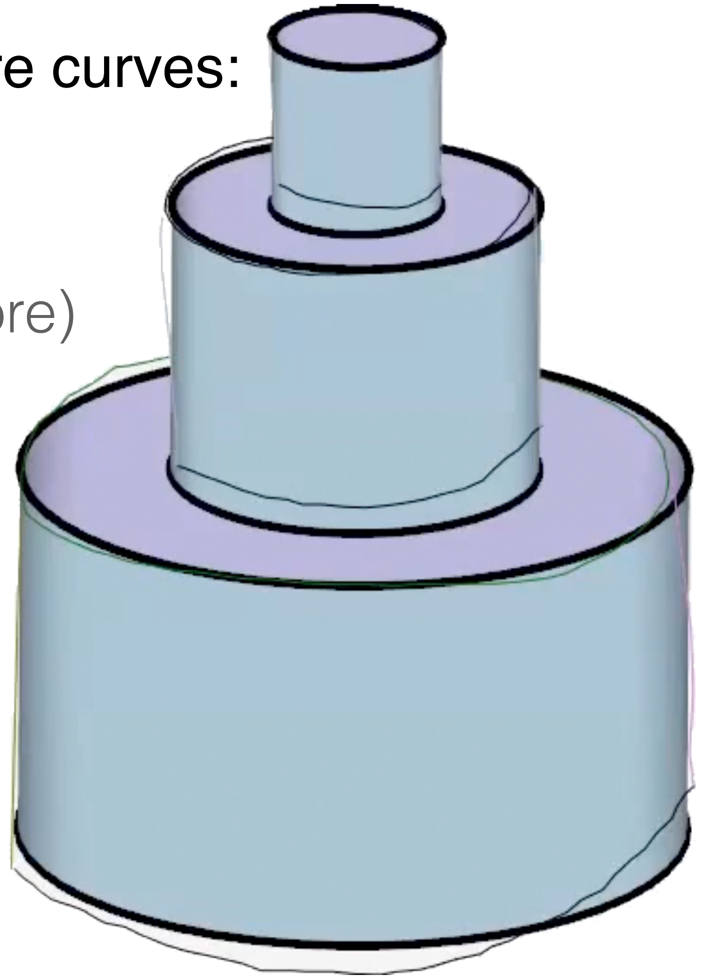
- Parallelism
- Orthogonality
- Collinear centers (three or more)
- Concentric



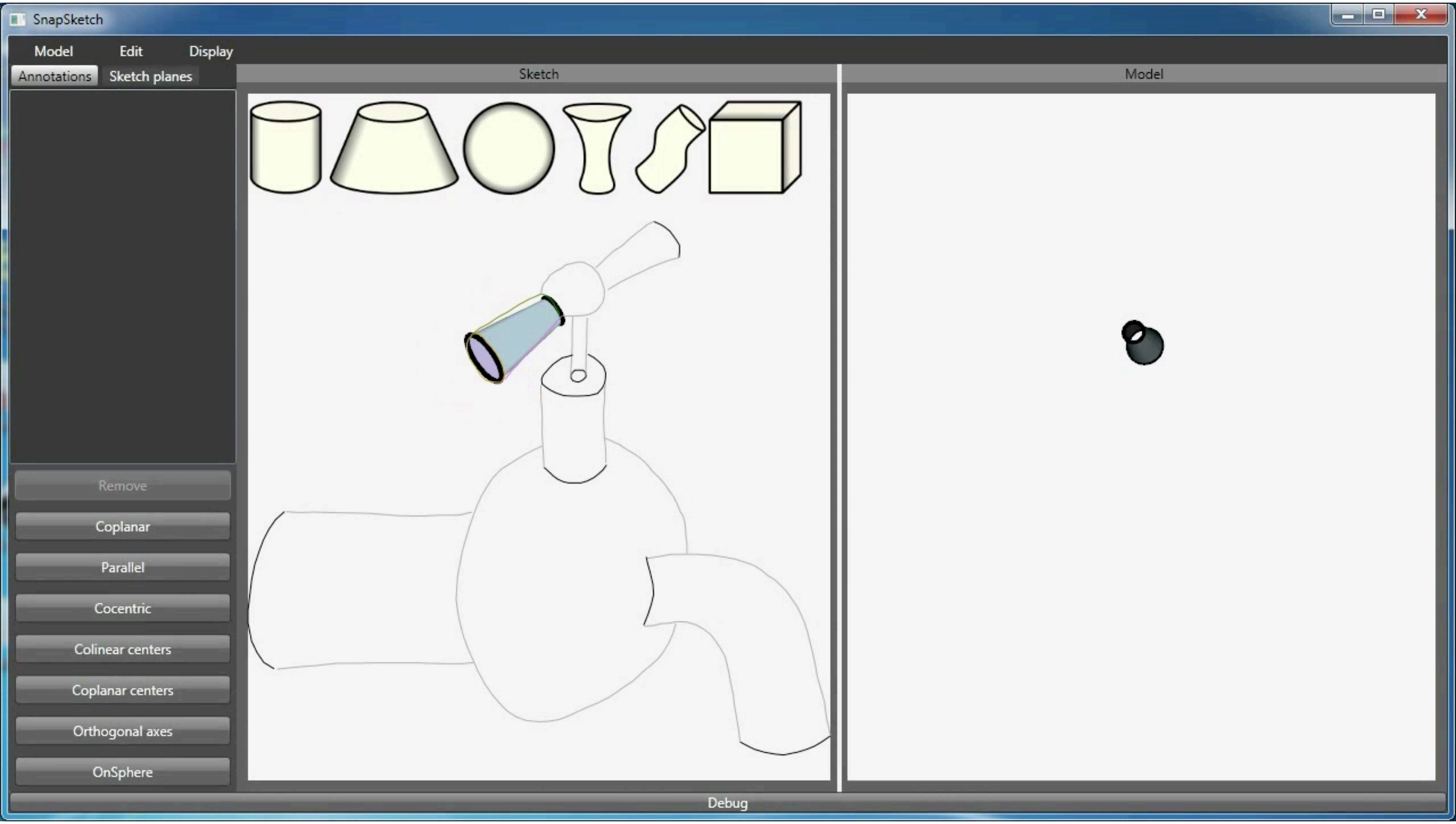
Geosemantic Relations

Constraints linking two or more feature curves:

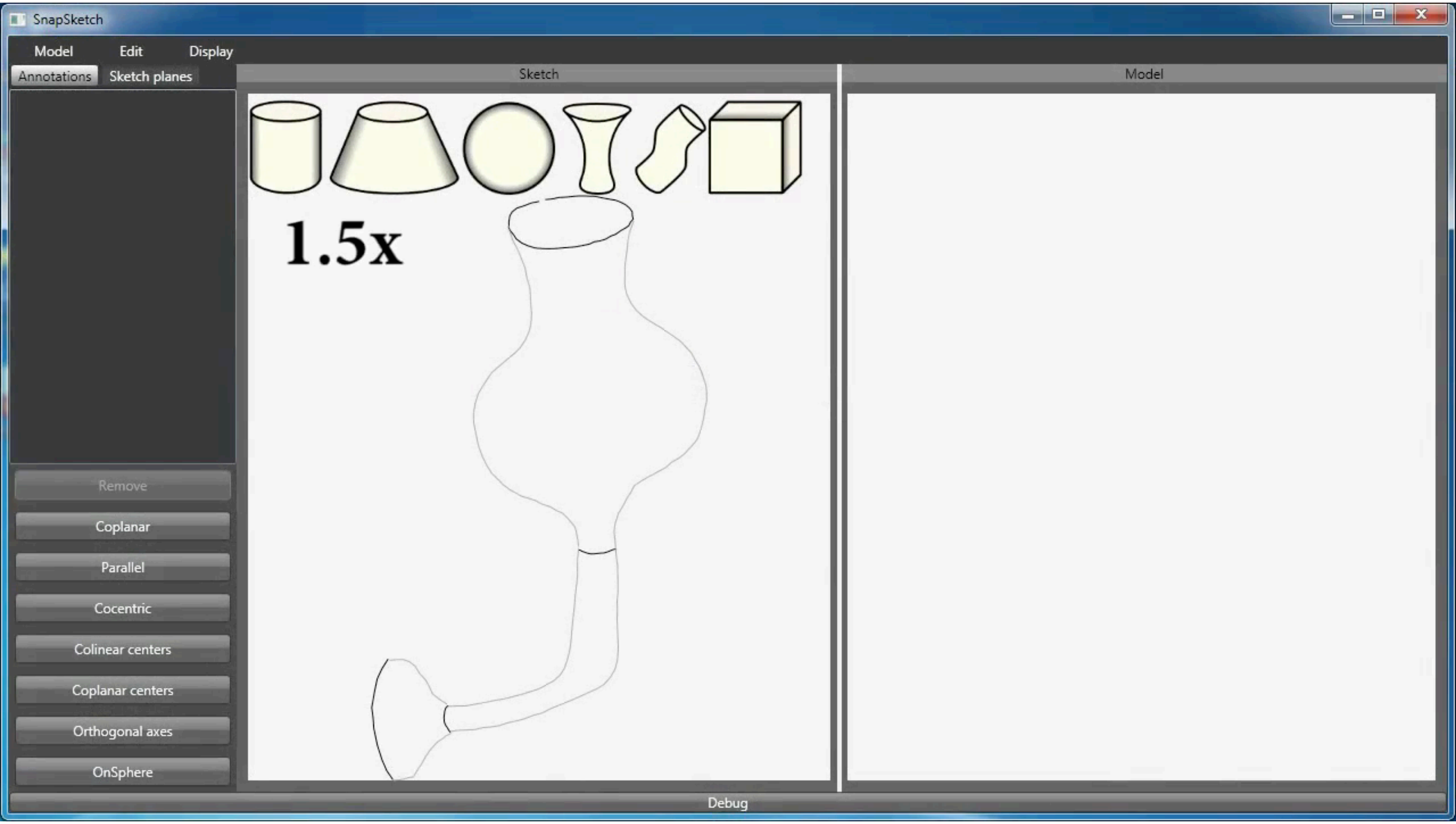
- Parallelism
- Orthogonality
- Collinear centers (three or more)
- Concentric
- Coplanar

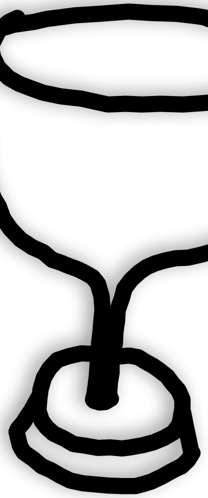
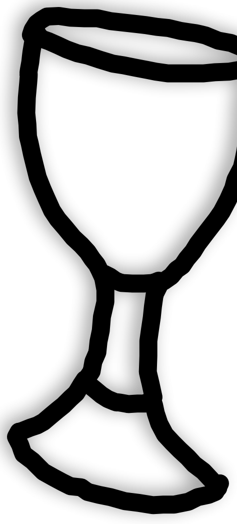
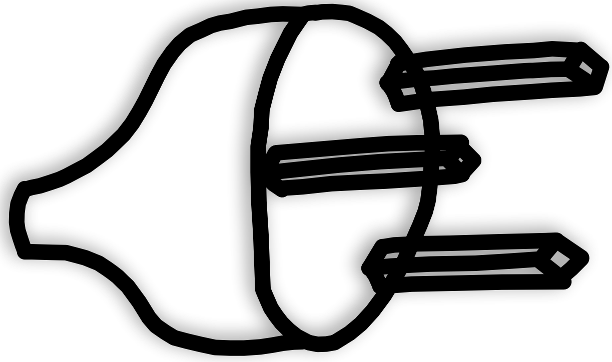
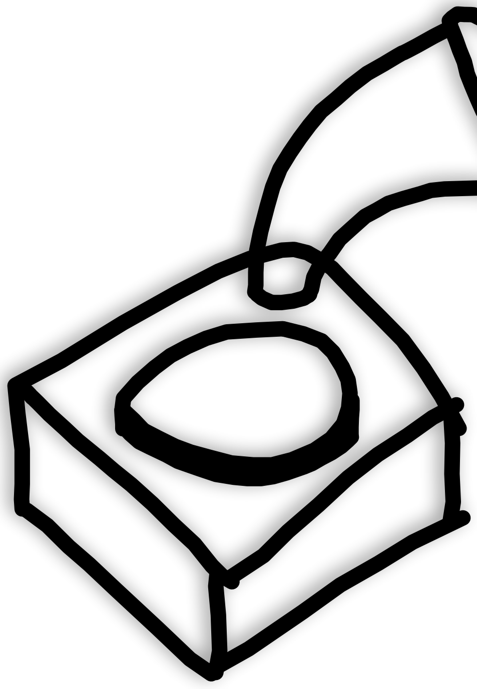
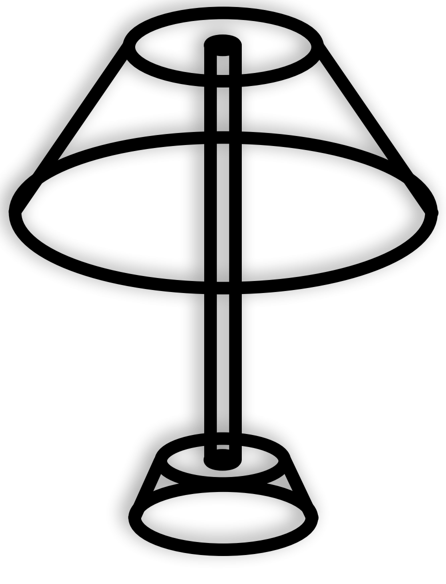
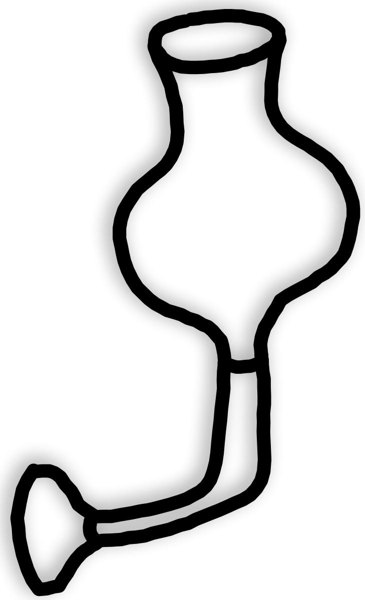


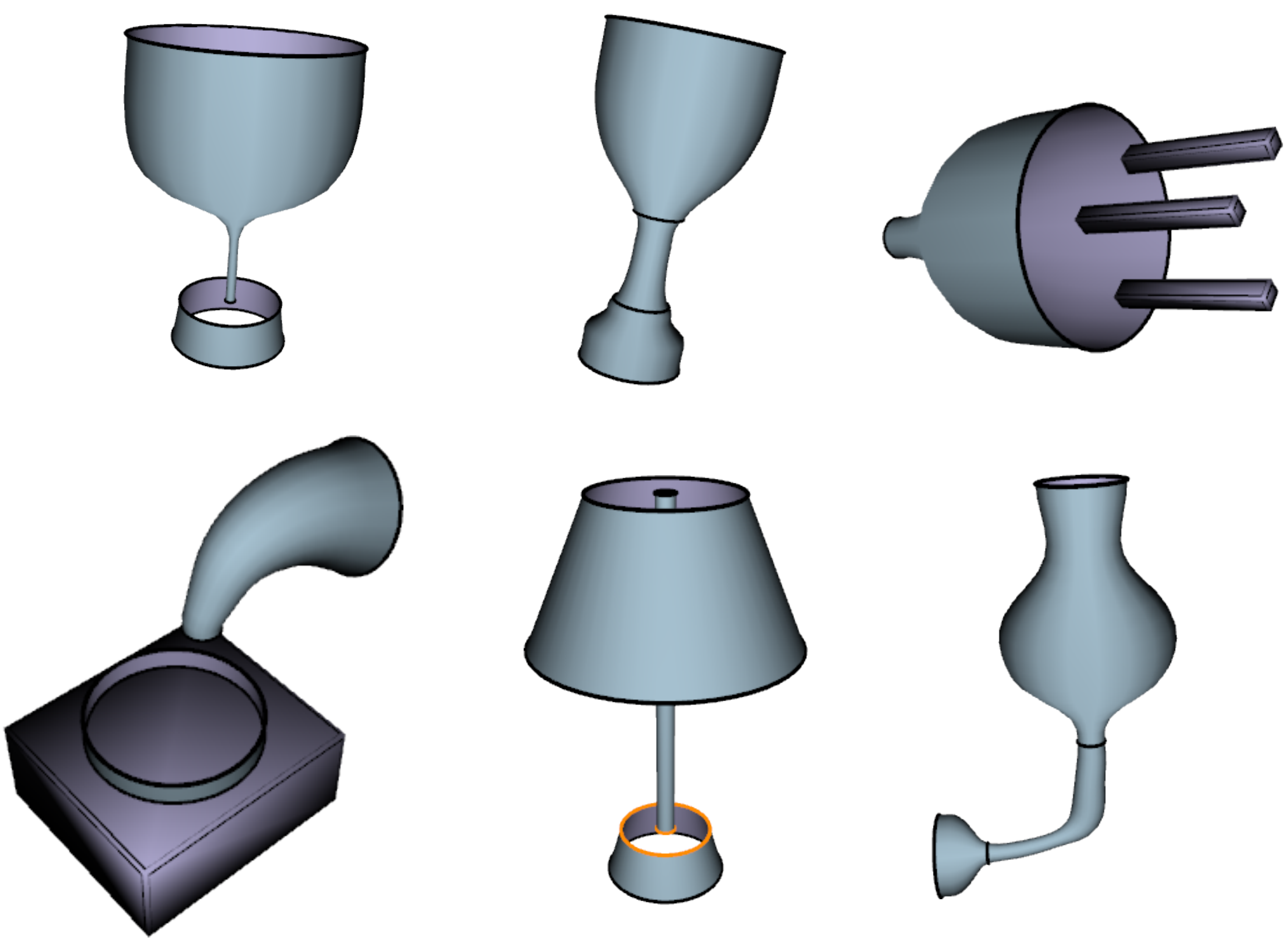
Results

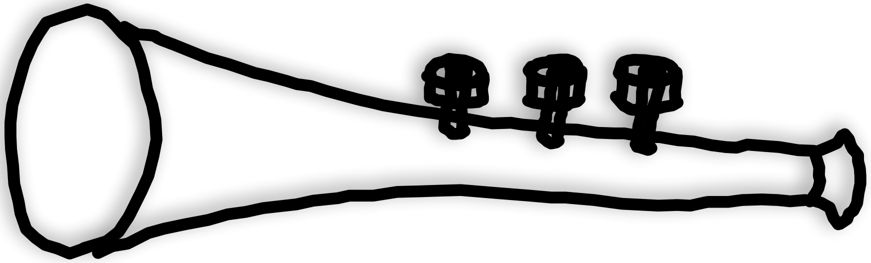

SnapSketch
Model
Edit
Display
Annotations
Sketch planes
Sketch
Model
Remove
Coplanar
Parallel
Cocentric
Colinear centers
Coplanar centers
Orthogonal axes
OnSphere
Debug

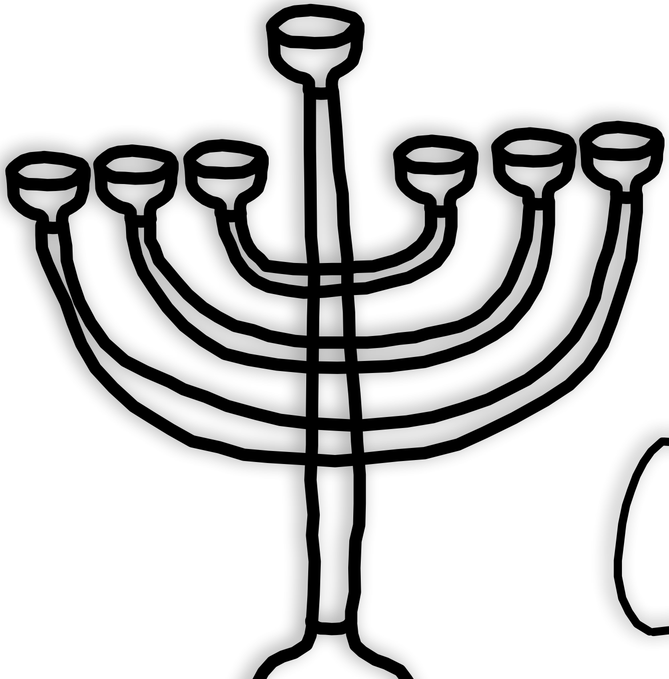
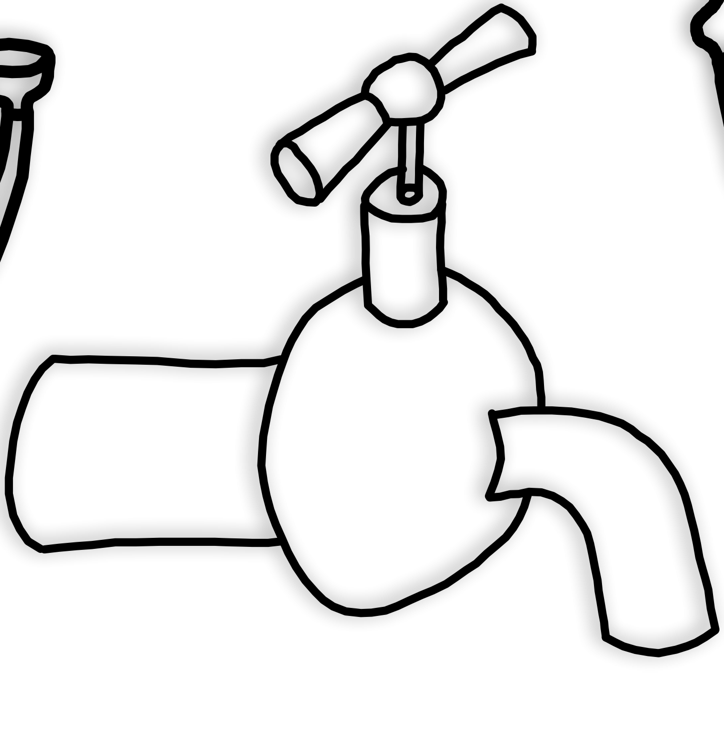
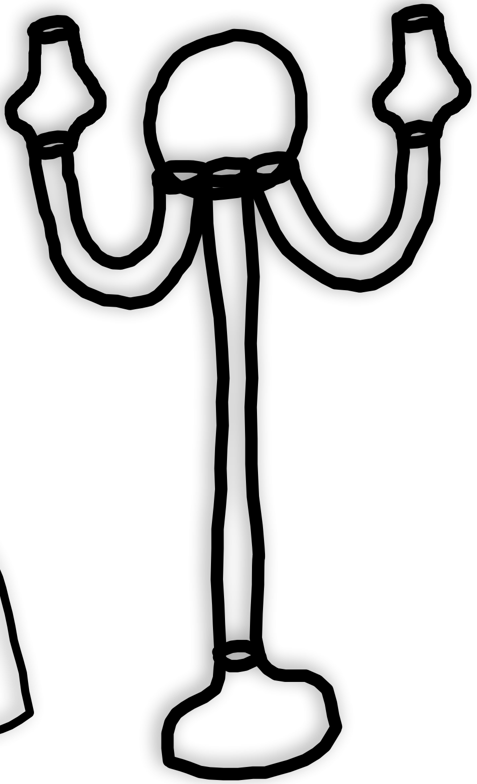
~2x

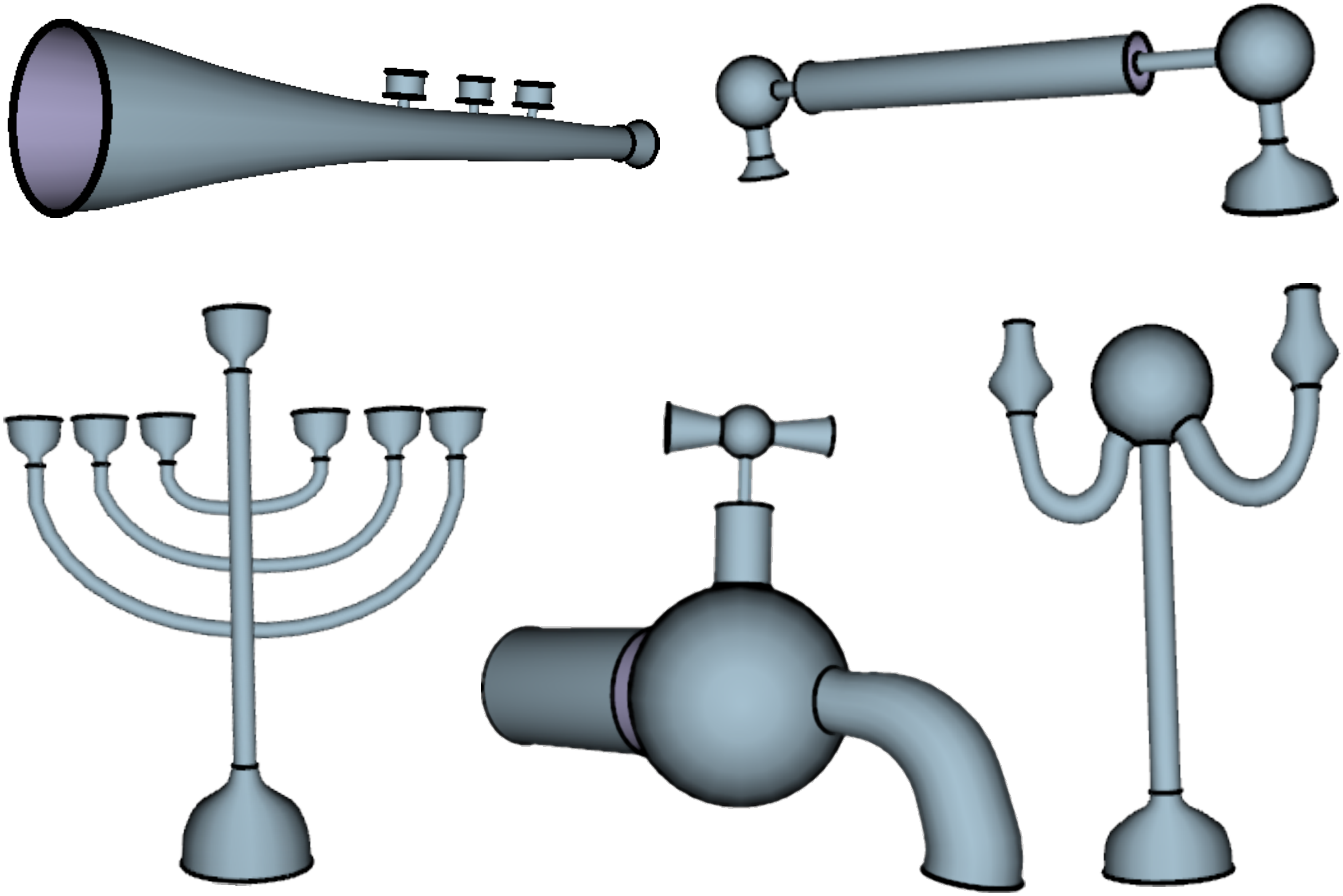

SnapSketch
Model
Edit
Display
Annotations
Sketch planes
Sketch
Model
1.5x
Remove
Coplanar
Parallel
Cocentric
Colinear centers
Coplanar centers
Orthogonal axes
OnSphere
Debug



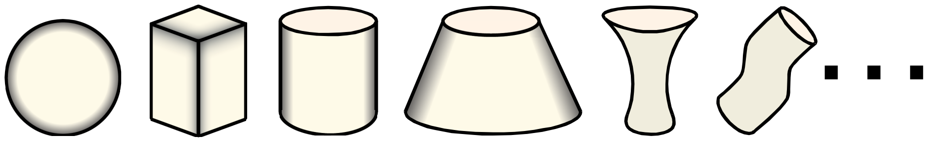


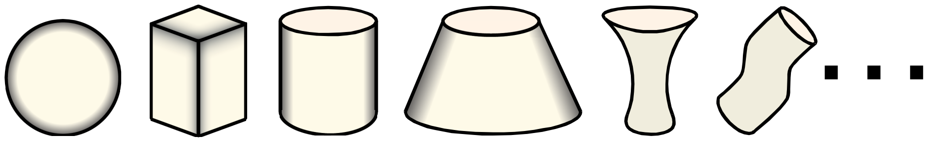


Limitations & Future Work

Limitations & Future Work

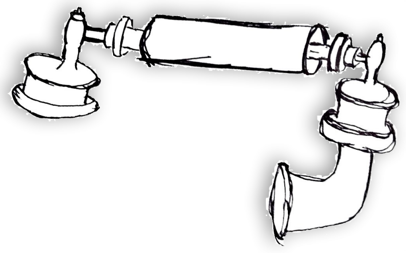
More primitives 

Limitations & Future Work

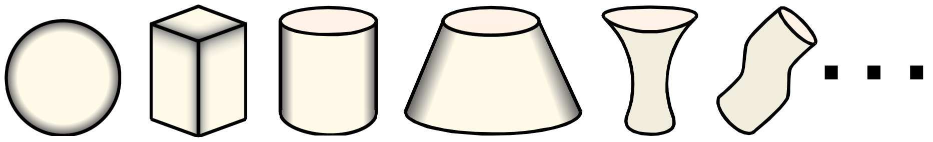
More primitives 

Operate directly on raster sketches

Eliminate sketch curve classification

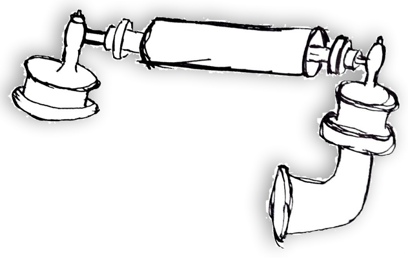


Limitations & Future Work

More primitives 

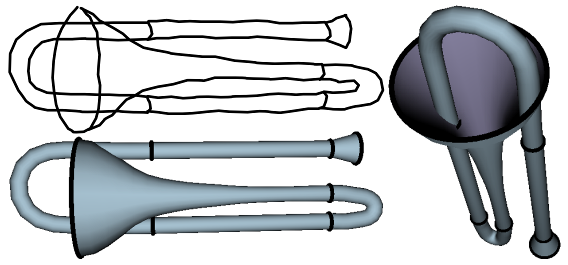
Operate directly on raster sketches

Eliminate sketch curve classification



Sketched occlusions

More geosemantic relations



Conclusion

Make a highly **non-convex** problem tractable by:

Conclusion

Make a highly **non-convex** problem tractable by:

- Introducing an interactive solution.

Conclusion

Make a highly **non-convex** problem tractable by:

- Introducing an interactive solution.
- Separating that which is easy for a **human** and challenging for a **computer**.

Conclusion

Make a highly **non-convex** problem tractable by:

- Introducing an interactive solution.
- Separating that which is easy for a **human** and challenging for a **computer**.
- Providing a good starting point via **drag-and-drop**.

Conclusion

Make a highly **non-convex** problem tractable by:

- Introducing an interactive solution.
- Separating that which is easy for a **human** and challenging for a **computer**.
- Providing a good starting point via **drag-and-drop**.
- Providing a flexible collection of parameterized **primitives**.

Conclusion

Make a highly **non-convex** problem tractable by:

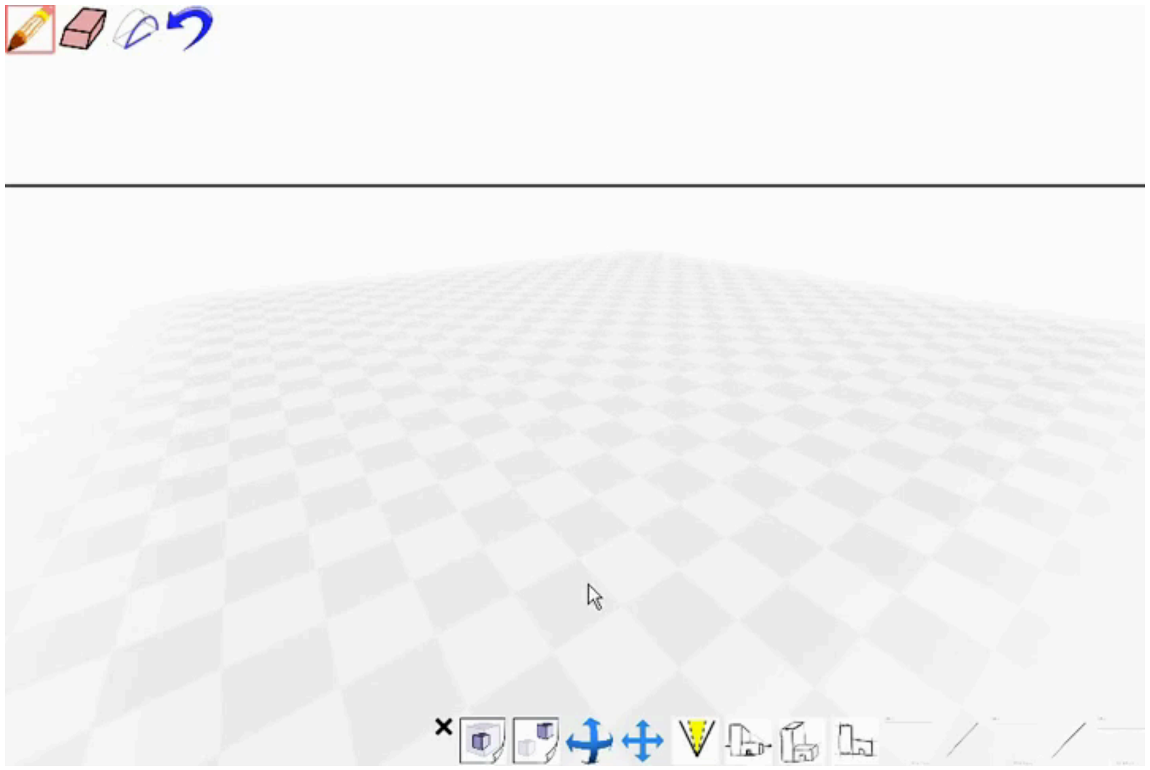
- Introducing an interactive solution.
- Separating that which is easy for a **human** and challenging for a **computer**.
- Providing a good starting point via **drag-and-drop**.
- Providing a flexible collection of parameterized **primitives**.
- Inferring **geosemantic relationships** for aligning primitives and placing them in depth.

Lifting curve networks into 3D

- Interactively
  - Analytic drawing of 3D scaffolds [Schmidt et al. 2009]
- Automatically
  - CrossShade: Shading Concept Sketches Using Cross-Section Curves [Shao et al. 2012]
  - True2Form: 3D curve networks from 2D sketches via selective regularization [Xu et al. 2014]

Analytic drawing of 3D scaffolds [Schmidt et al. 2009]

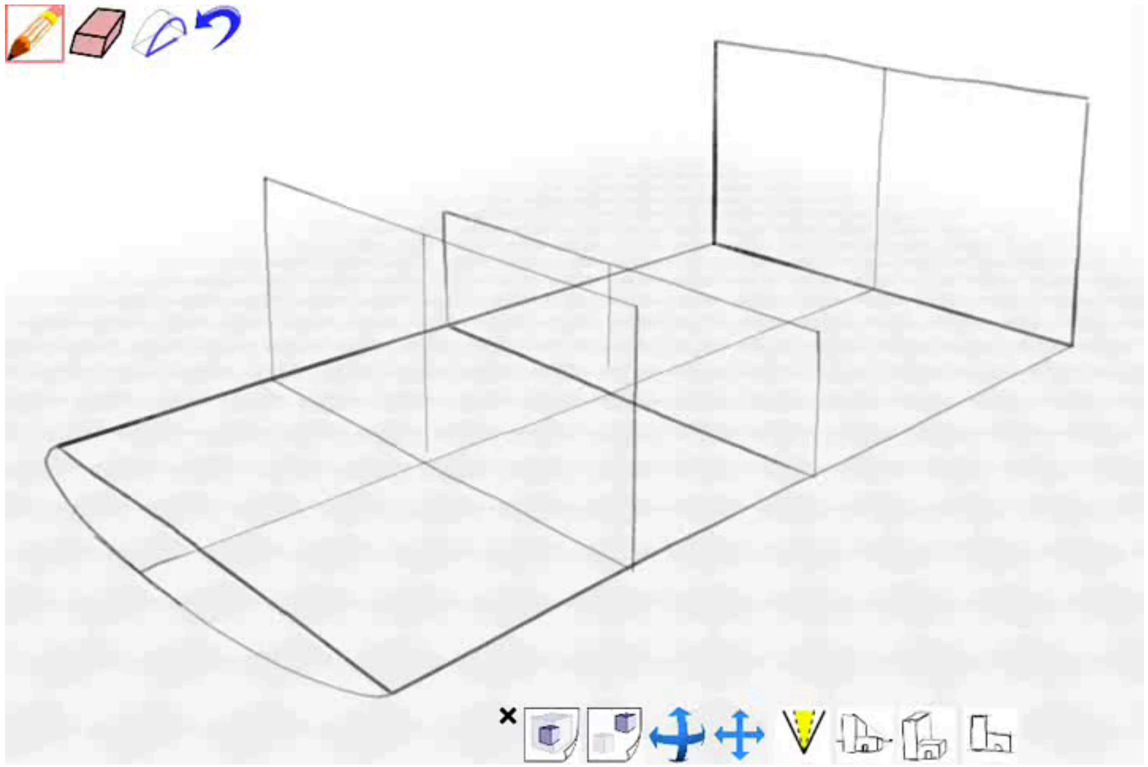
- Draw precise scaffold lines by connecting them to 2-point perspective vanishing points



EUROGRAPHICS 2016

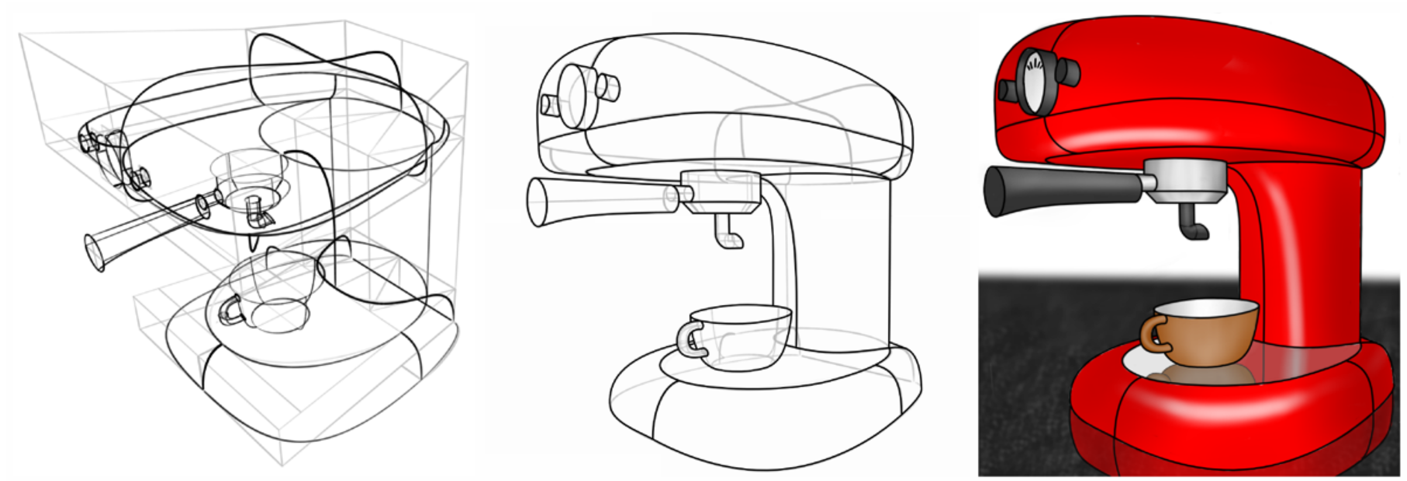
Analytic drawing of 3D scaffolds [Schmidt et al. 2009]

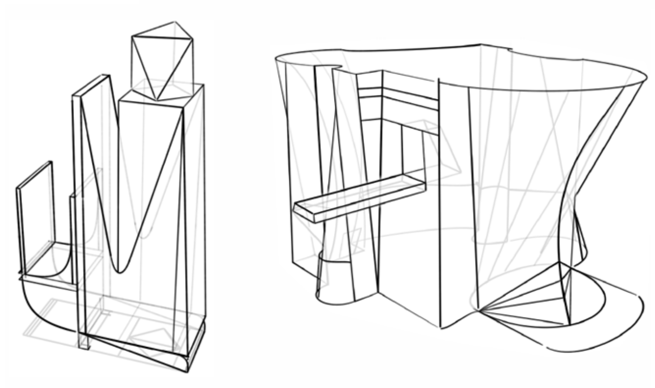
- The scaffolds make it possible to draw complex curves

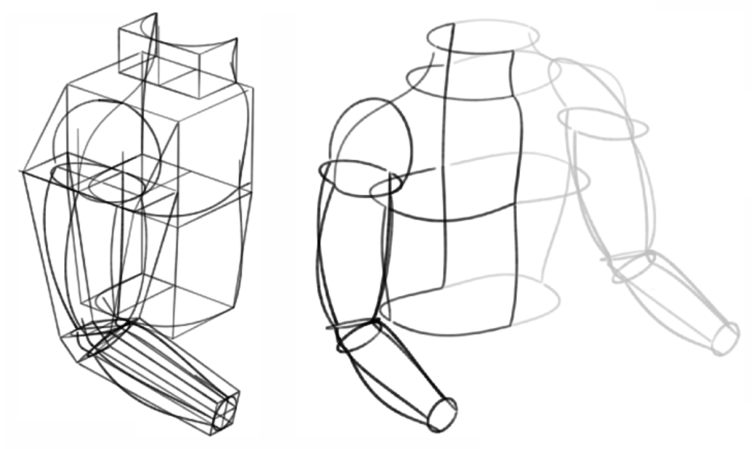


Analytic drawing of 3D scaffolds [Schmidt et al. 2009]

- … and complex shapes



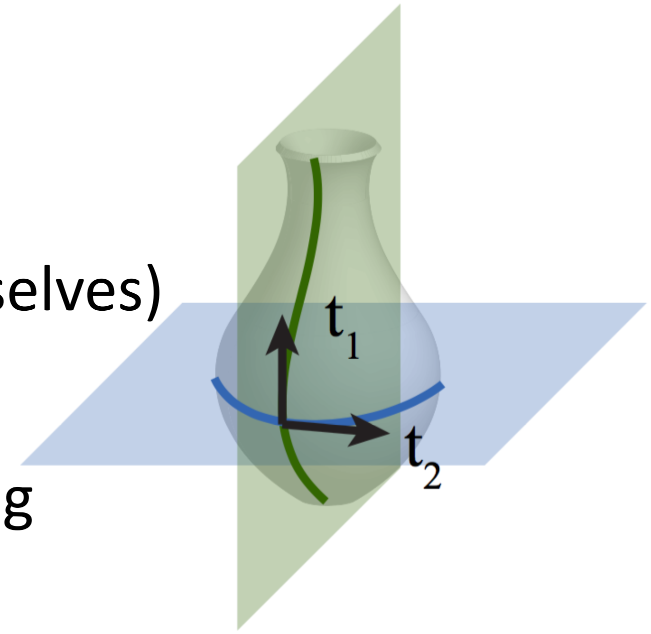




EUROGRAPHICS 2016

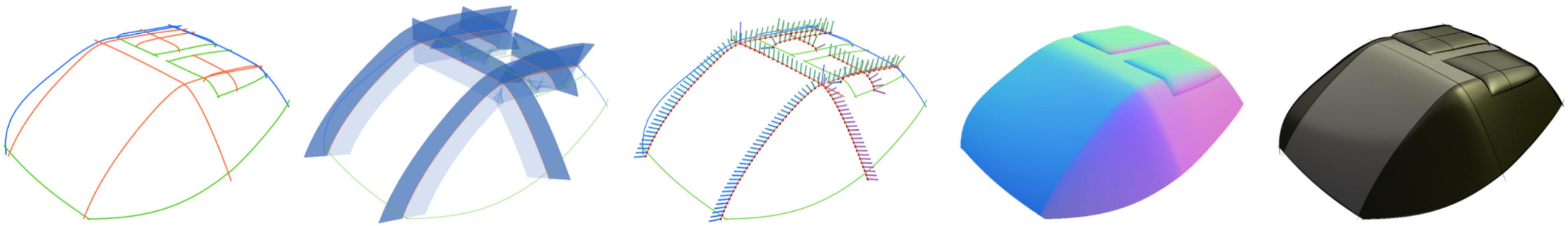
CrossShade: Shading Concept Sketches Using Cross-Section Curves [Shao et al. 2012]

- We can infer a good normal map from labeled cross section and silhouette curves via properties of designer-drawn cross sections. Cross-sections:
  - intersect on orthogonal planes
  - are aligned with principal curvature (and therefore are orthogonal themselves)
  - are geodesics
  - intersect with minimal foreshortening

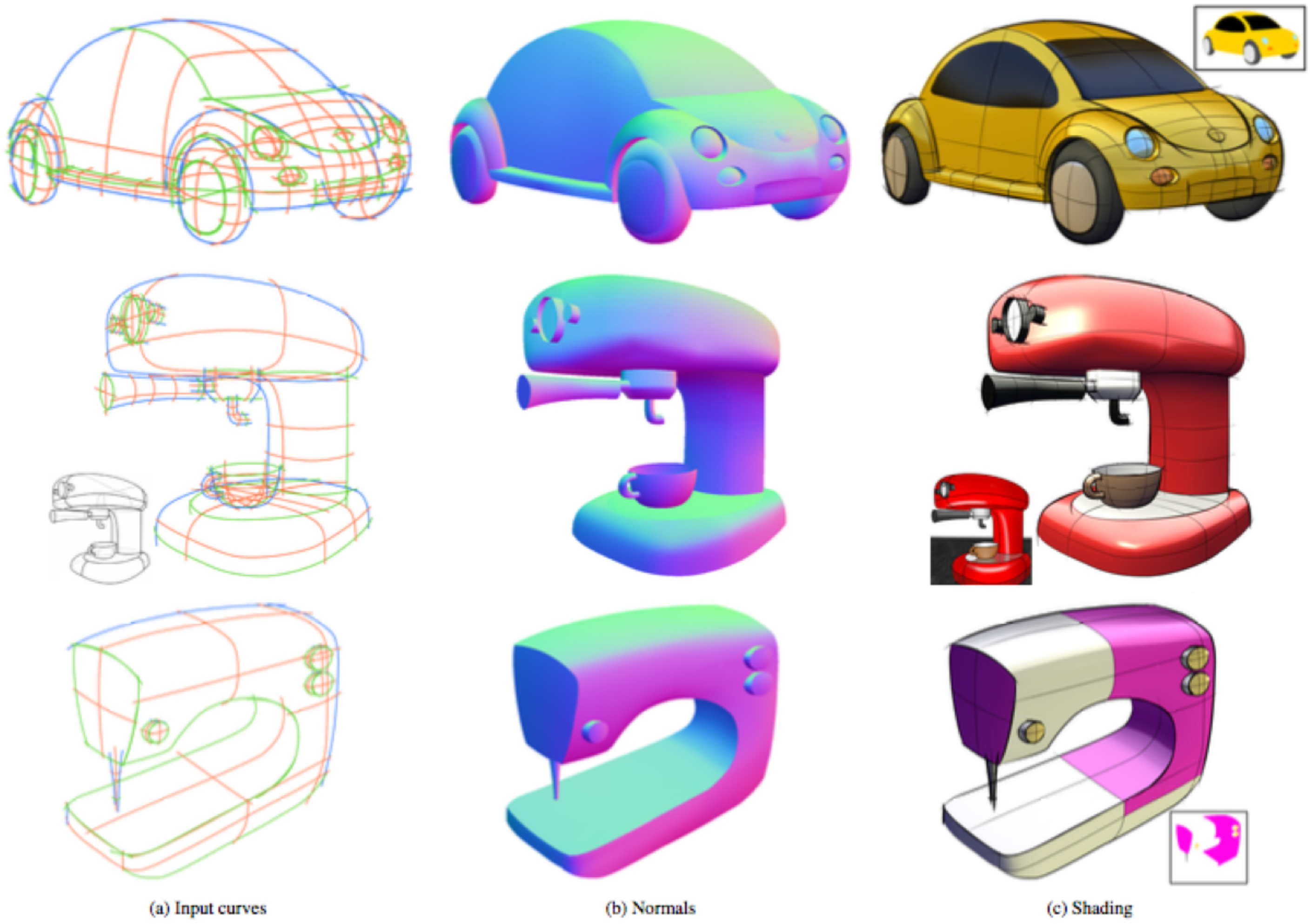


CrossShade: Shading Concept Sketches Using Cross-Section Curves [Shao et al. 2012]

- With these cues, we can propagate normals everywhere:

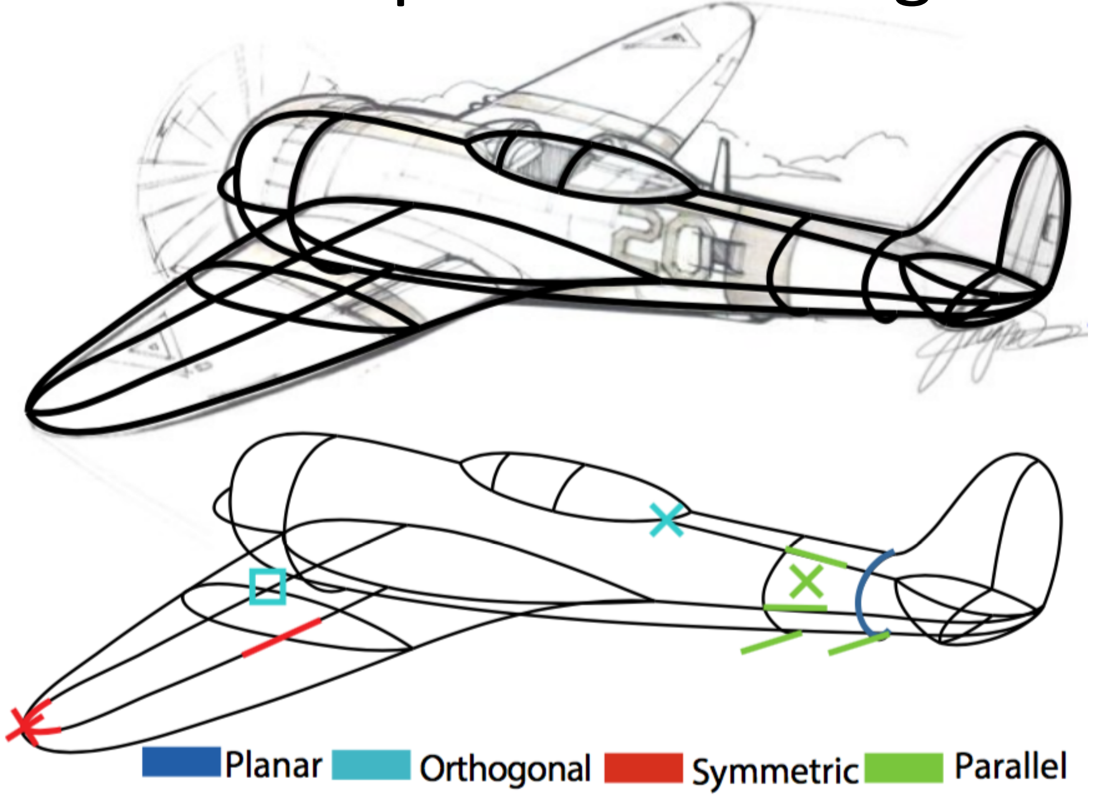

(a) Input curves (b) Estimated curve planes (c) Normals along curves (d) Normals over the sketch (e) Resulting shading

CrossShade: Shading Concept Sketches Using Cross-Section Curves [Shao et al. 2012]

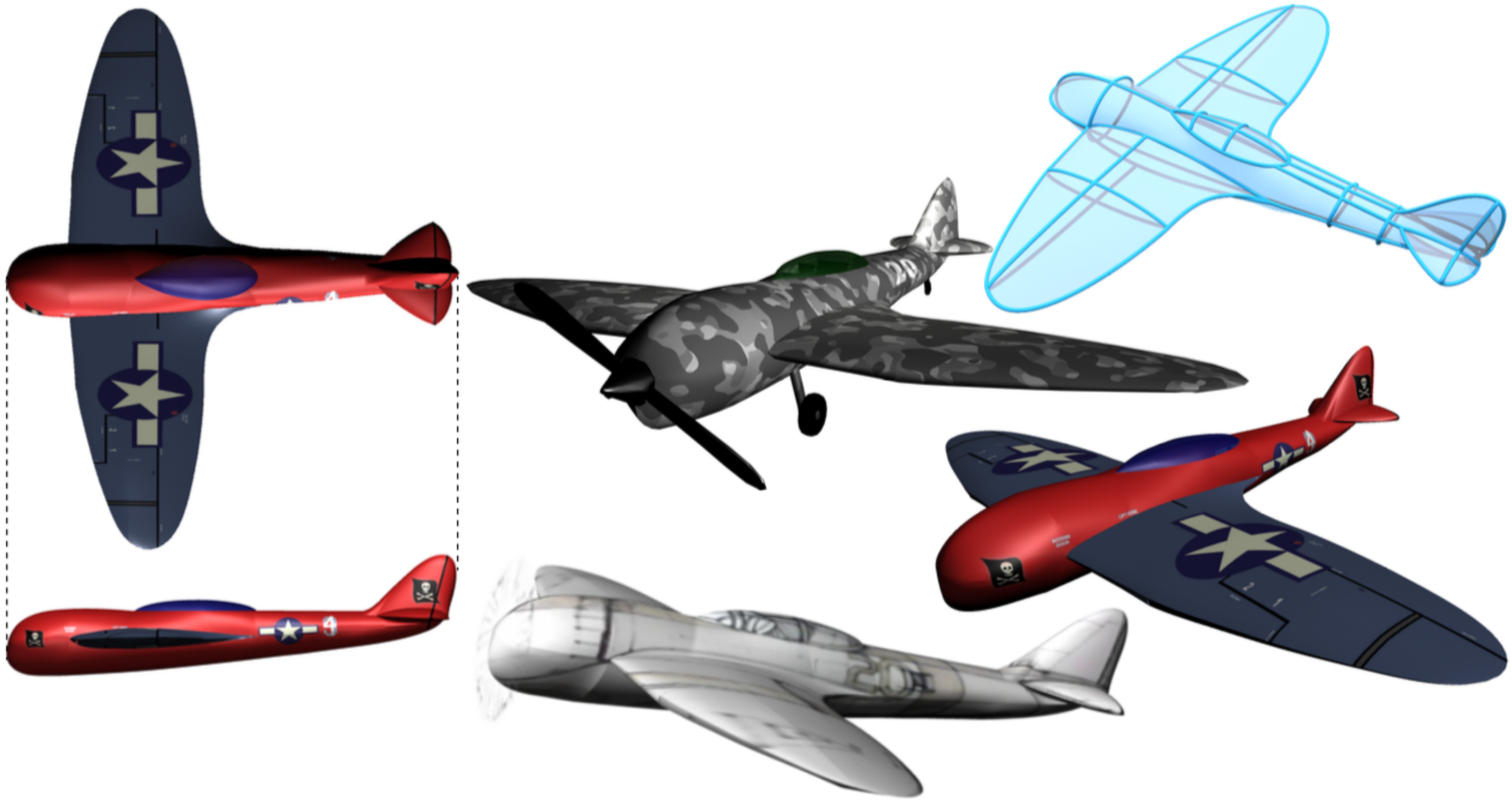


True2Form: 3D curve networks from 2D sketches via selective regularization [Xu et al. 2014]

- Given 2D curves, we can selectively apply the constraints in an optimization to get 3D curves

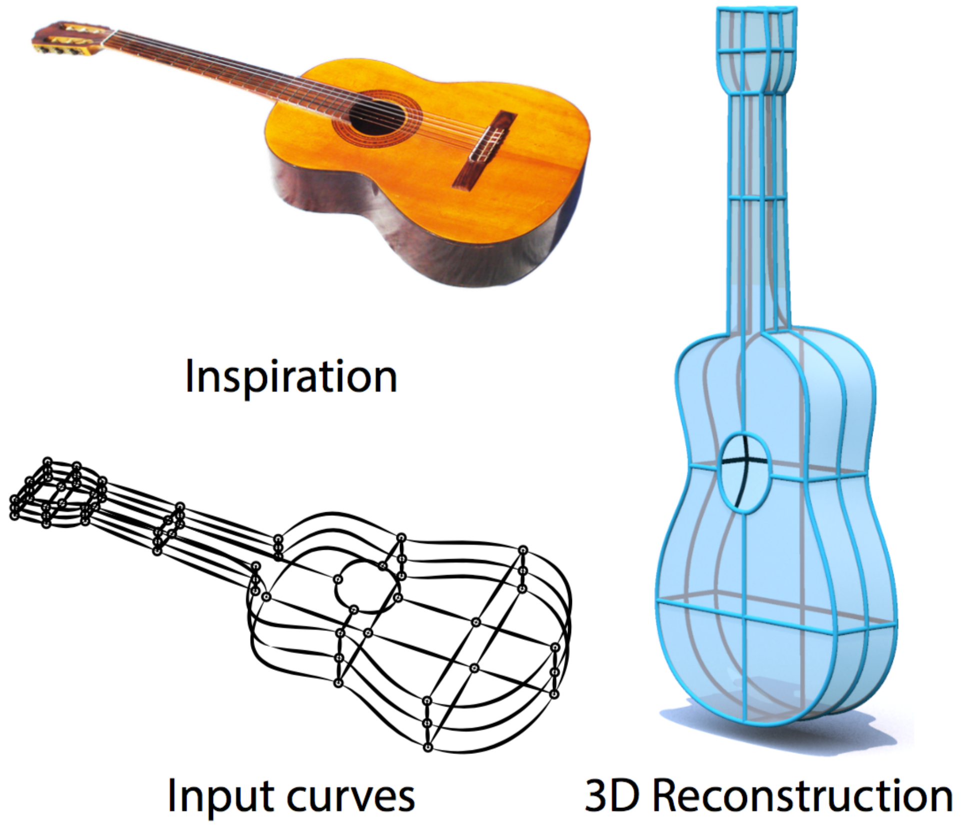


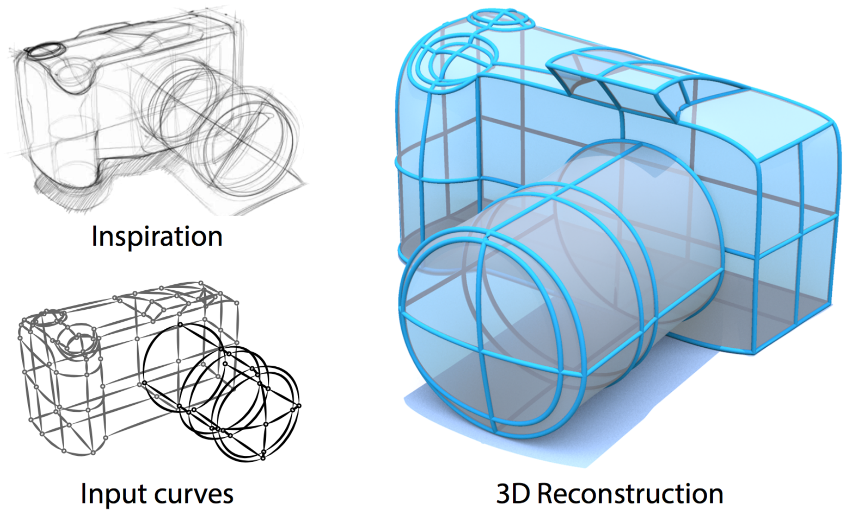
True2Form: 3D curve networks from 2D sketches via selective regularization [Xu et al. 2014]

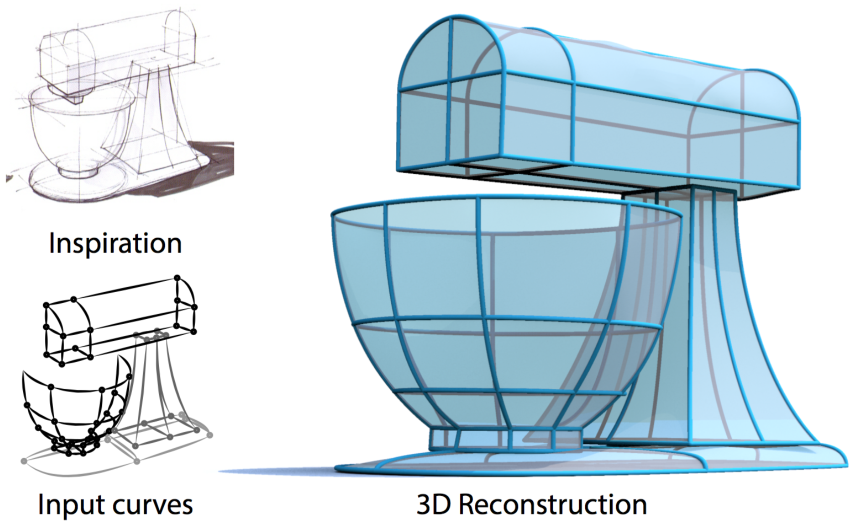


EUROGRAPHICS 2016

True2Form: 3D curve networks from 2D sketches via selective regularization [Xu et al. 2014]







EUROGRAPHICS 2016

Takeaways

- Make “intractable” problems tractable with perceptually grounded assumptions or by asking the user to help.
  - Don’t ask the user for too much. Separate that which is easy for a **human** and challenging for a **computer**.
- Consult artistic practice and perceptual psychology for inspiration.

References

[Cherlin et al 2005] Joseph Jacob Cherlin, Faramarz Samavati, Mario Costa Sousa, Joaquim A. Jorge. Sketch-based modeling with few strokes. SCCG 2005: 137-145

[Gingold et al 2009] Yotam I. Gingold, Takeo Igarashi, Denis Zorin: Structured annotations for 2D-to-3D modeling. ACM Trans. Graph. 28(5): 148:1-148:9 (2009)

[Vilppu 1997] VILPPU, G. 1997. Vilppu Drawing Manual. Vilppu Studio, Acton, California.

[Blair 1994] BLAIR, P. 1994. Cartoon Animation. Walter Foster, Laguna Hills, California

References

[Schmidt et al. 2009b] Ryan Schmidt, Azam Khan, Karan Singh, Gordon Kurtenbach: Analytic drawing of 3D scaffolds. ACM Trans. Graph. 28(5): 149:1-149:10 (2009)

[Andre and Saito 2011] Alexis Andre, Suguru Saito: Single-View Sketch Based Modeling. SBM 2011: 133-140

[Tsang et al. 2004] Steve Tsang, Ravin Balakrishnan, Karan Singh, Abhishek Ranjan: A suggestive interface for image guided 3D sketching. CHI 2004: 591-598

References

[Shtof et al. 2013] Alex Shtof, Alexander Agathos, Yotam I. Gingold, Ariel Shamir, Daniel Cohen-Or: Geosemantic Snapping for Sketch-Based Modeling. Comput. Graph. Forum 32(2): 245-253 (2013)

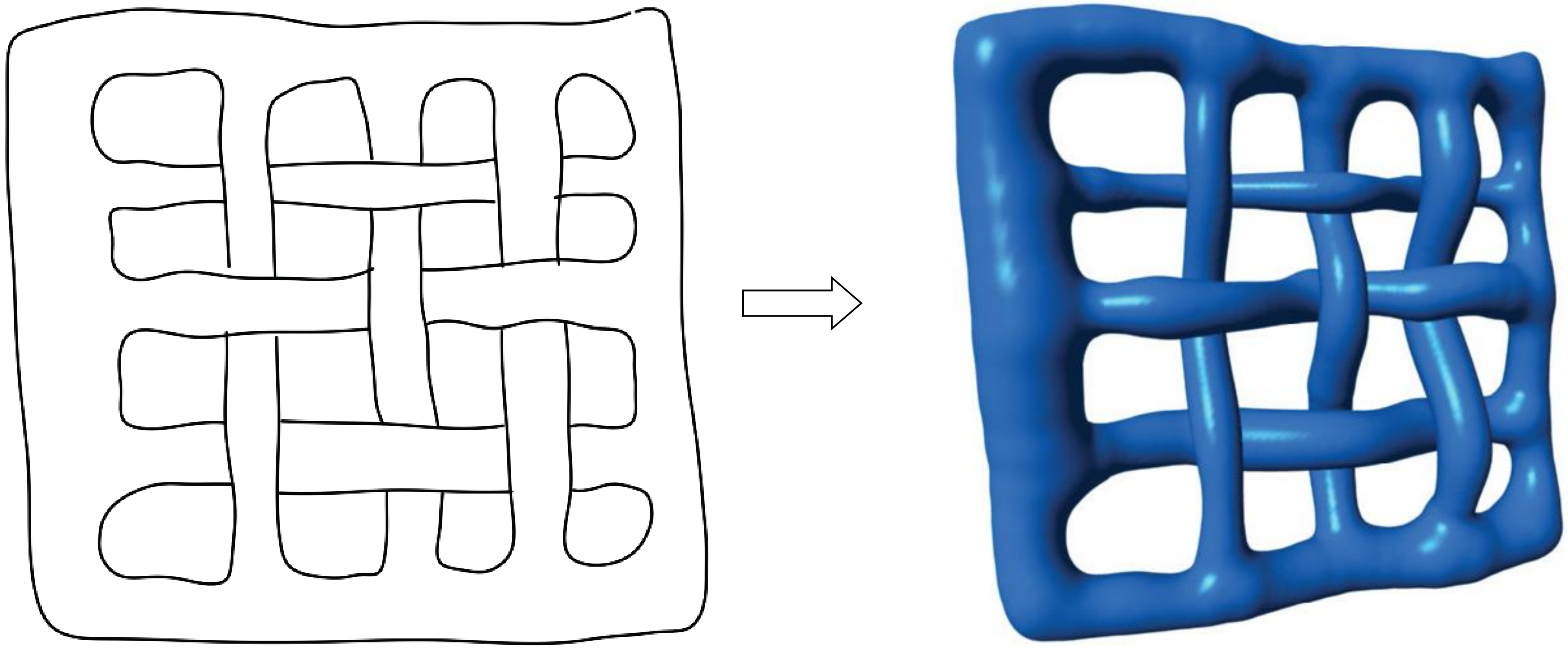
[Shao et al. 2012] Cloud Shao, Adrien Bousseau, Alla Sheffer, Karan Singh: CrossShade: shading concept sketches using cross-section curves. ACM Trans. Graph. 31(4): 45:1-45:11 (2012)

[Xu et al. 2014] Bao-Xuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, Karan Singh: True2Form: 3D curve networks from 2D sketches via selective regularization. ACM Trans. Graph. 33(4): 131:1-131:13 (2014)

Single-view sketch-based modeling of 3D curves and surfaces Part II

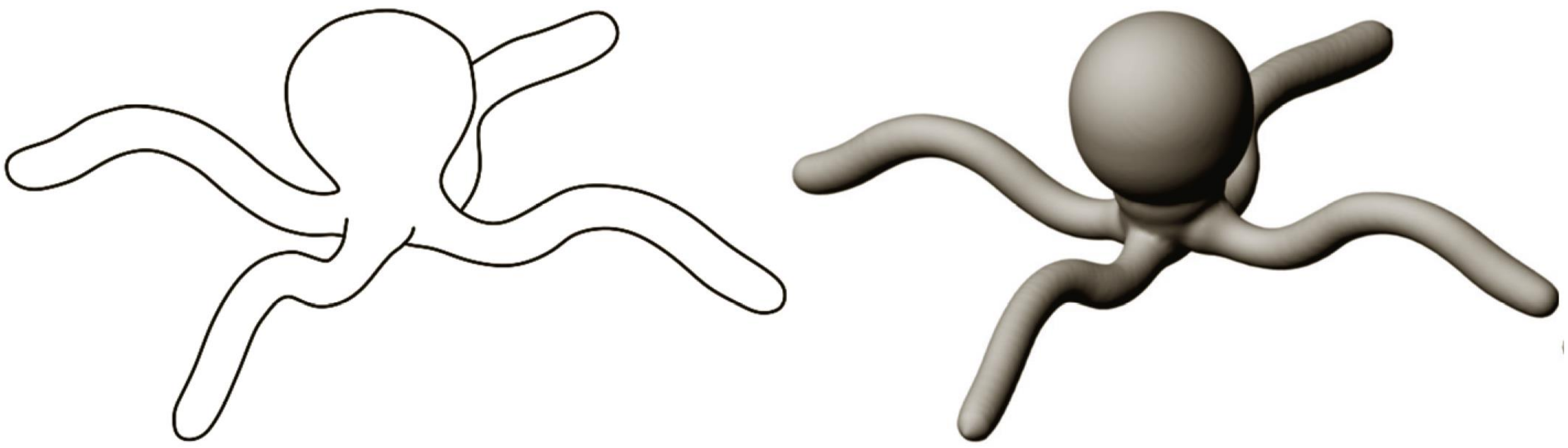
Frederic Cordier

Free-Form Sketching of Self-Occluding Objects



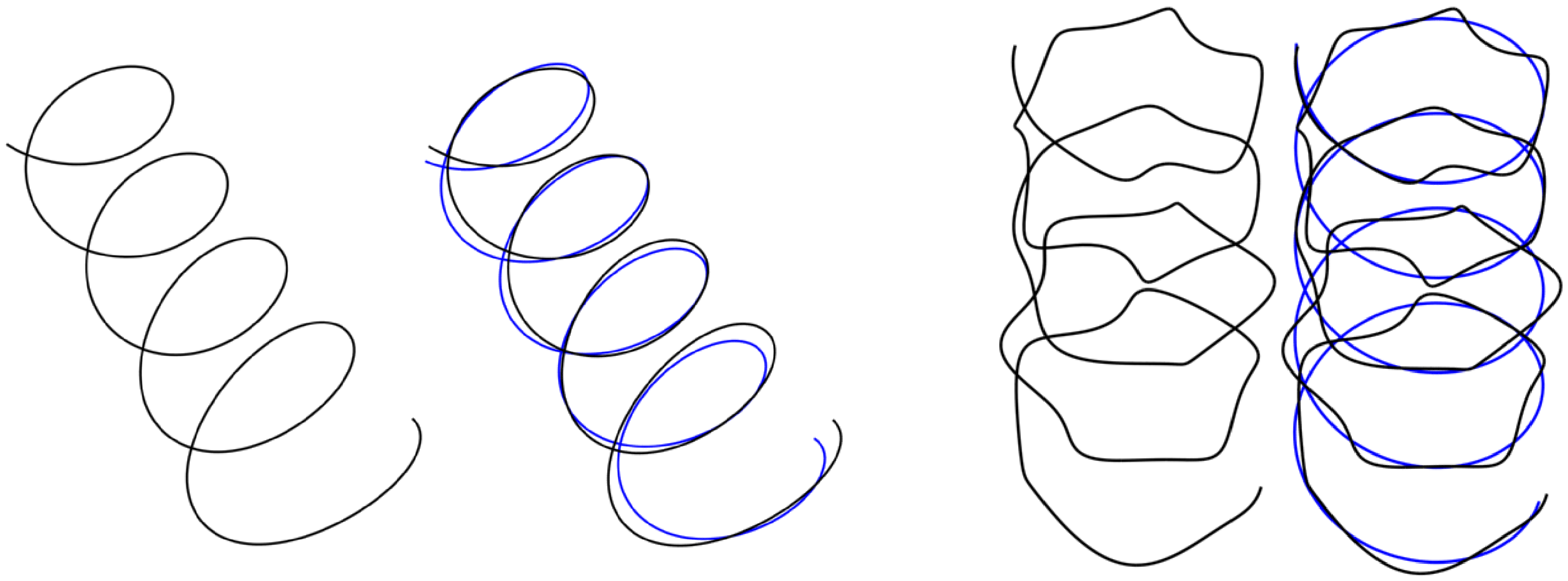
Frederic Cordier, Hyewon Seo: Free-Form Sketching of Self-Occluding Objects. IEEE Computer Graphics and Applications 27(1): 50-59 (2007)

Sketching of Mirror-Symmetric Shapes



Frederic Cordier, Hyewon Seo, Jinho Park, Jun-yong Noh: Sketching of Mirror-Symmetric Shapes. IEEE Trans. Vis. Comput. Graph. 17(11): 1650-1662 (2011)

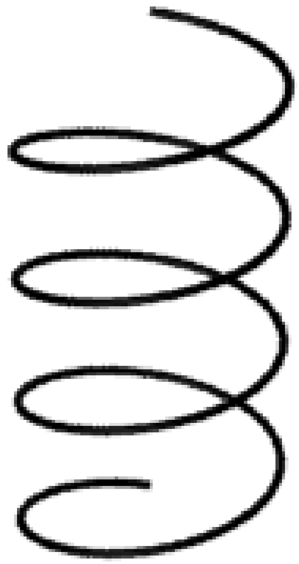
Reconstruction of helices from their orthogonal projection



Frederic Cordier, Mahmoud Melkemi, Hyewon Seo: Reconstruction of helices from their orthogonal projection. Computer Aided Geometric Design. In press.

Reconstruction of helices from their orthogonal projection

- Reconstruction of curves of constant curvature

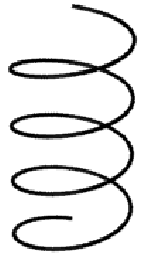


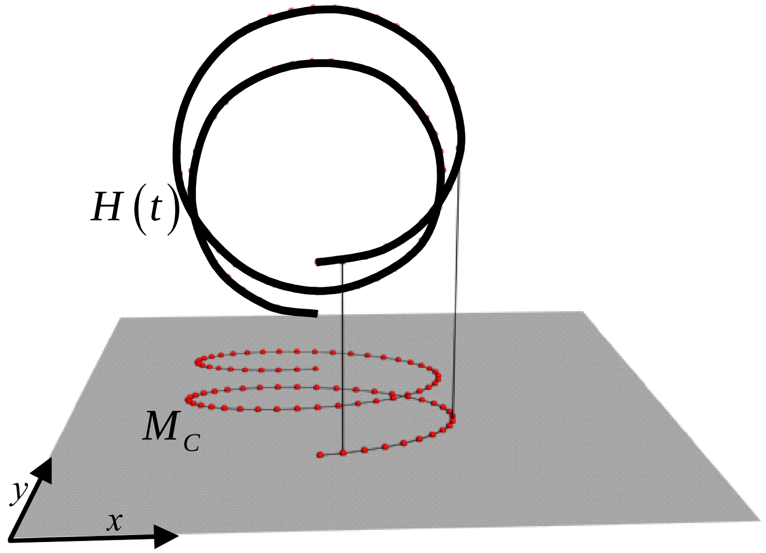
EUROGRAPHICS 2016

Reconstruction of helices from their orthogonal projection

Parametric equation of a helix of radius *r* and pitch *p* :

$$H(t) = \begin{bmatrix} r\cos(t) \\ pt \\ r\sin(t) \end{bmatrix}$$





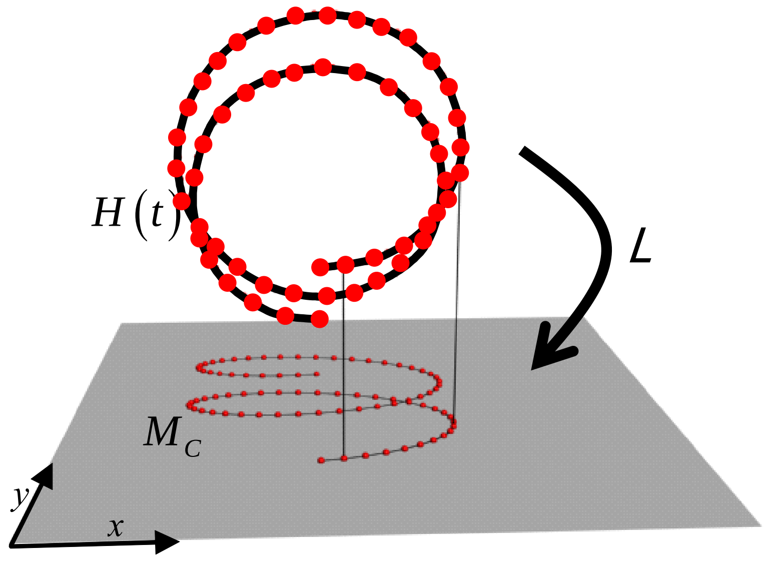
Computing *r*, *p* and the projection matrix requires non-linear optimization !

Reconstruction of helices from their orthogonal projection

Sampling of the helix

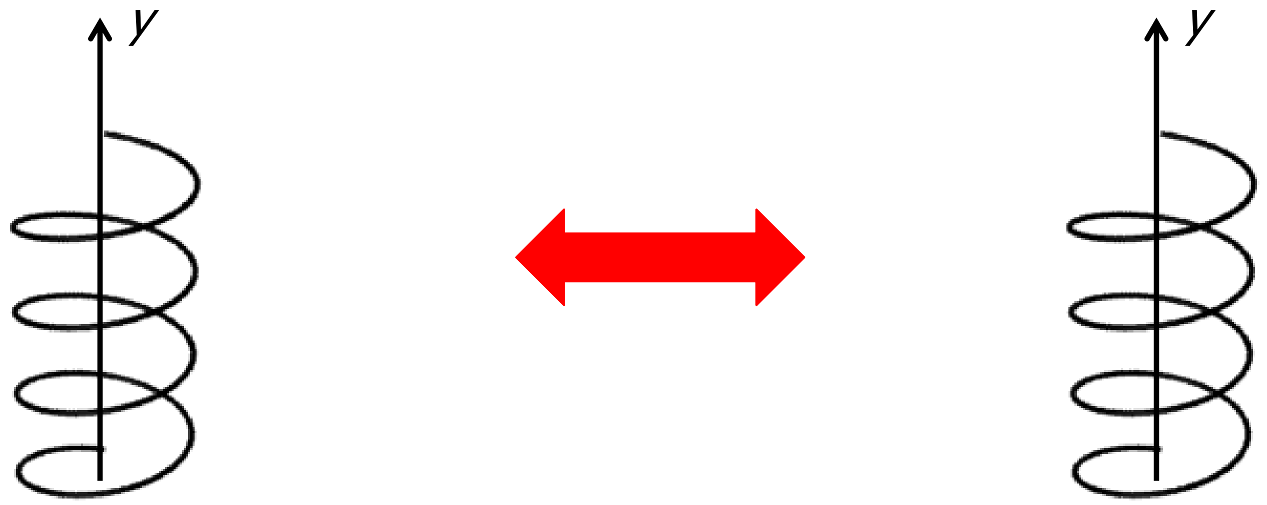
Compute the affine transformation *L*

Compute the rotation matrix and the helix parameters



Reconstruction of helices from their orthogonal projection

The key idea:



Pitch equal to 1

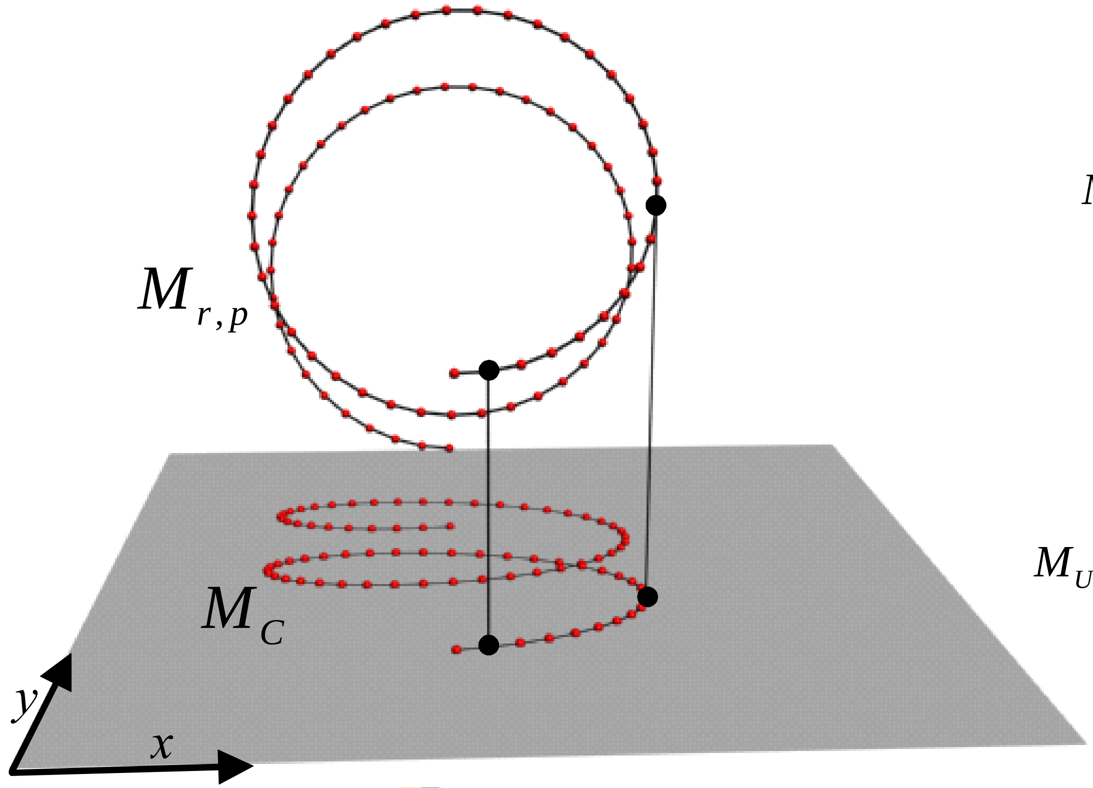
Scale along *y* equal to 2

Pitch equal to 2

Scale along *y* equal to 1

Reconstruction of helices from their orthogonal projection

Sampling of the helix



$$M_{\mathrm{r},p} = \begin{bmatrix} r\cos(t_1) & pt_1 & r\sin(t_1) \\ r\cos(t_2) & pt_2 & r\sin(t_2) \\ \vdots & \vdots & \vdots \\ r\cos(t_n) & pt_n & r\sin(t_n) \end{bmatrix} = M_U S_{rp}$$

$$M_U = \begin{bmatrix} \cos(t_1) & t_1 & \sin(t_1) \\ \cos(t_2) & t_2 & \sin(t_2) \\ \vdots & \vdots & \vdots \\ \cos(t_n) & t_n & \sin(t_n) \end{bmatrix} \quad S_{rp} = \begin{bmatrix} r & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & r \end{bmatrix}$$

EUROGRAPHICS 2016

Reconstruction of helices from their orthogonal projection

$$\min_L \left\| M_U L - M_C \right\|_F^2$$

Affine transformation
- Rotation
- Shear
- Scale…

$$\left\| M_U L - M_C \right\|_F^2 = \left\| \left(M_U S_{rp}\right) \underbrace{\left(S_{rp}^{-1} L\right)} - M_C \right\|_F^2$$

Should be close to orthonormal
(i.e. rotation matrix)

Key idea: changing the scaling transformation of the helix is equivalent to changing its radius and pitch

Reconstruction of helices from their orthogonal projection

$\left(S_{rp}^{-1}L\right)$ is a matrix with orthonormal columns if

$$\left(S_{rp}^{-1}L\right)^T\left(S_{rp}^{-1}L\right) = I$$

We solve

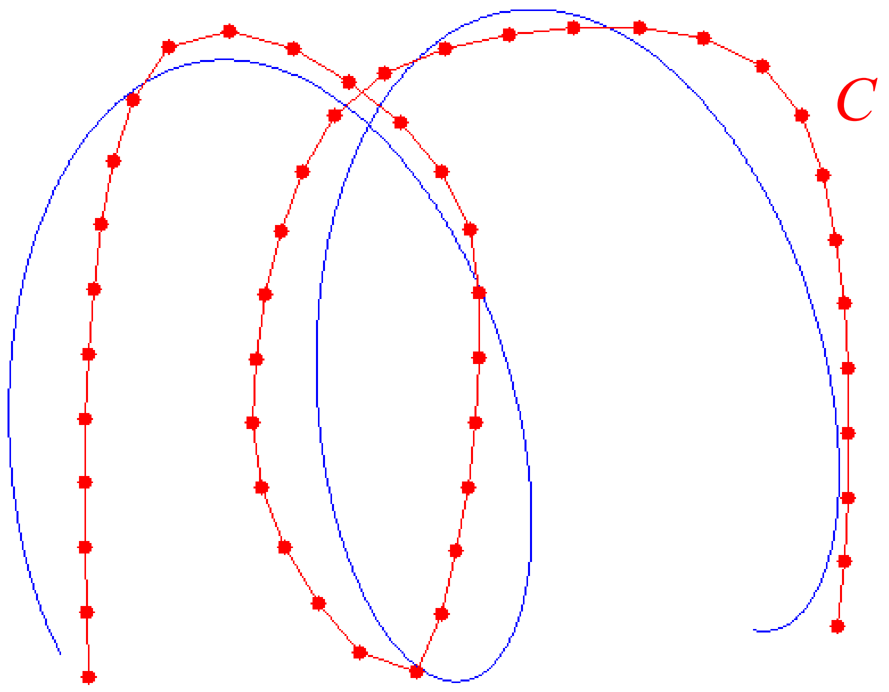
$$\min_{r,p}\left\|\left(S_{rp}^{-1}L\right)^T\left(S_{rp}^{-1}L\right) - I\right\|_F^2$$

Reconstruction of helices from their orthogonal projection

Advantages:

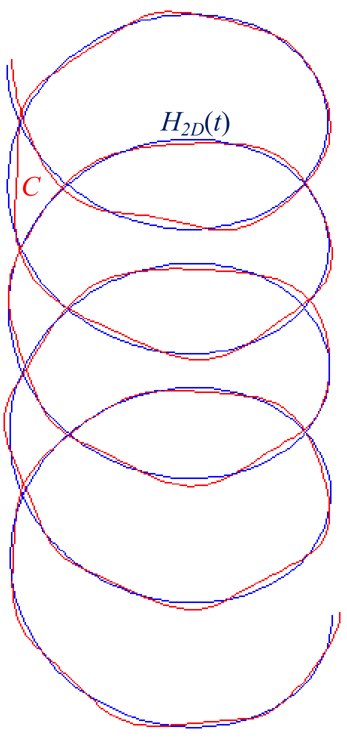
- Method that requires solving simple linear systems
- Much faster than using non-linear optimization
- Provides an approximate solution which is very close to the exact solution

Reconstruction of helices from their orthogonal projection

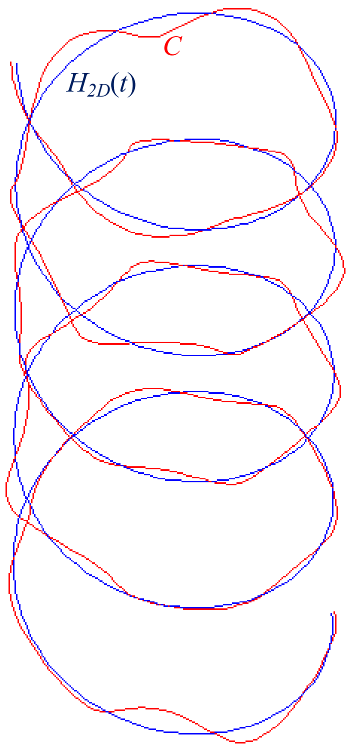


EUROGRAPHICS 2016

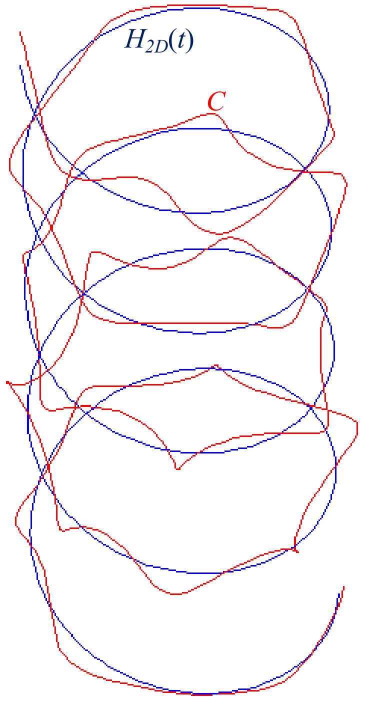
Reconstruction of helices from their orthogonal projection



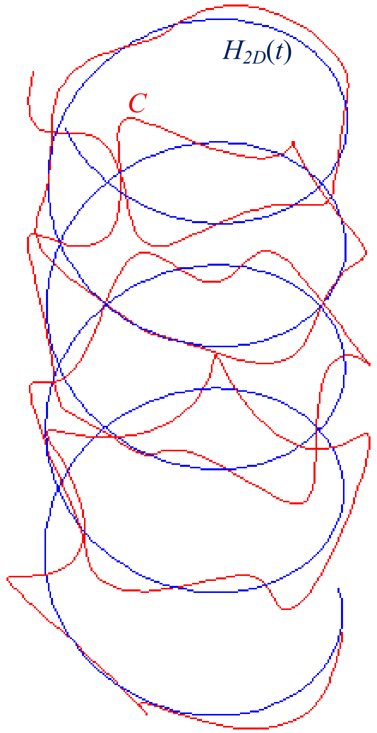
Magnitude of the noise: 0.5
r=4.0076
p=0.9879
α=14.1372
Fitting error: 0.2586%



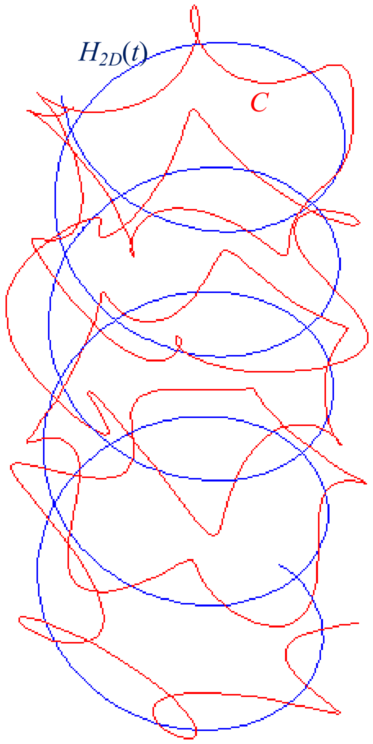
Magnitude of the noise: 1.0
r=4.0055
p=0.9992
α=14.1372
Fitting error: 0.6561%



Magnitude of the noise: 1.5
r=3.9417
p=0.9603
α=14.0744
Fitting error: 0.7880%



Magnitude of the noise: 2.0
r=3.7610
p=1.0684
α=14.0115
Fitting error: 1.0579%

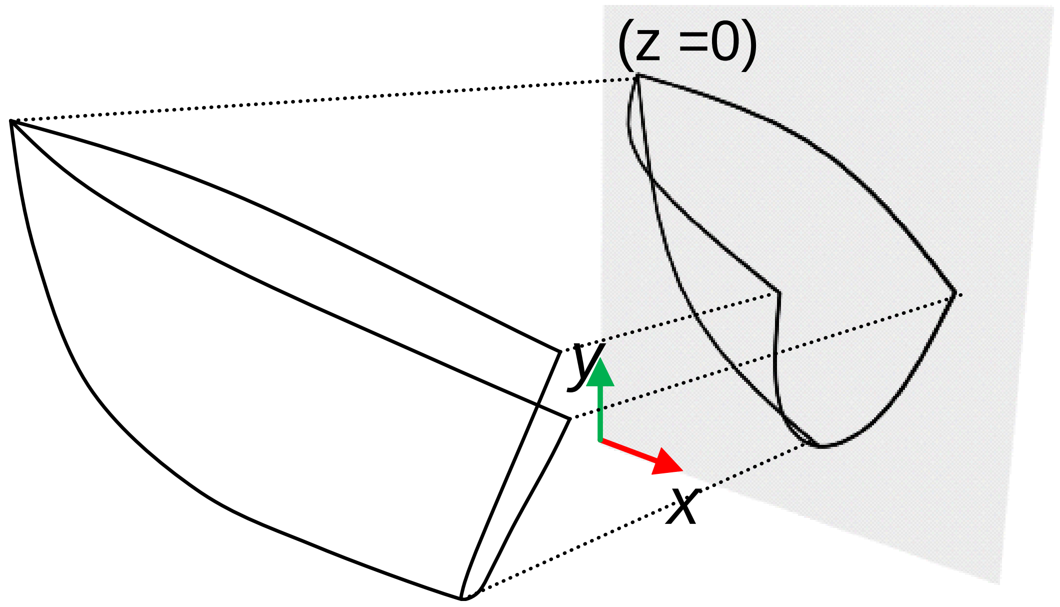


Magnitude of the noise: 3.0
r=3.8014
p=1.0714
α=14.4514
Fitting error: 2.6036%

EUROGRAPHICS 2016

Inferring mirror symmetric 3D curves from sketches

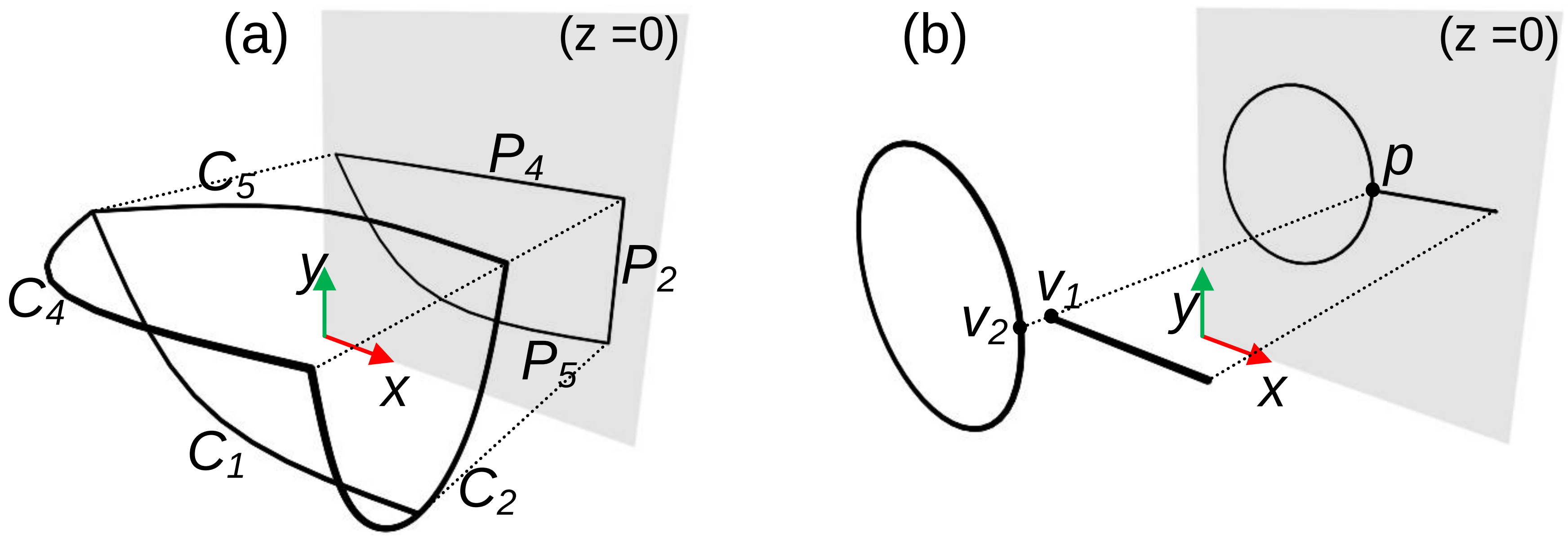
- Input: the 2D sketch of a mirror-symmetric 3D shape
- Output: a set of 3D curves such that their orthogonal projection matches the input sketch



EUROGRAPHICS

Inferring mirror symmetric 3D curves from sketches

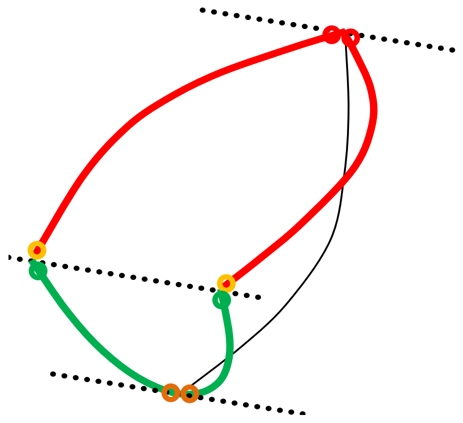
- Assumptions:
- Mirror-symmetric shape composed of curves
- Orthogonal projection



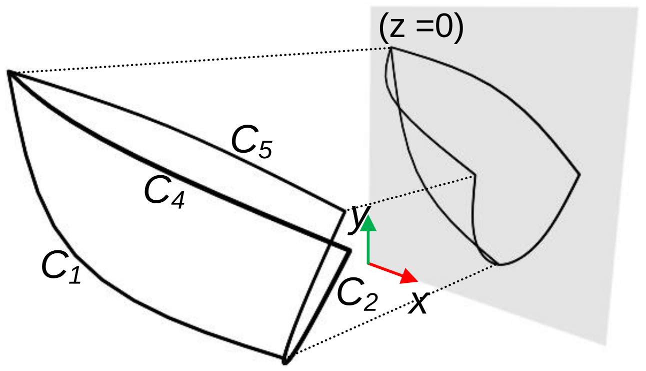
Inferring mirror symmetric 3D curves from sketches

Overview

- Finding pairs of symmetric curves:



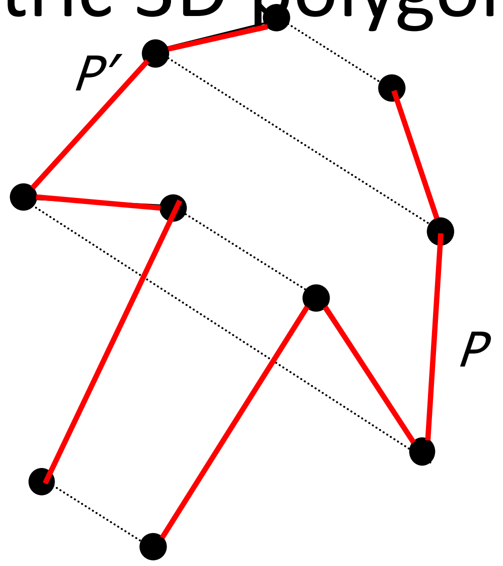
- 3D reconstruction:



Inferring mirror symmetric 3D curves from sketches

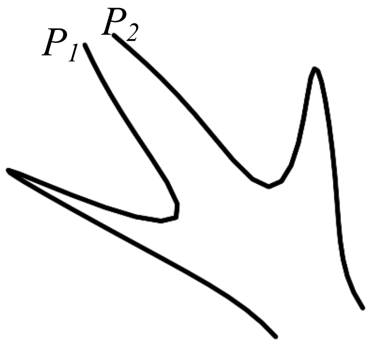
Properties of symmetric polygons

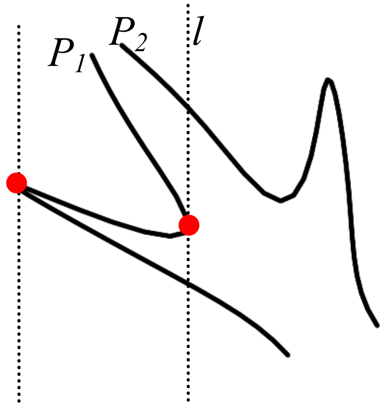
- P and P’ are the orthogonal projections of a pair of symmetric 3D polygonal curves:

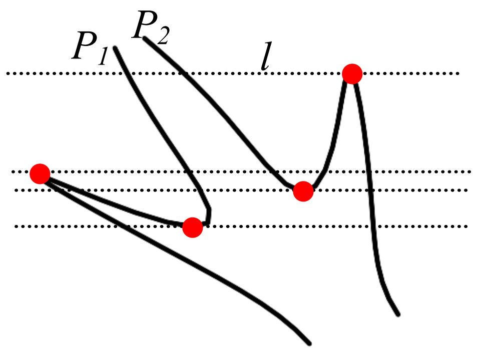


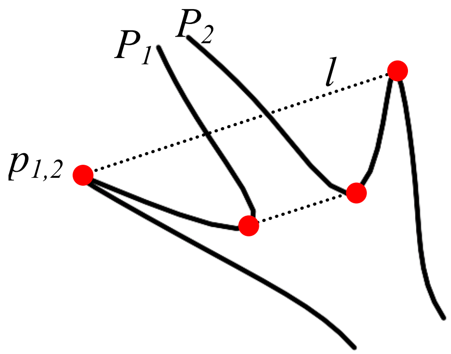
Inferring mirror symmetric 3D curves from sketches

How to find that P1 is symmetric to P2?



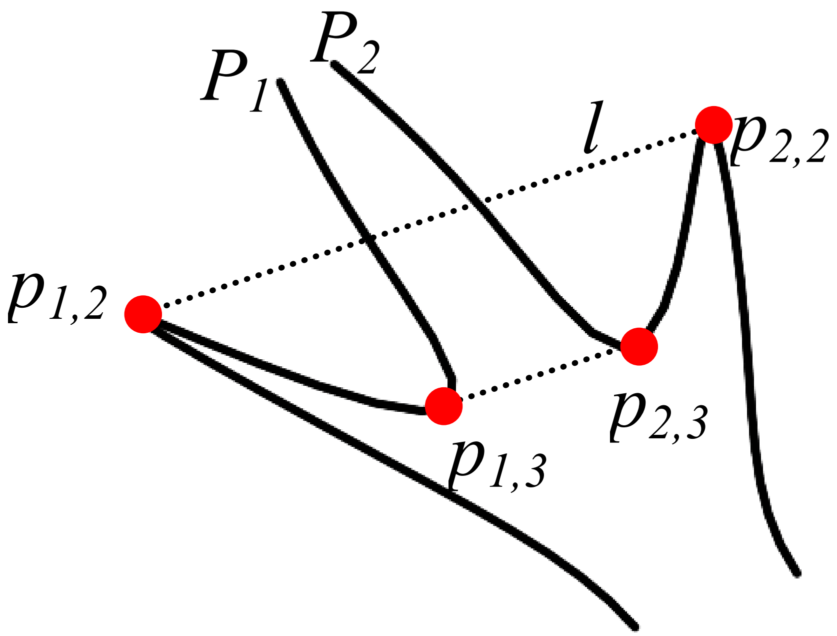






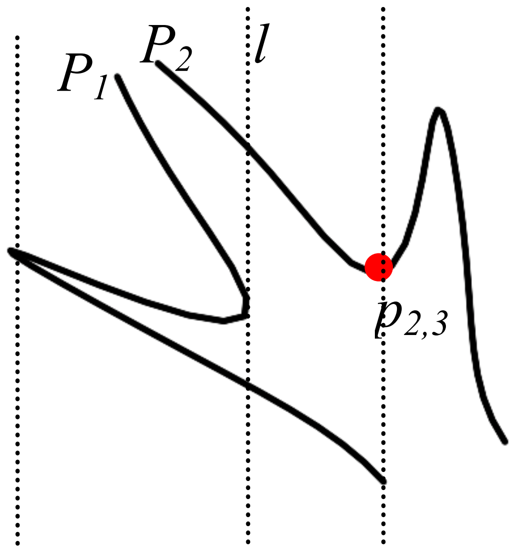
Inferring mirror symmetric 3D curves from sketches

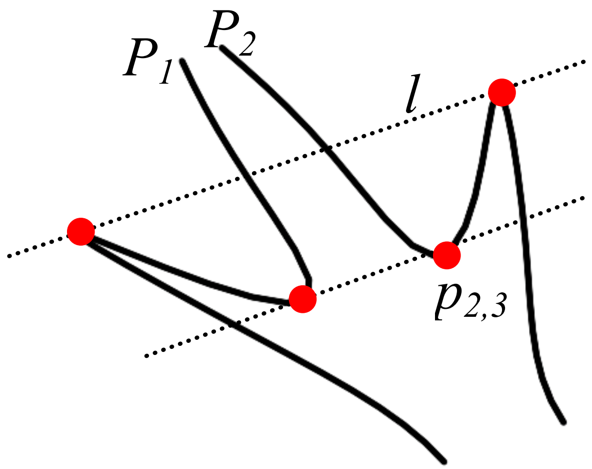
- A turn vertex is a vertex such that the two adjacent vertices are located in the same half-plane delimited by l.



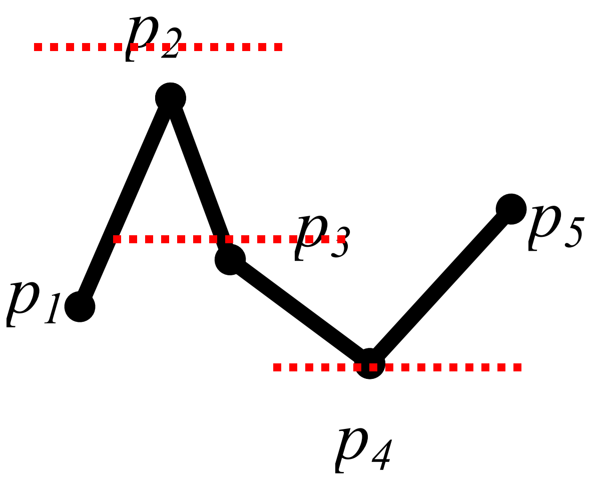
Inferring mirror symmetric 3D curves from sketches

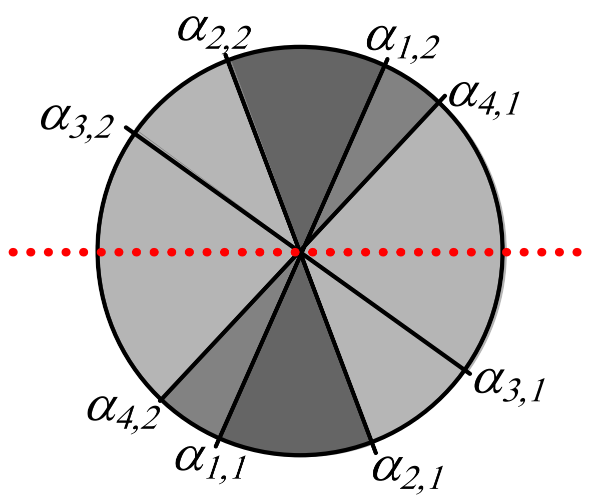
- A turn vertex is a vertex such that the two adjacent vertices are located in the same half-plane delimited by l.

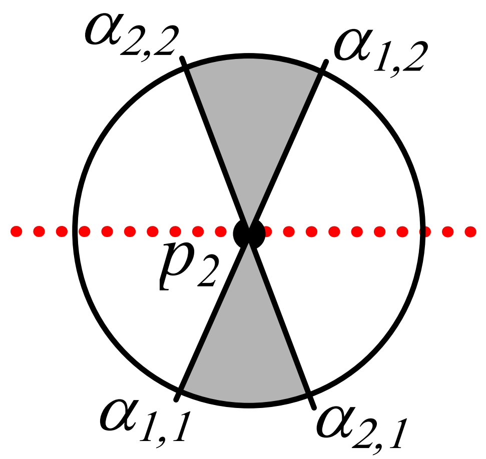


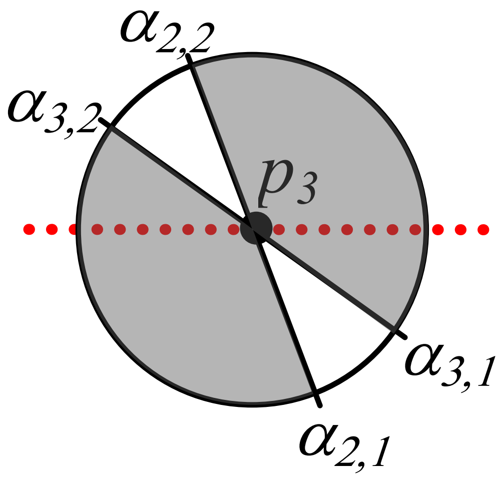


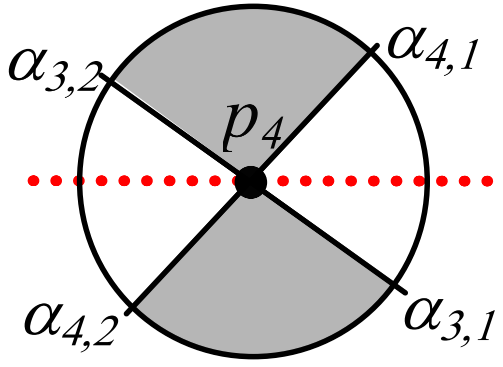
Inferring mirror symmetric 3D curves from sketches





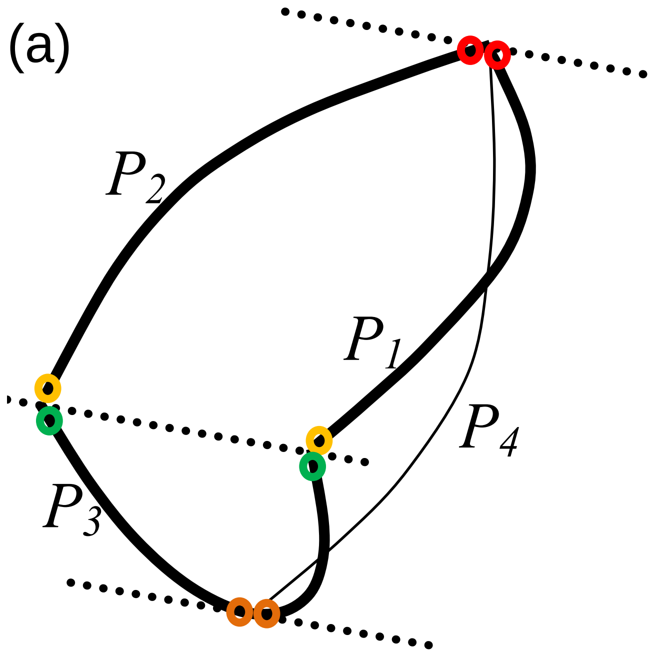


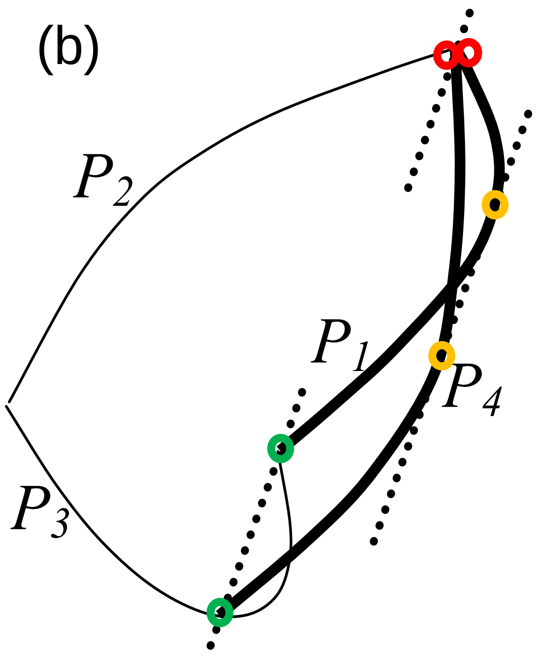


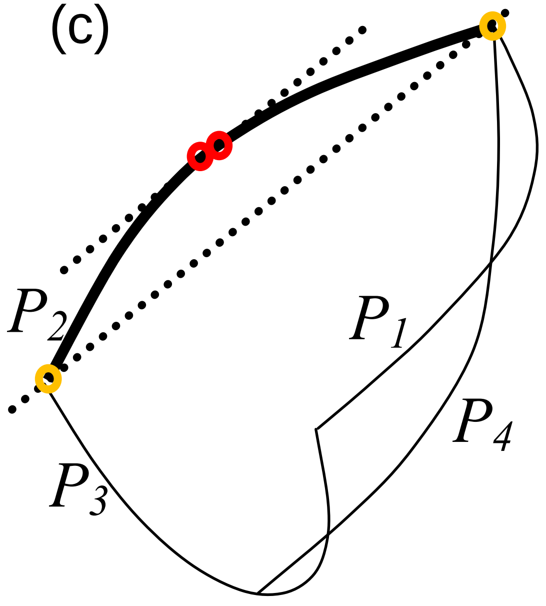


Inferring mirror symmetric 3D curves from sketches

Finding the symmetric curves







P_1 and P_2 symmetric
P_3 self-symmetric
P_4 non-symmetric

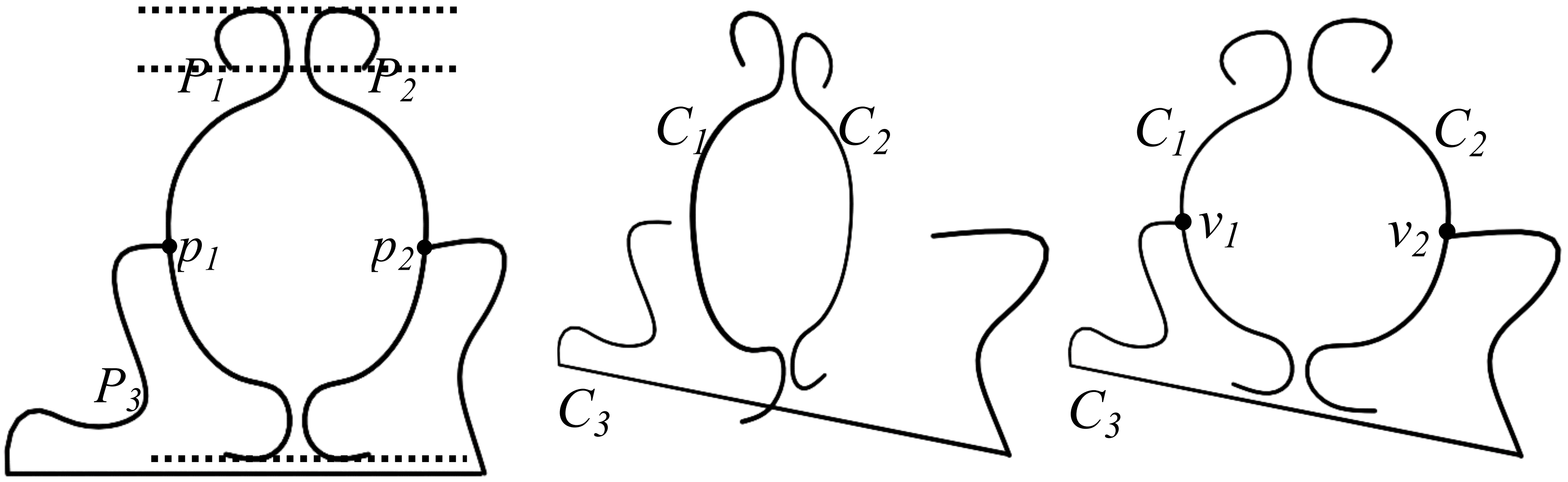
P_1 and P_4 symmetric
P_3 and P_2 non-symmetric

P_2 self-symmetric
P_1, P_3 and P_4 non-symmetric

Inferring mirror symmetric 3D curves from sketches

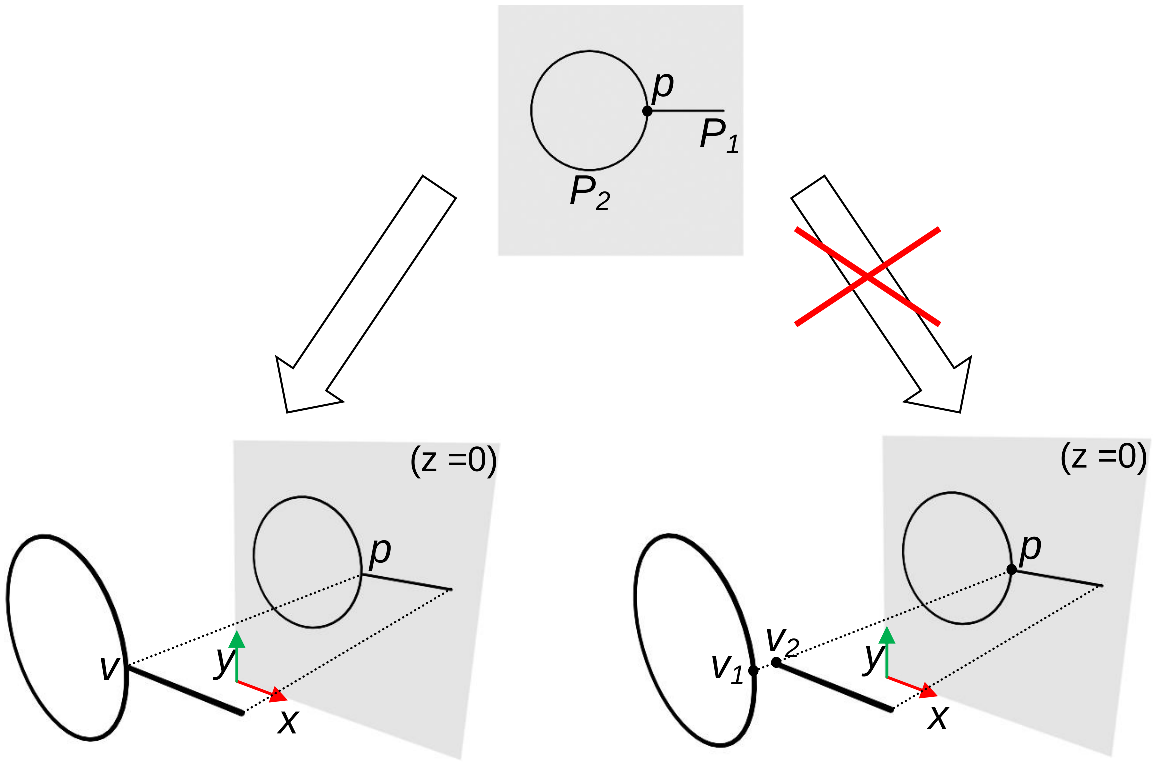
Computing the symmetry relationship

- Unnatural 3D reconstruction



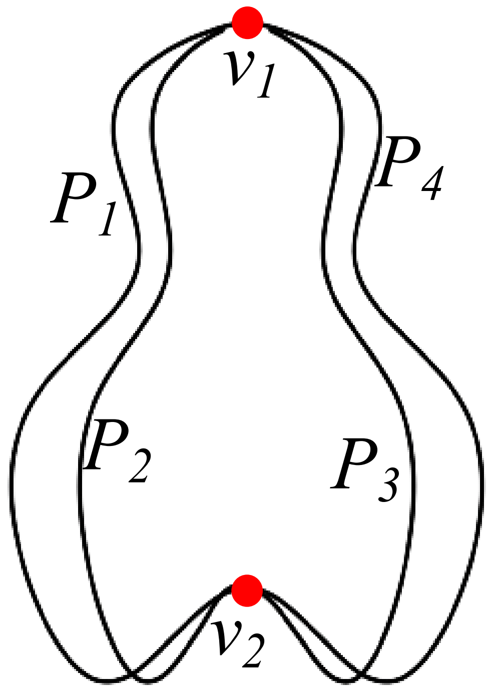
Inferring mirror symmetric 3D curves from sketches

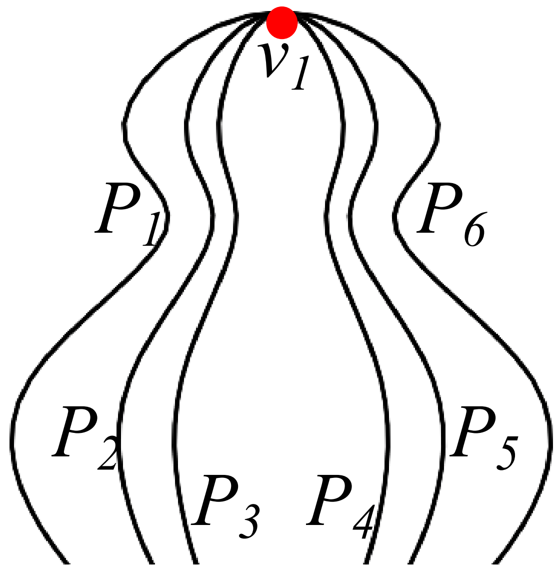
- Exploiting the curve connectivity and the generic viewpoint assumption

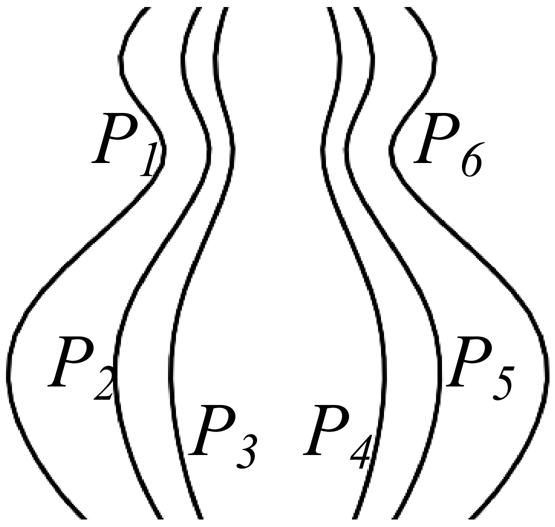


Inferring mirror symmetric 3D curves from sketches

- The curve connectivity is not sufficient to uniquely define the symmetry relationship.



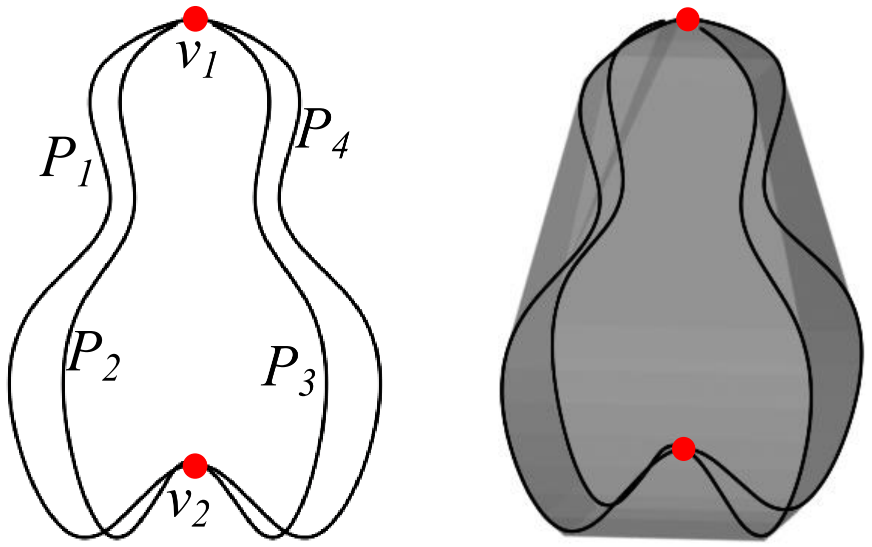




Inferring mirror symmetric 3D curves from sketches

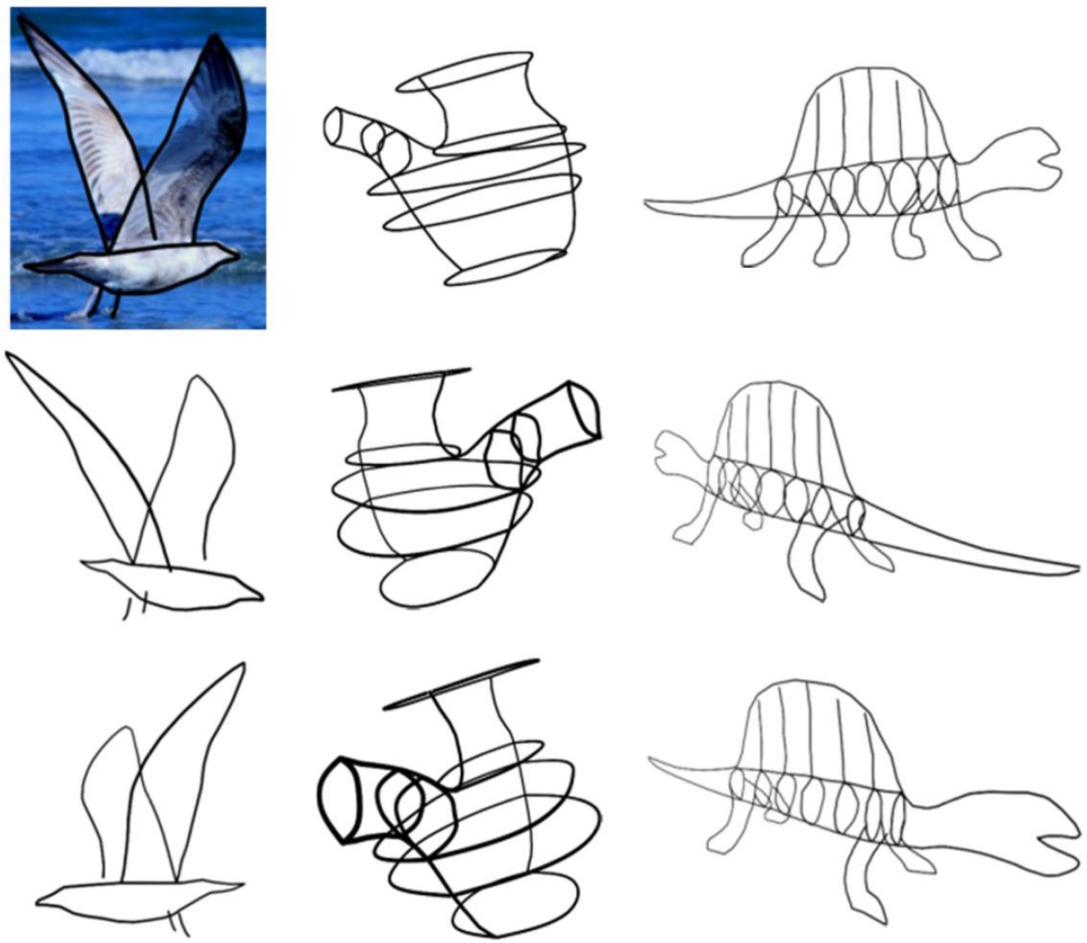
Maximizing the compactness of the reconstructed curves:

$$C(O) = \frac{V(O)^2}{S(O)^3}$$



Li Y, Pizlo Z, Steinman RM. A computational model that recovers the 3D shape of an object from a single 2D retinal representation. Vision Research. 2009; 49(9):979–91. 33 34

Inferring mirror symmetric 3D curves from sketches



Sketch-based editing

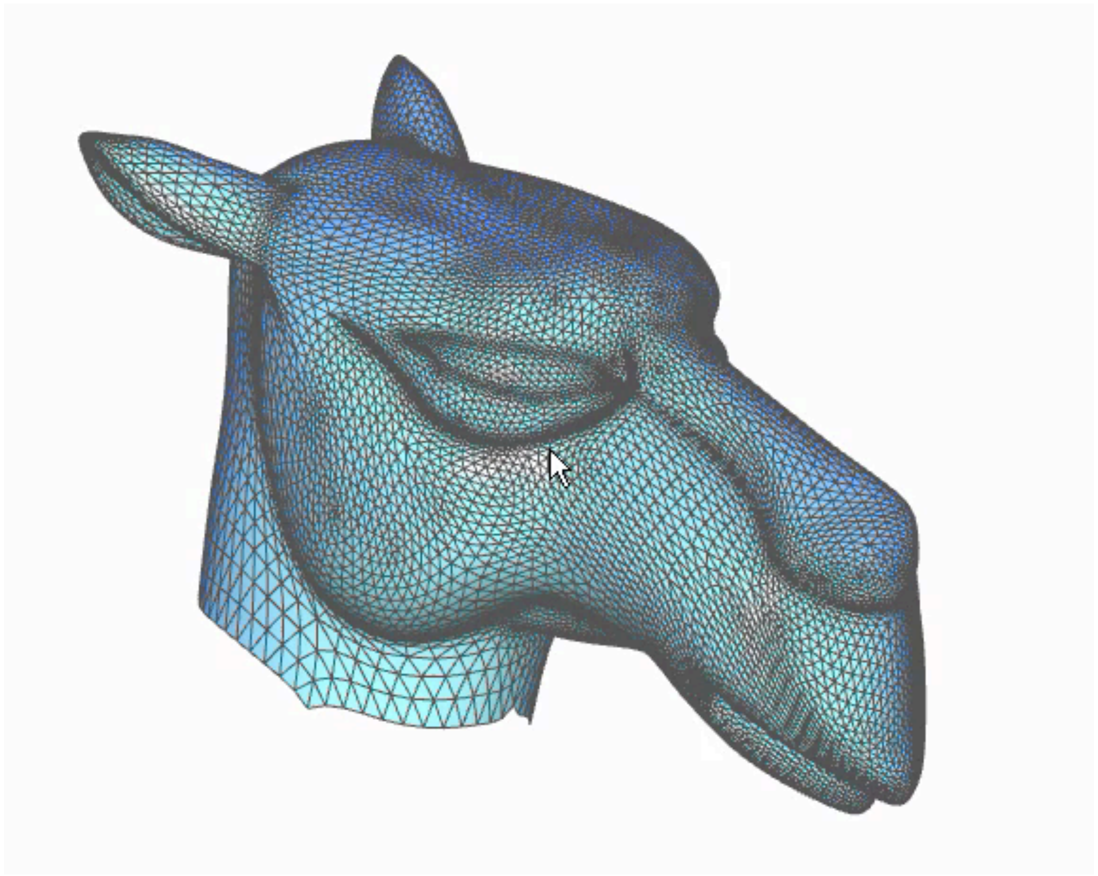
Yotam Gingold

Editing operations

- Cutting (we saw earlier)
- Deform by sketching new silhouettes
- Edit relief by sketching shading

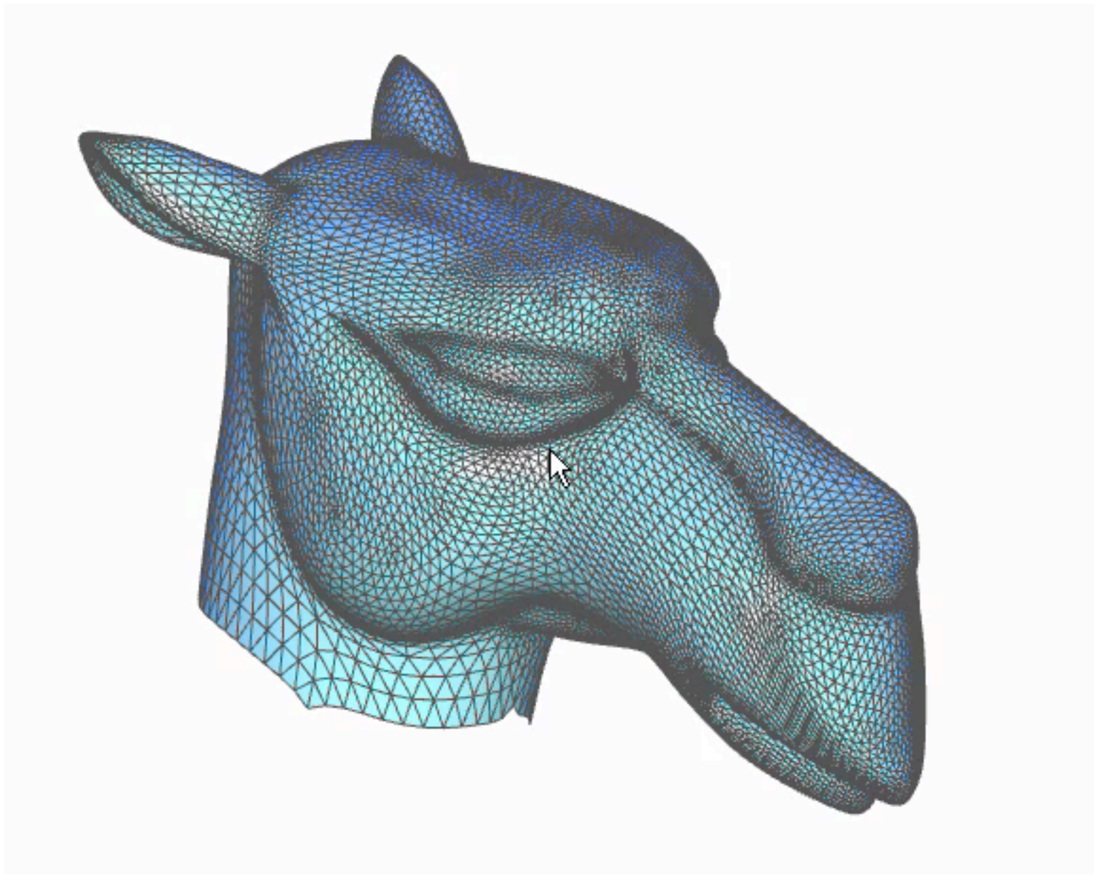
A Sketch-Based Interface for Detail-Preserving Mesh Editing [Nealen et al. 2005]

- Silhouette editing



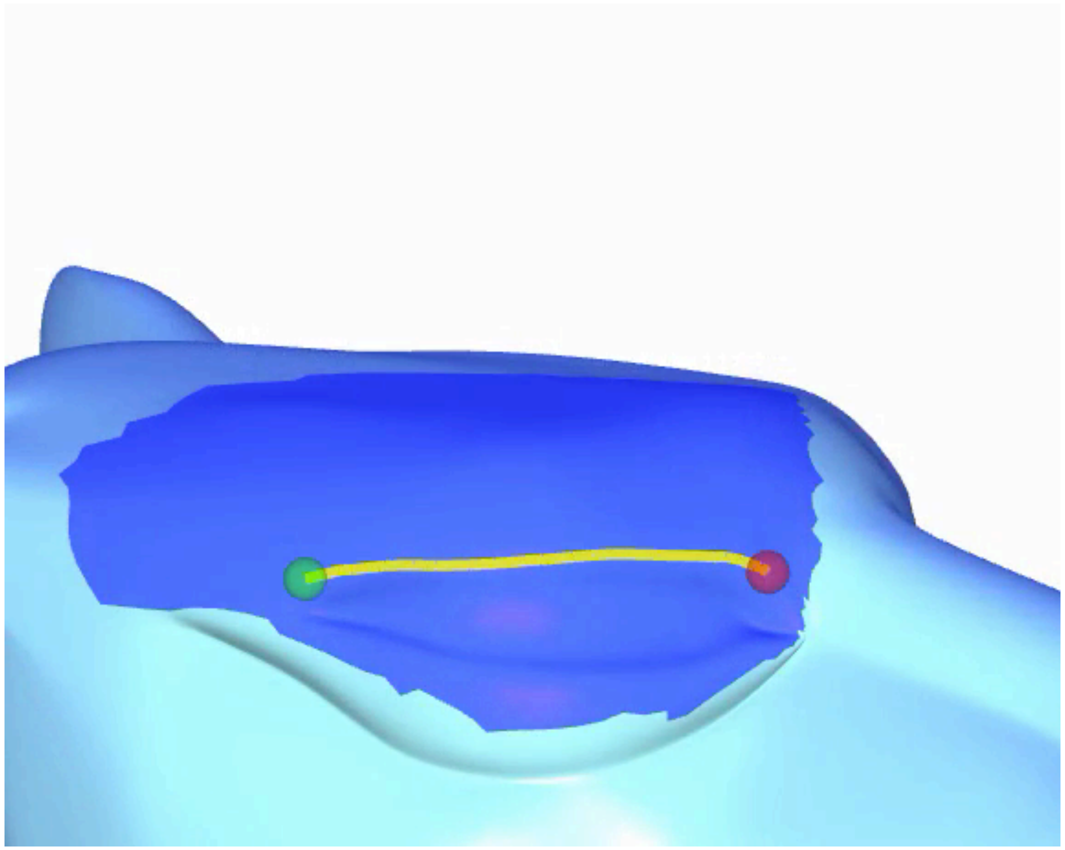
A Sketch-Based Interface for Detail-Preserving Mesh Editing [Nealen et al. 2005]

- Silhouette editing



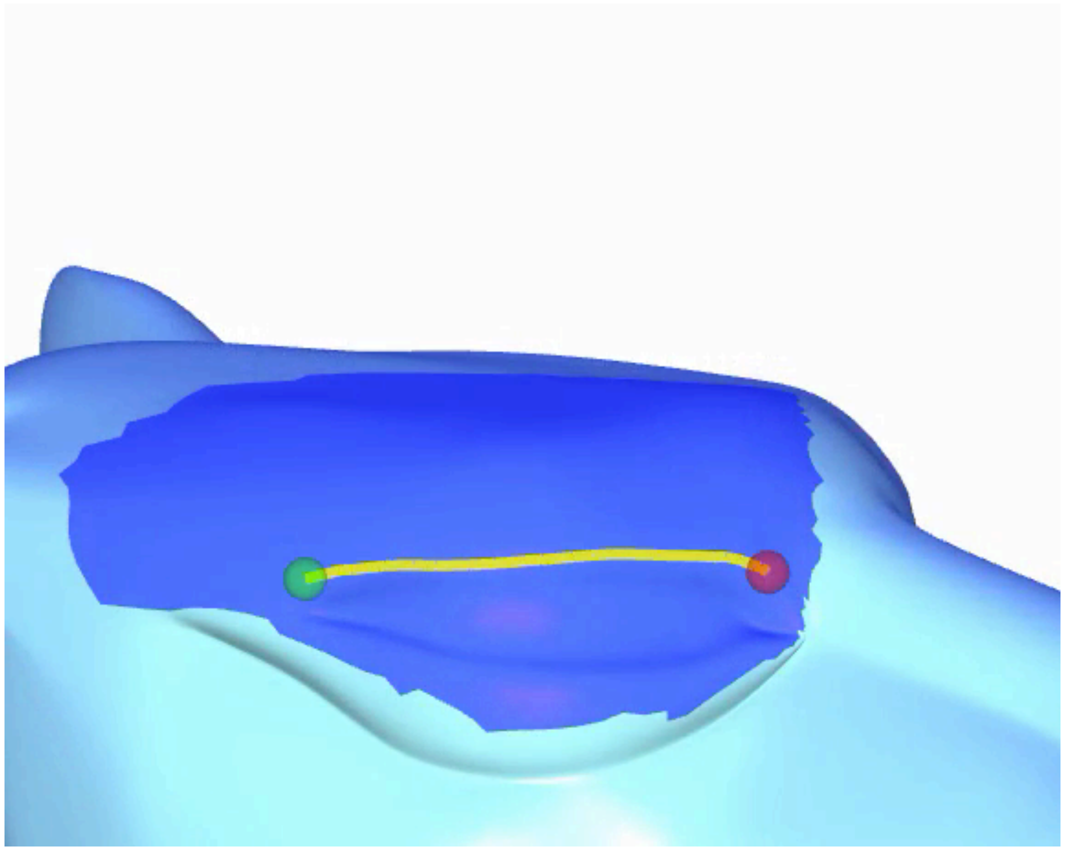
A Sketch-Based Interface for Detail-Preserving Mesh Editing [Nealen et al. 2005]

- Silhouette creation



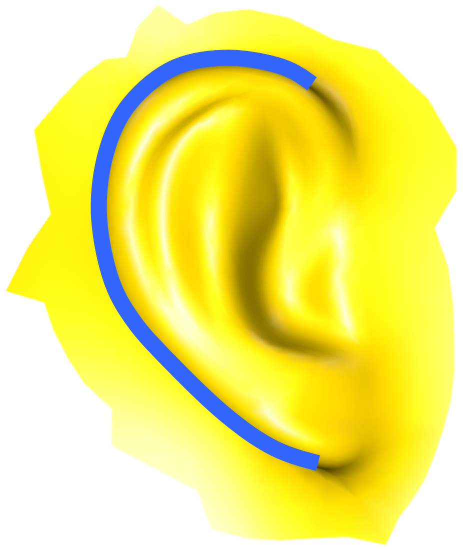
A Sketch-Based Interface for Detail-Preserving Mesh Editing [Nealen et al. 2005]

- Silhouette creation



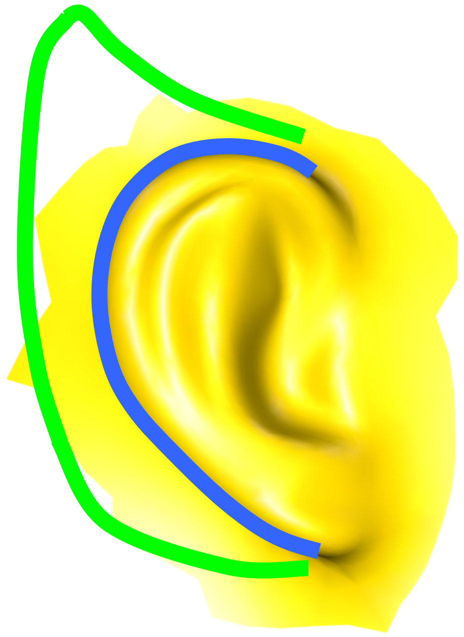
A Sketch-Based Interface for Detail-Preserving Mesh Editing [Nealen et al. 2005]

- To edit a silhouette:
  - Parameterize silhouette edges



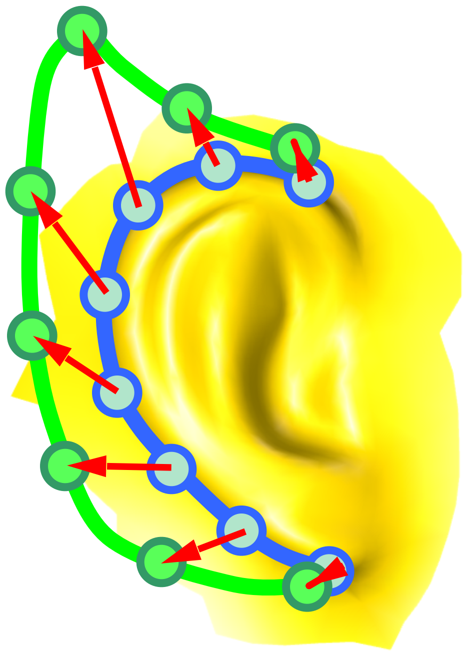
A Sketch-Based Interface for Detail-Preserving Mesh Editing [Nealen et al. 2005]

- To edit a silhouette:
  - Parameterize silhouette edges
  - Parameterize sketch



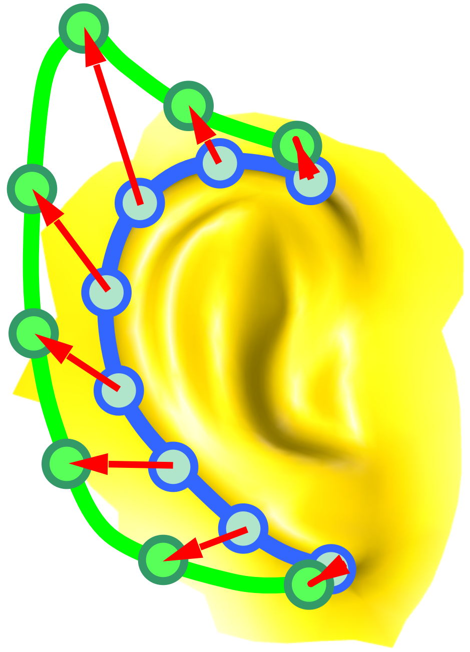
A Sketch-Based Interface for Detail-Preserving Mesh Editing [Nealen et al. 2005]

- To edit a silhouette:
  - Parameterize silhouette edges
  - Parameterize sketch
  - Find correspondences



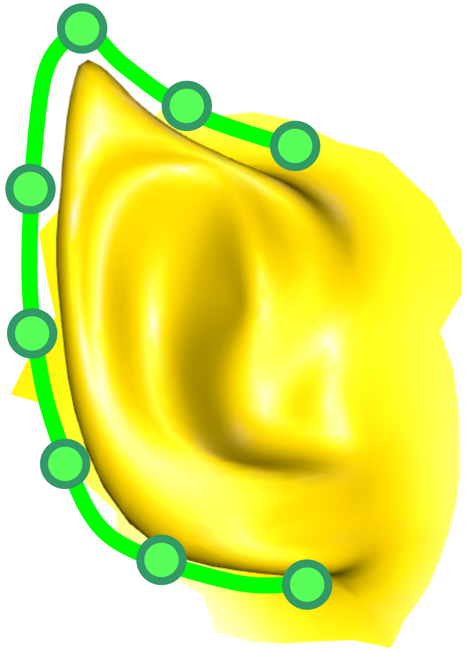
A Sketch-Based Interface for Detail-Preserving Mesh Editing [Nealen et al. 2005]

- To edit a silhouette:
  - Parameterize silhouette edges
  - Parameterize sketch
  - Find correspondences
  - Use as xy position constraints (keep z unchanged)

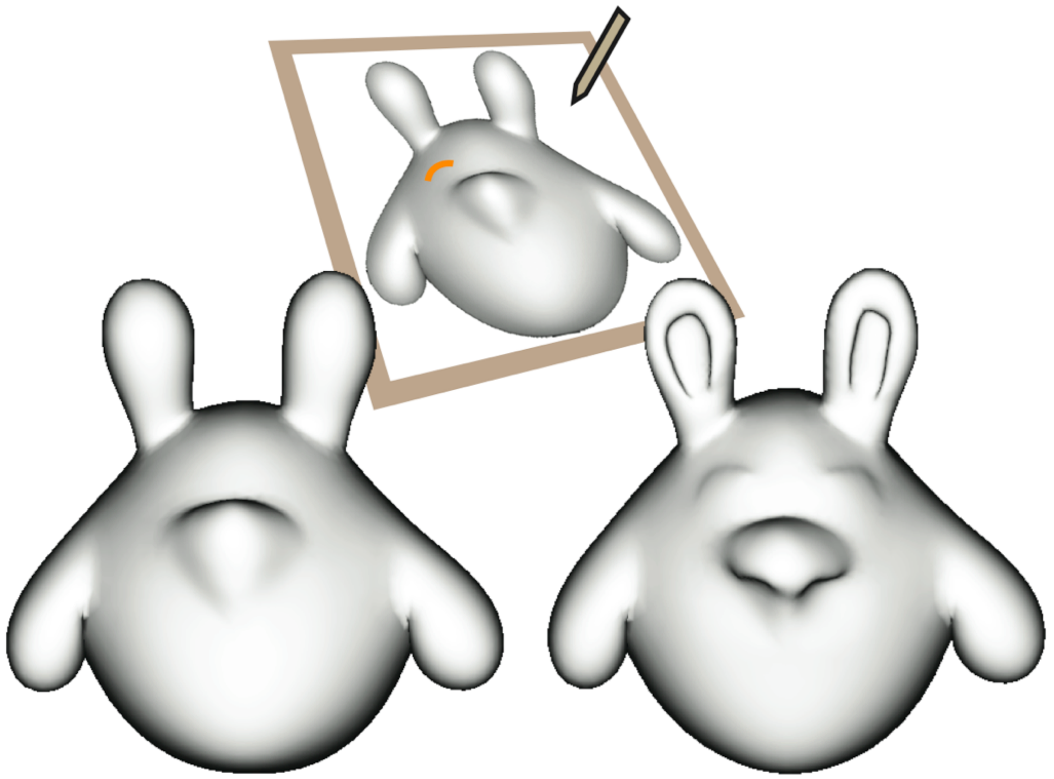


A Sketch-Based Interface for Detail-Preserving Mesh Editing [Nealen et al. 2005]

- To edit a silhouette:
  - Parameterize silhouette edges
  - Parameterize sketch
  - Find correspondences
  - Use as xy position constraints (keep z unchanged)
  - Minimize Laplacian Surface Editing energy [Sorkine et al. 2004]



Surface relief editing by sketching shading

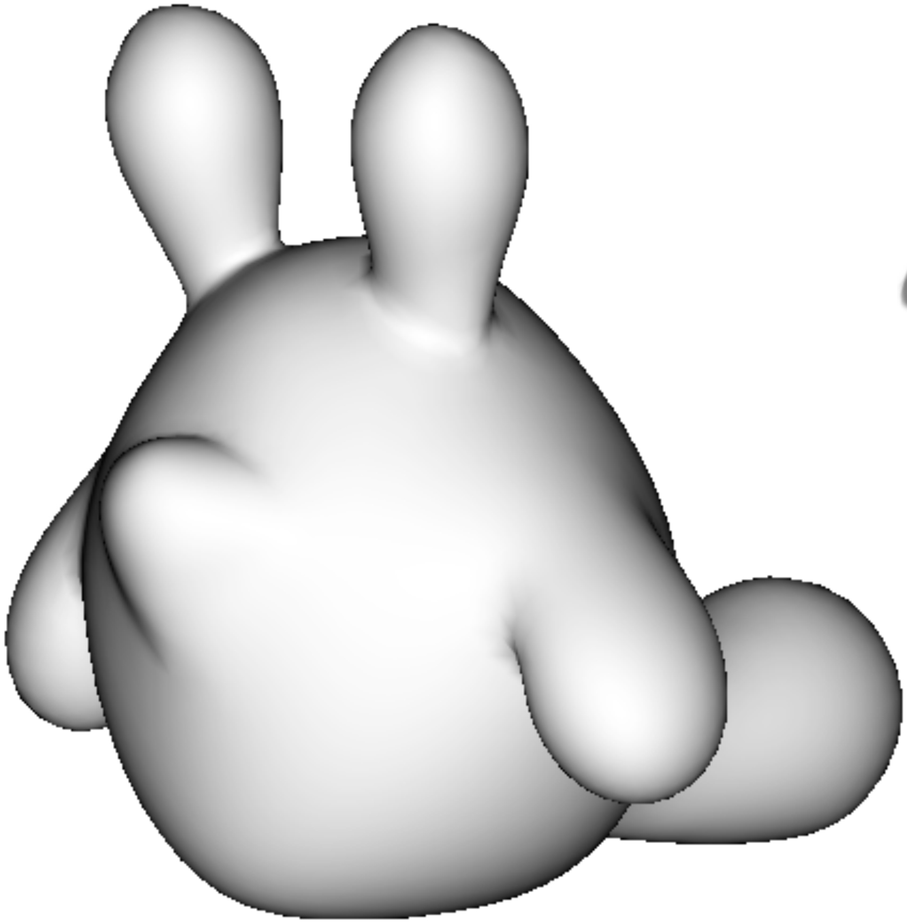

Shading-Based Surface Editing [Gingold and Zorin 2008]

Shading

[Michelangelo]

[Dürer]

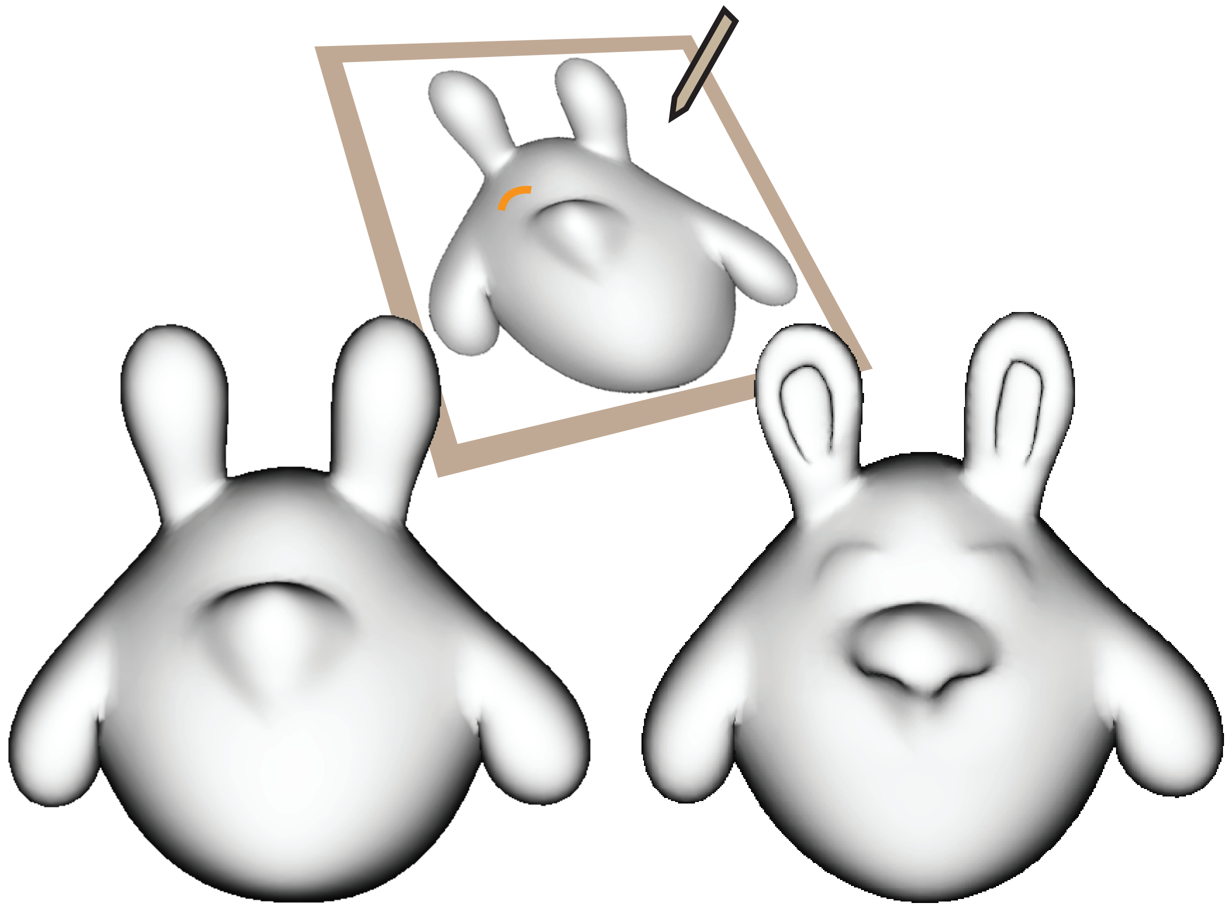
Shaded 3D Models


FiberMesh [Nealen et al. 2007]

[Malanjo]

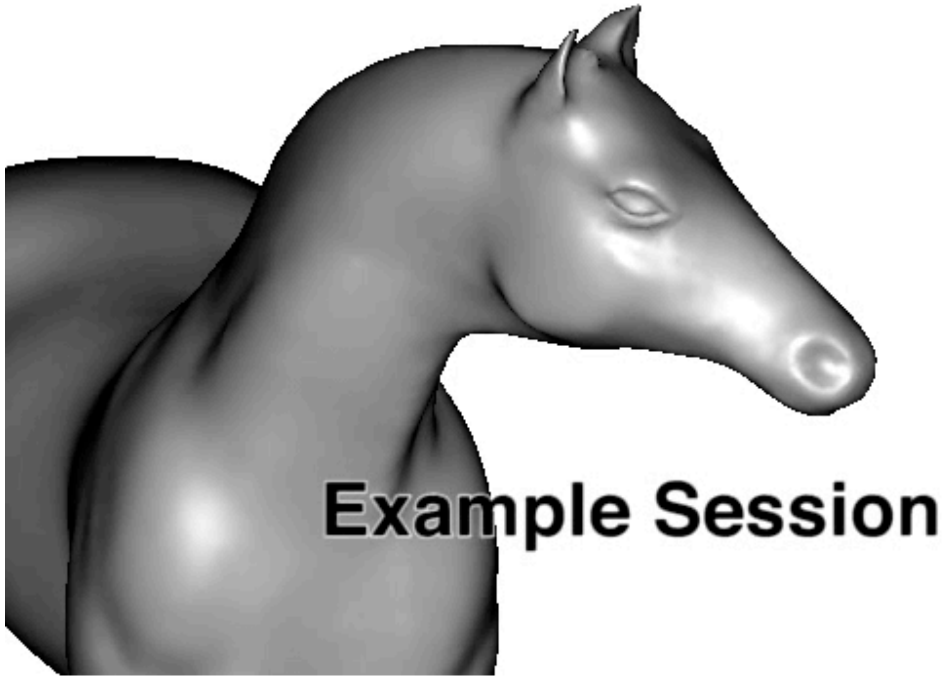
Approach

Obtain a new 3D model by shading over an existing one.



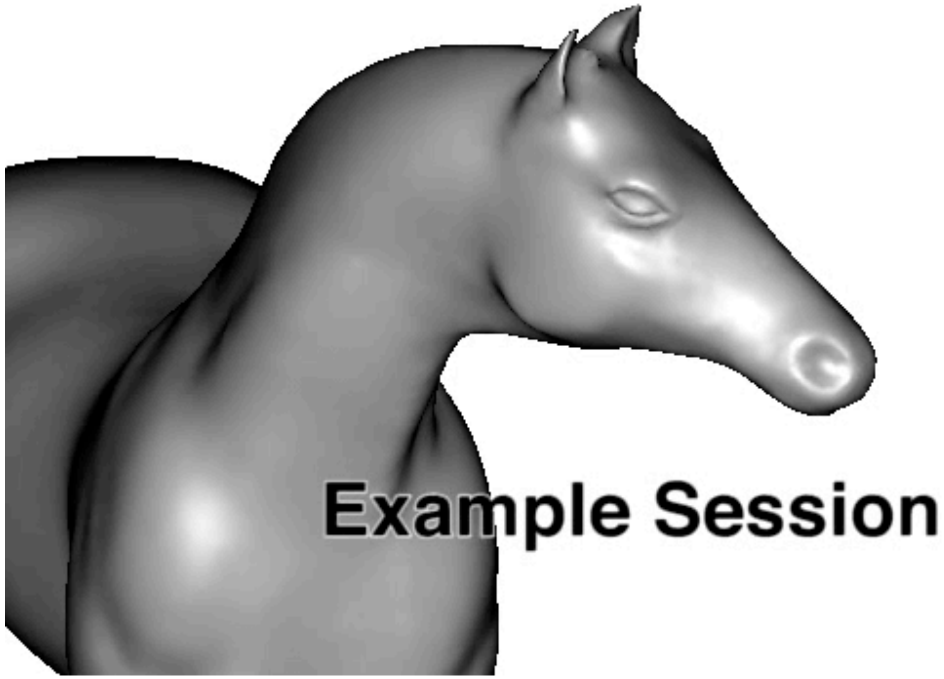
Approach

Obtain a new 3D model by shading over an existing one.



Approach

Obtain a new 3D model by shading over an existing one.



Goal

An interactive tool for surface editing by “*drawing what you want to see.*”

Goal

An interactive tool for surface editing by “*drawing what you want to see.*”

- Leverages artists’ experience with shading

Goal

An interactive tool for surface editing by "*drawing what you want to see.*"

- Leverages artists' experience with shading
- Brush parameters similar to paint programs

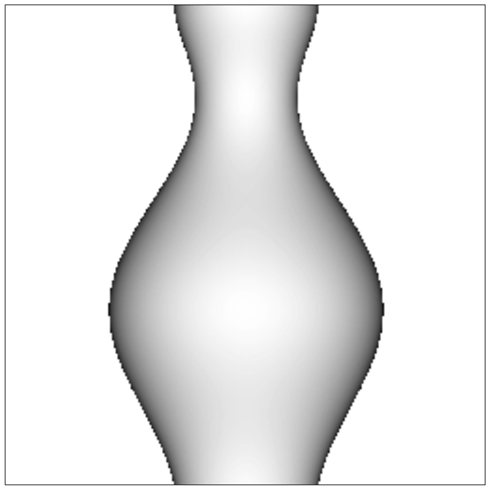
Goal

An interactive tool for surface editing by “*drawing what you want to see.*”

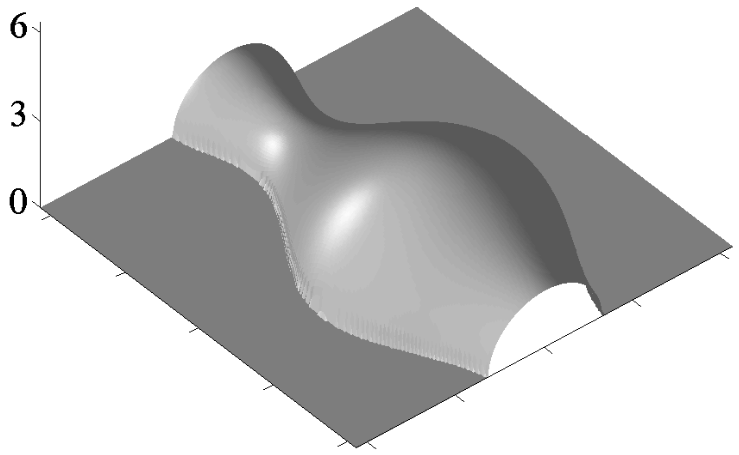
- Leverages artists’ experience with shading
- Brush parameters similar to paint programs
- Stable, predictable, approximate solution for a special case of Shape-from-Shading

Shape-from-Shading

Given a shaded image of an object, can we recover its shape?



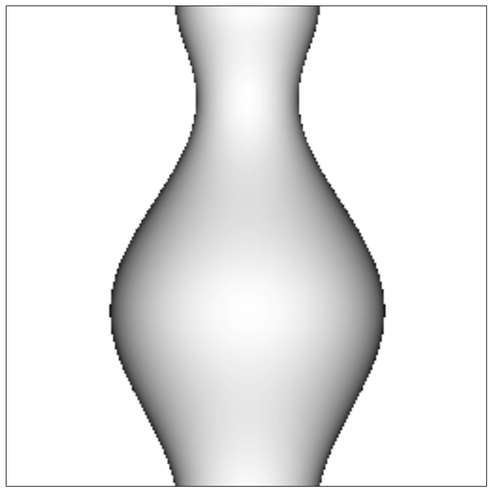
Shaded Image



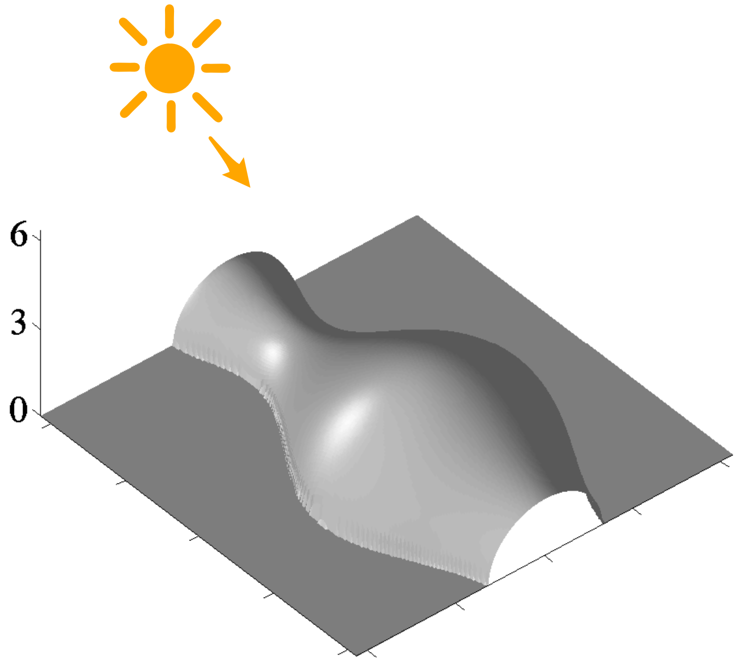
Shape (Height Field)

Shape-from-Shading

Given a shaded image of an object, can we recover its shape?



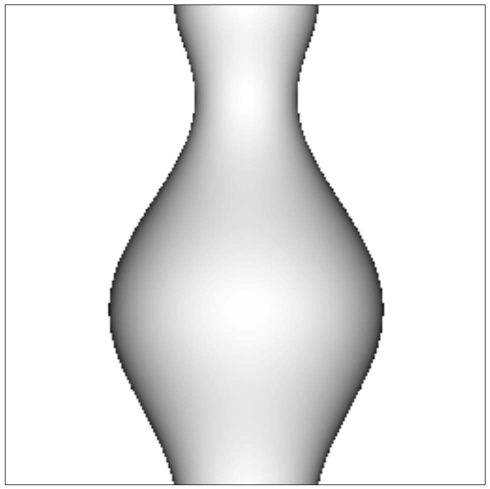
Shaded Image



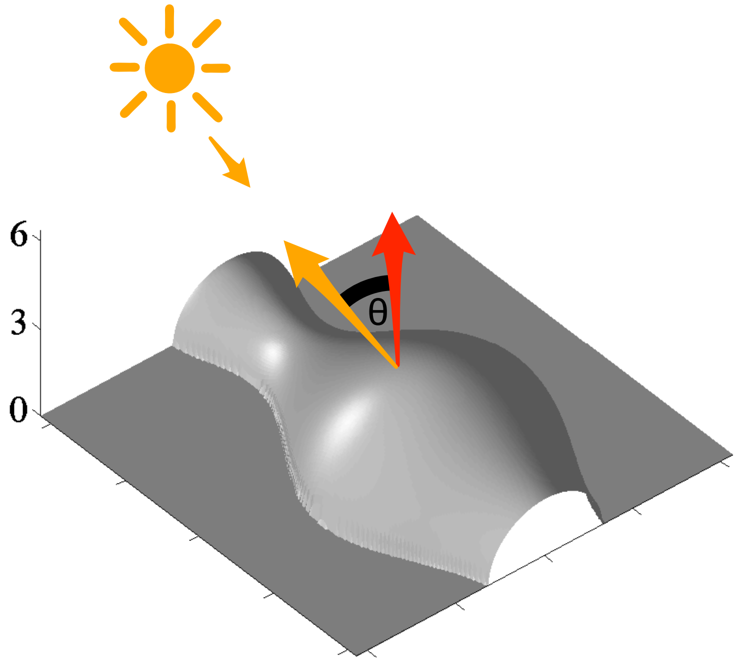
Shape (Height Field)

Shape-from-Shading

Given a shaded image of an object, can we recover its shape?



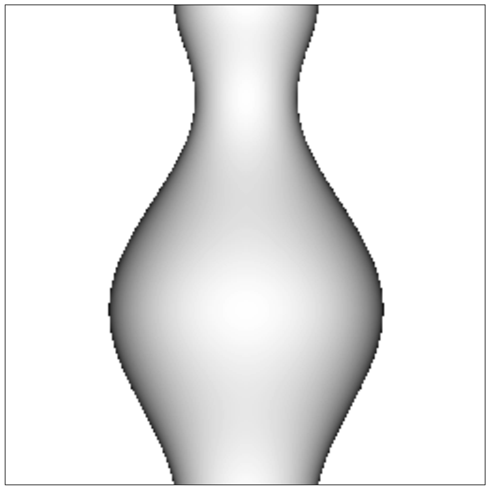
Shaded Image

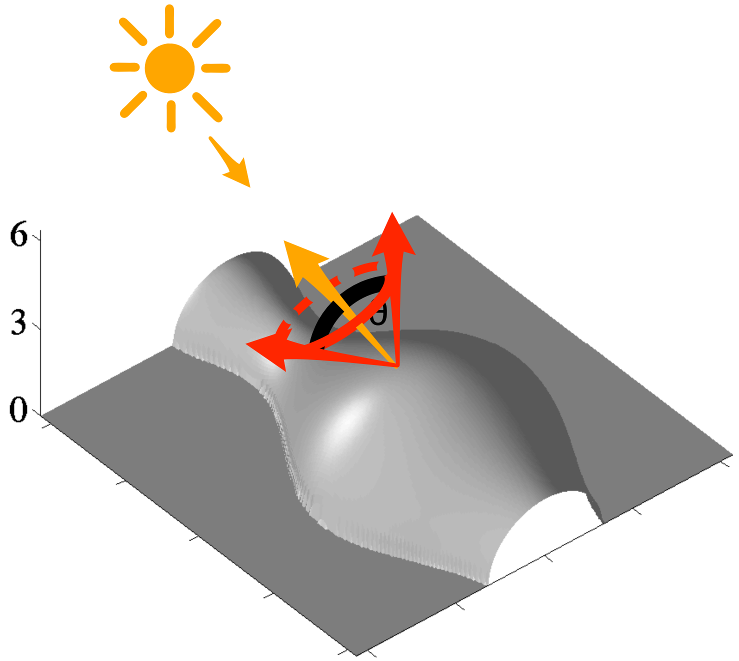


Shape (Height Field)

Shape-from-Shading

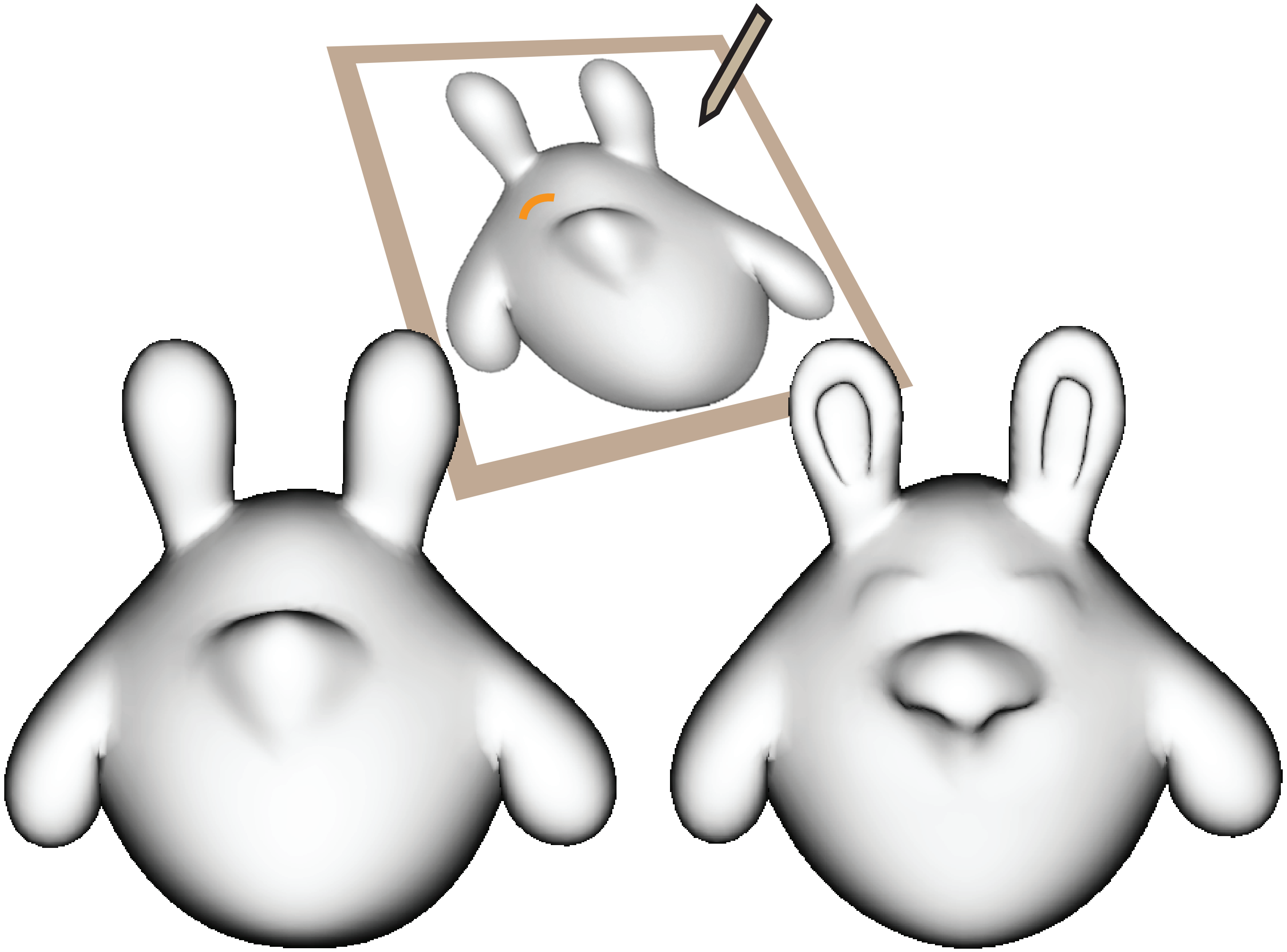
Given a shaded image of an object, can we recover its shape?


Shaded Image



Shape (Height Field)

User Interface



Shading Strokes

Shading Strokes

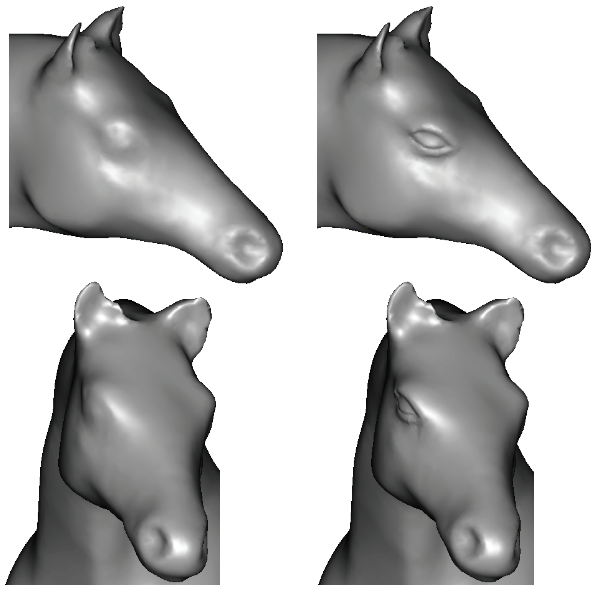
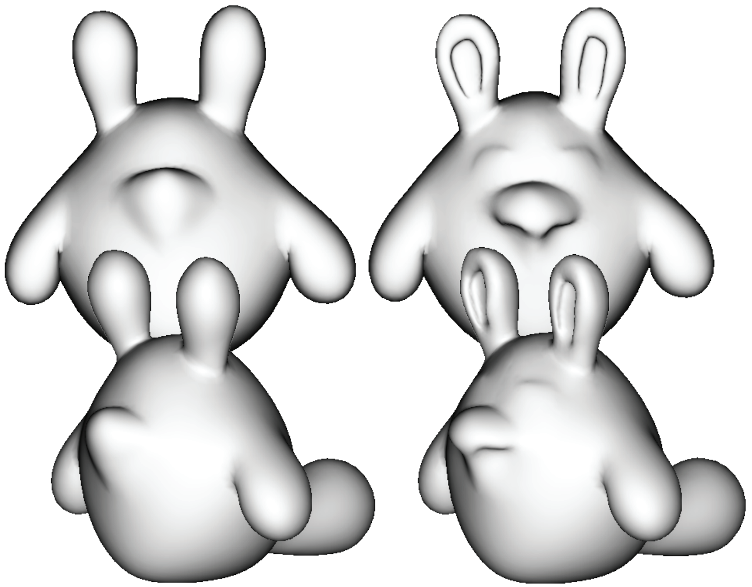
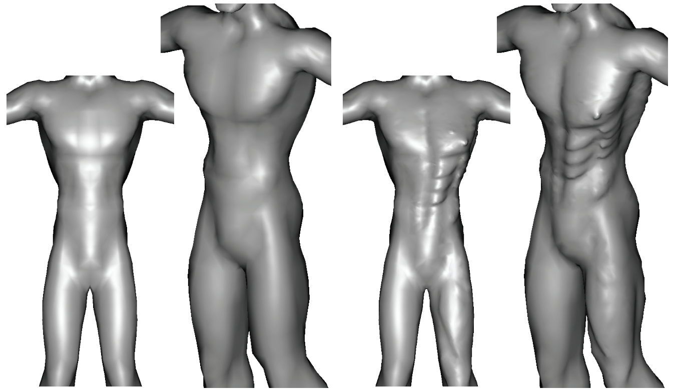
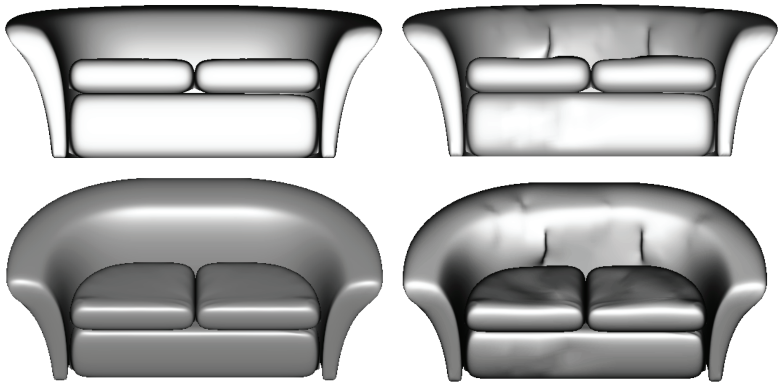
Thin Strokes

Silhouette Stroke

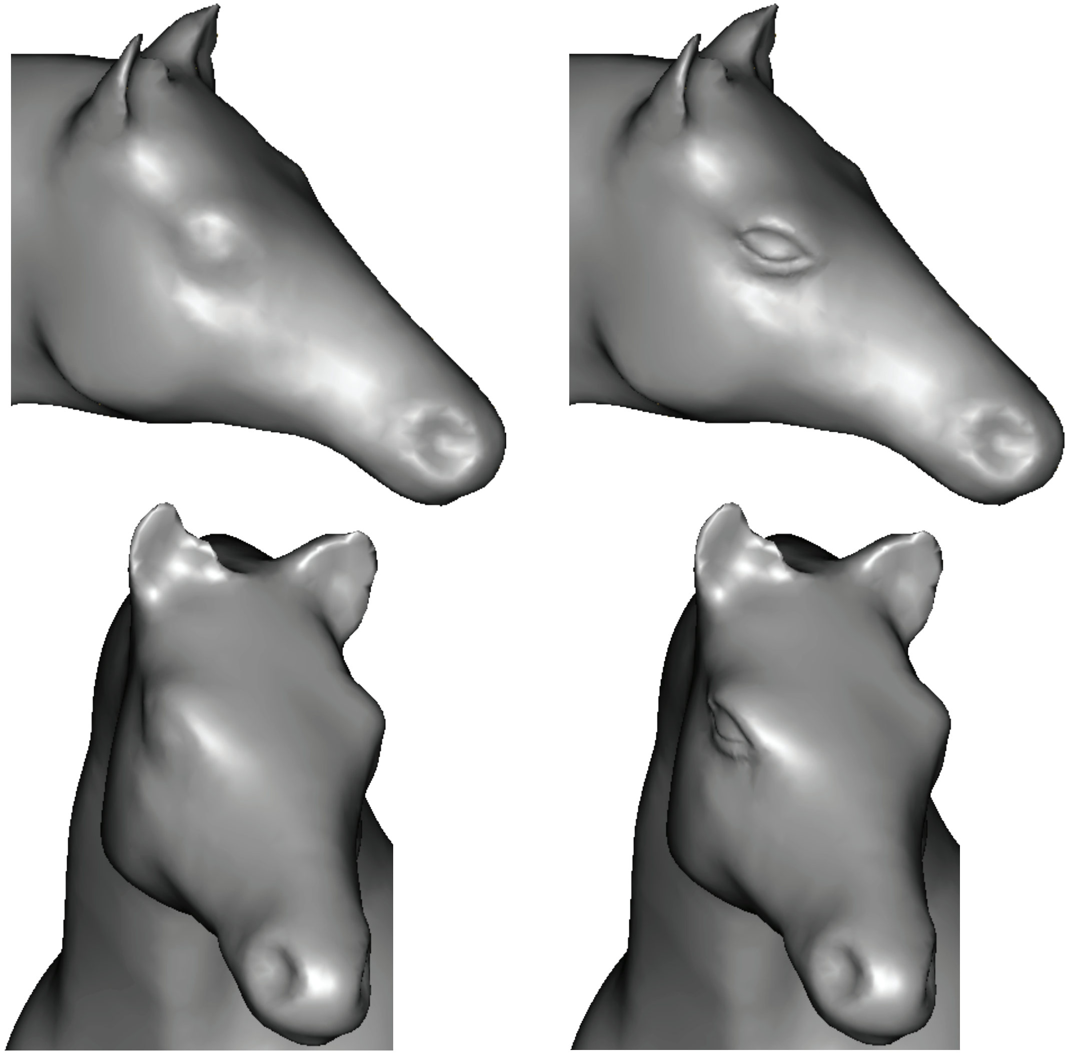
Silhouette Stroke

Silhouette Strokes

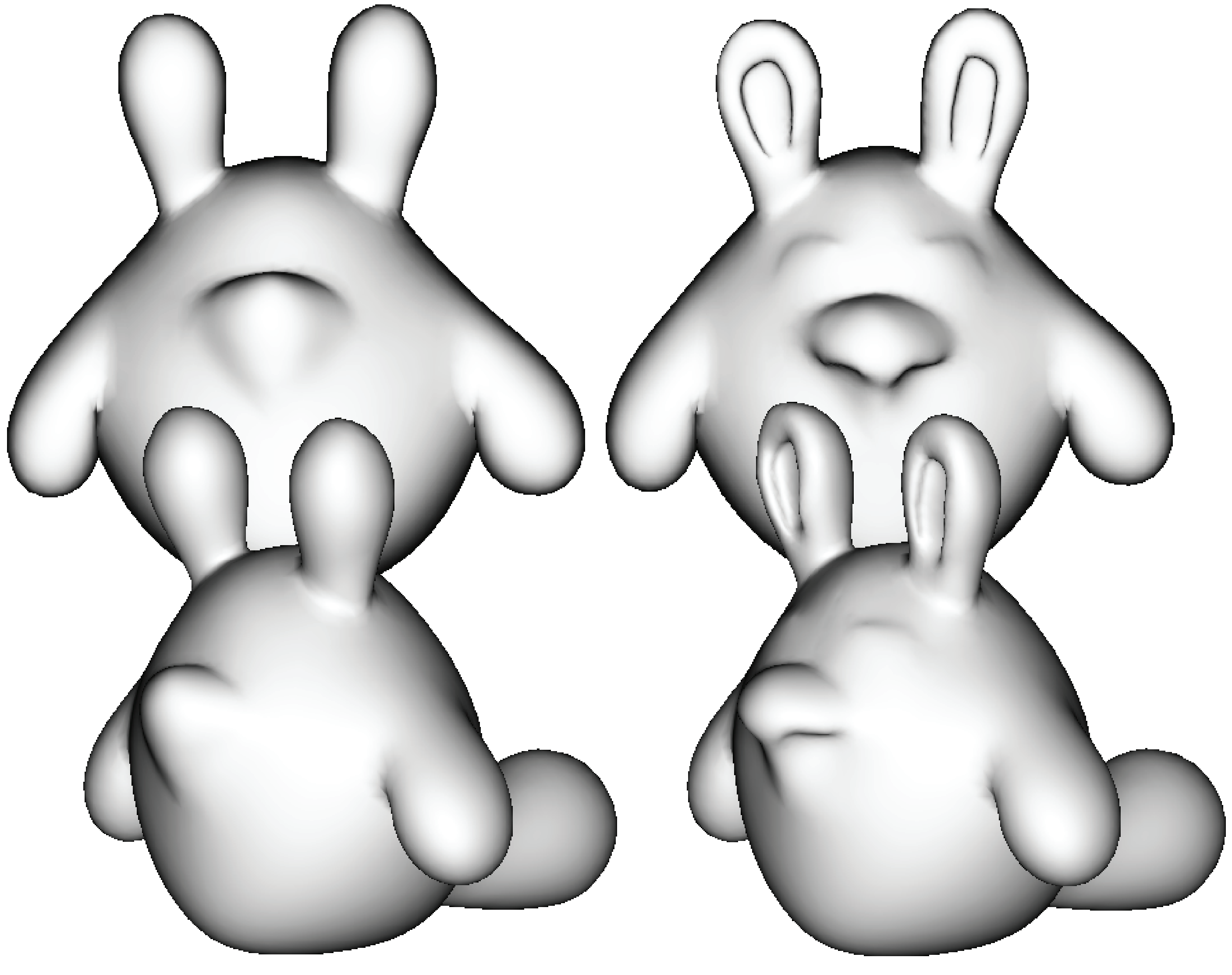
Results

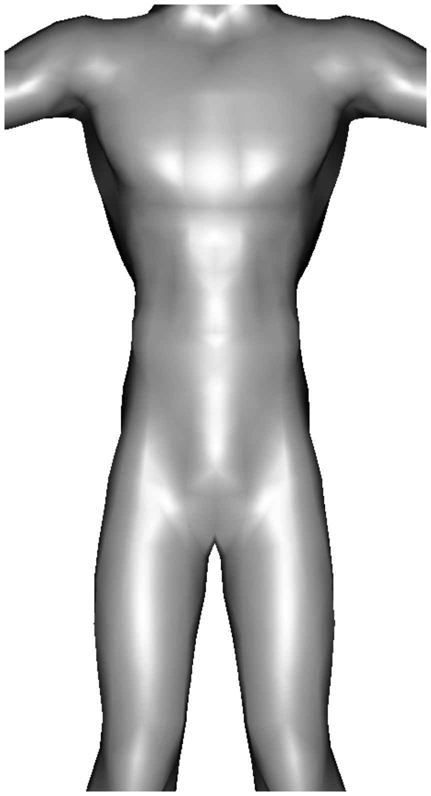
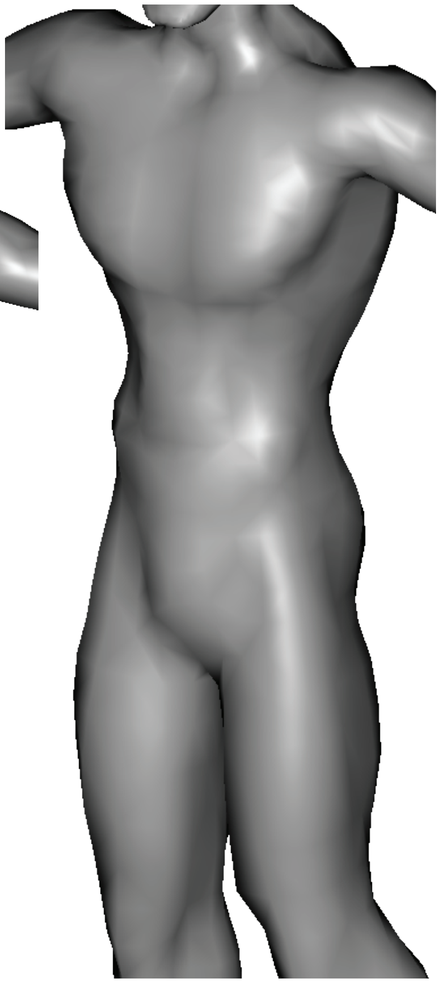
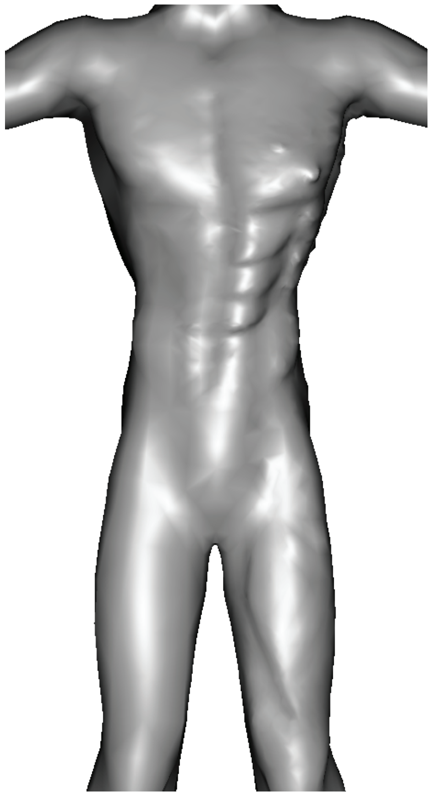
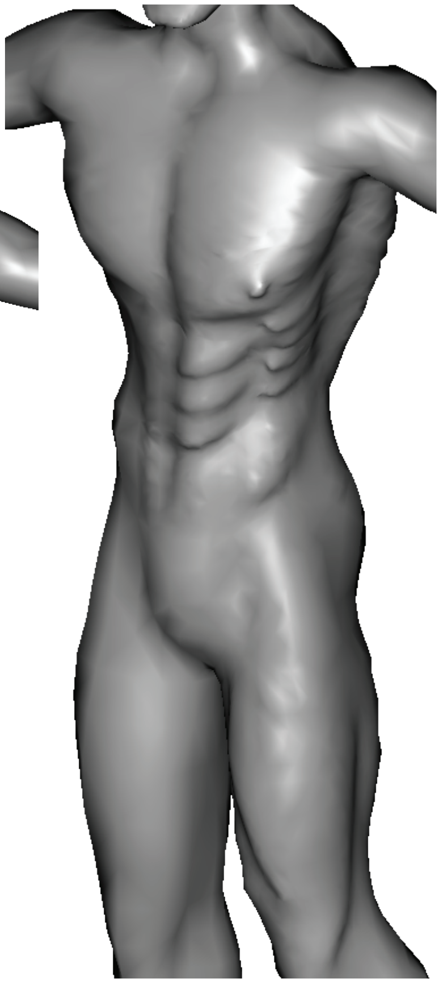
Results



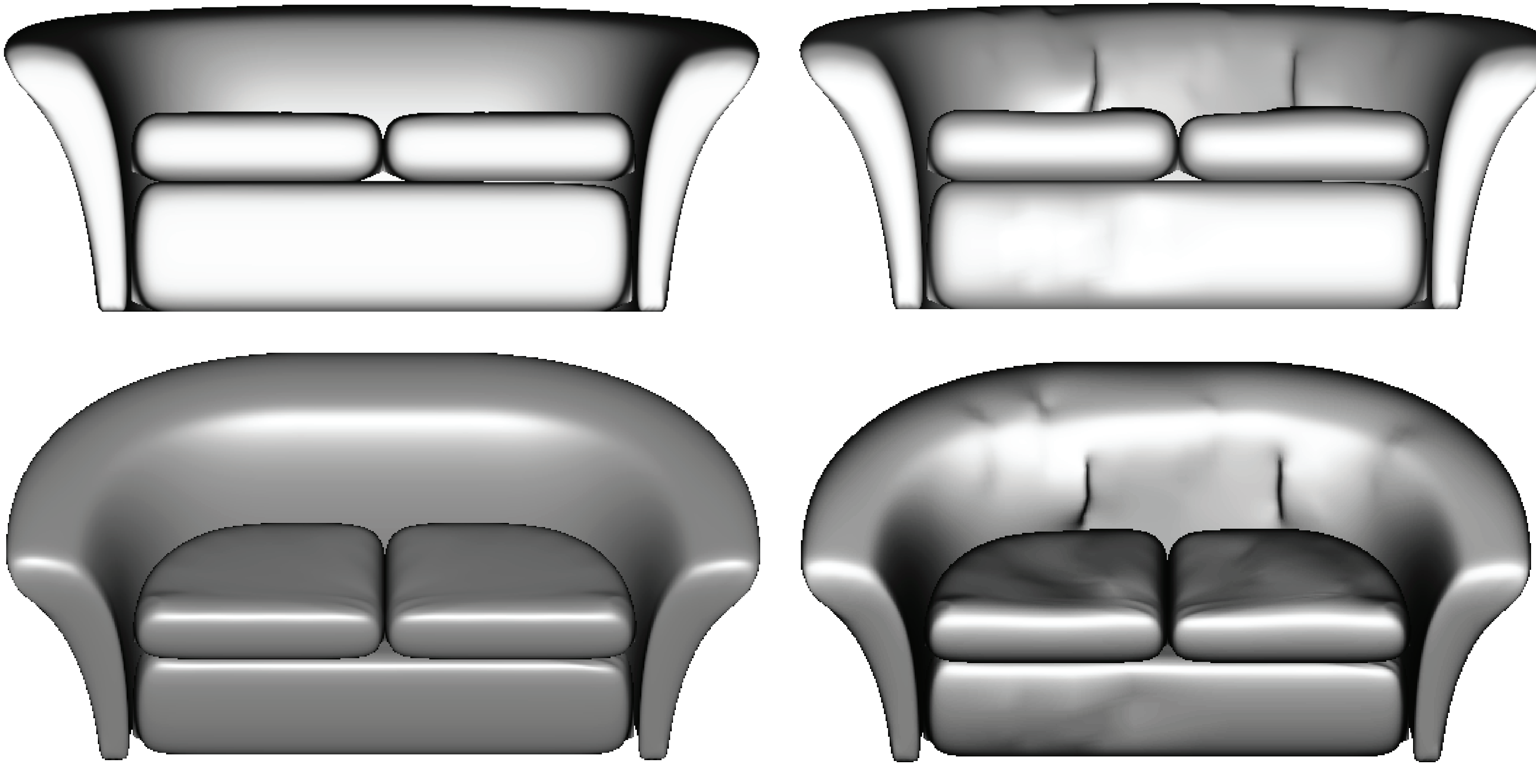
Results



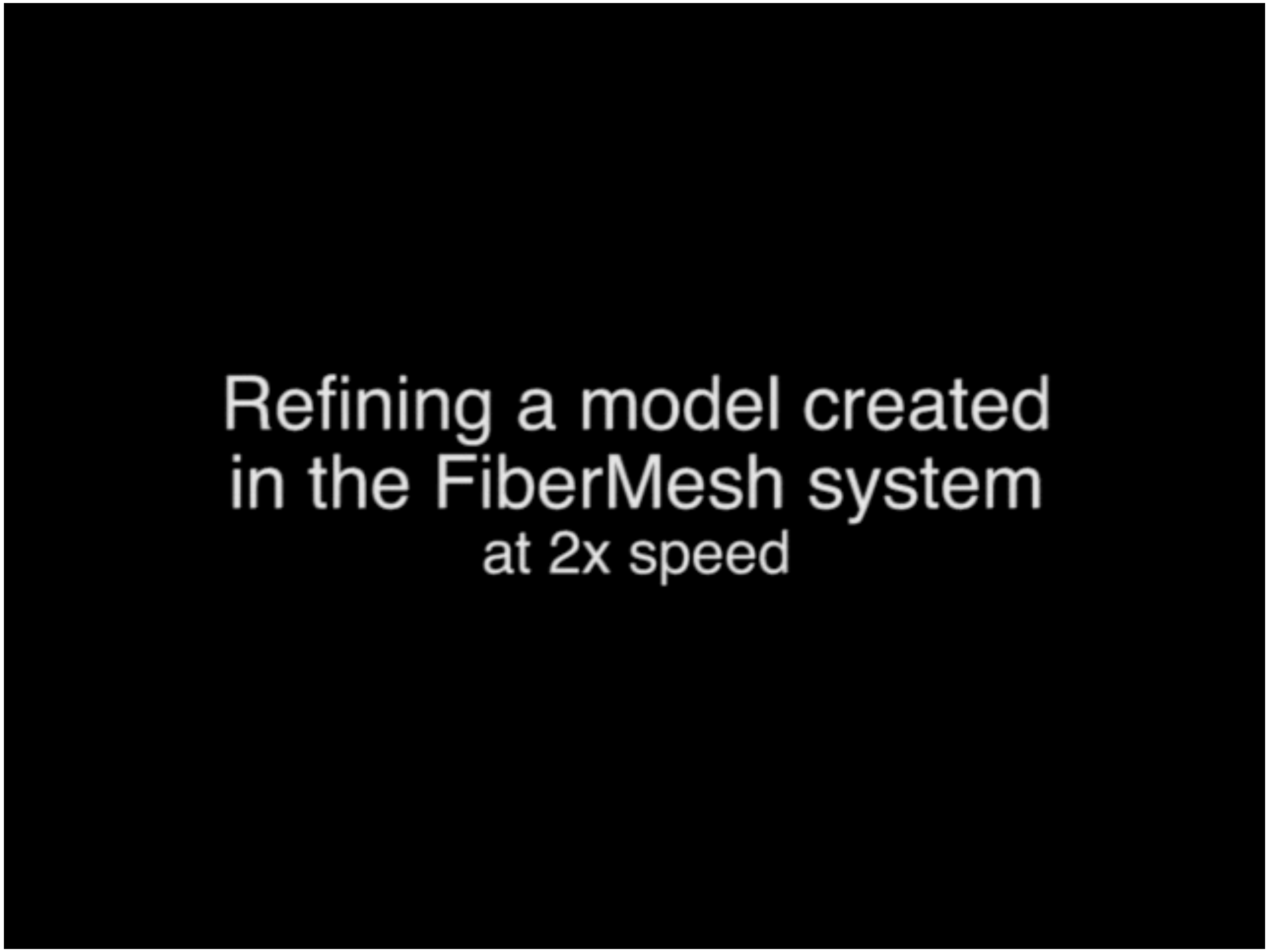
Results

Results

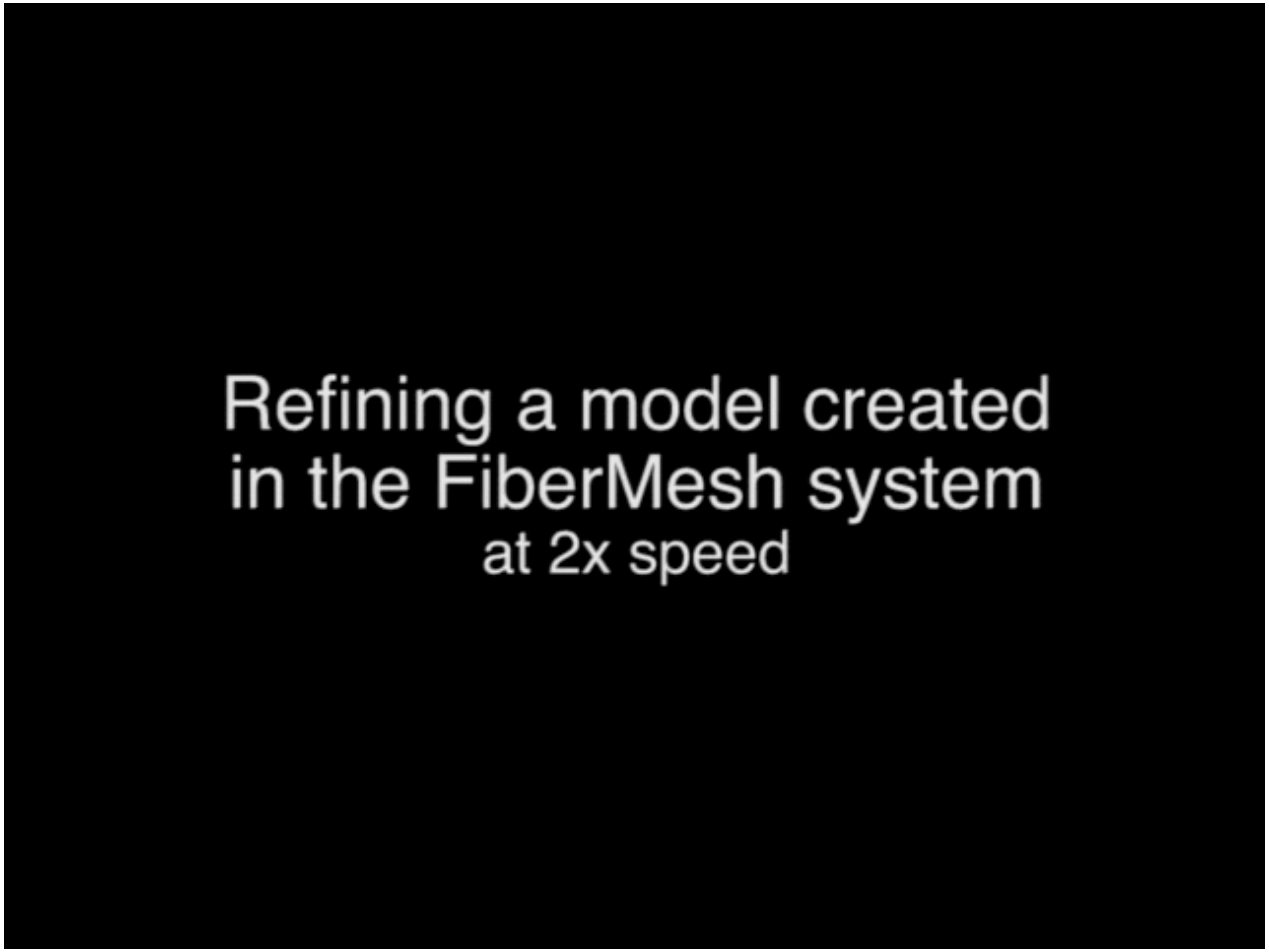


Video



Refining a model created
in the FiberMesh system
at 2x speed

Video


Refining a model created
in the FiberMesh system
at 2x speed

Criteria for Controllability

Interface should balance:

Criteria for Controllability

Interface should balance:

- Stability (small changes produce small effects)

Criteria for Controllability

Interface should balance:

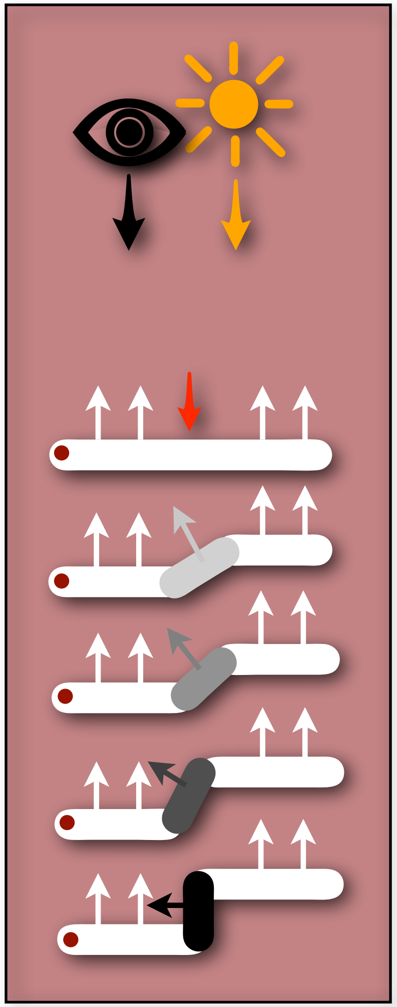
- Stability (small changes produce small effects)
- Appearance and shape preserved elsewhere

Criteria for Controllability

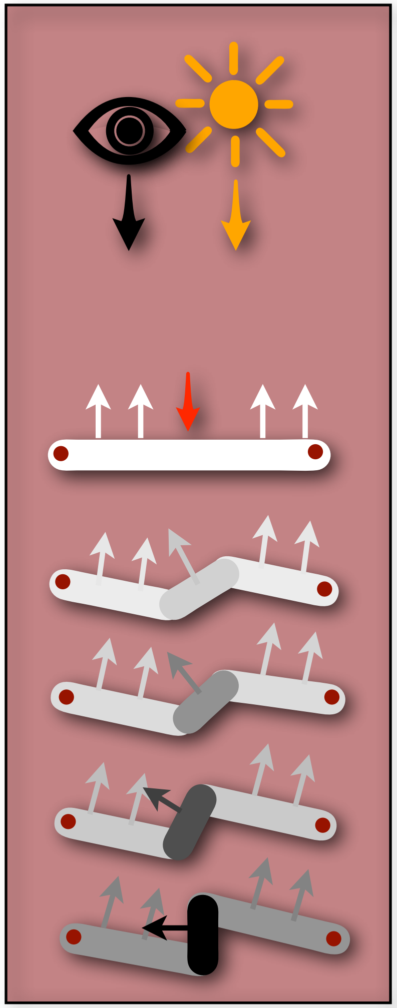
Interface should balance:

- Stability (small changes produce small effects)
- Appearance and shape preserved elsewhere
- Predictability

Why is this hard?

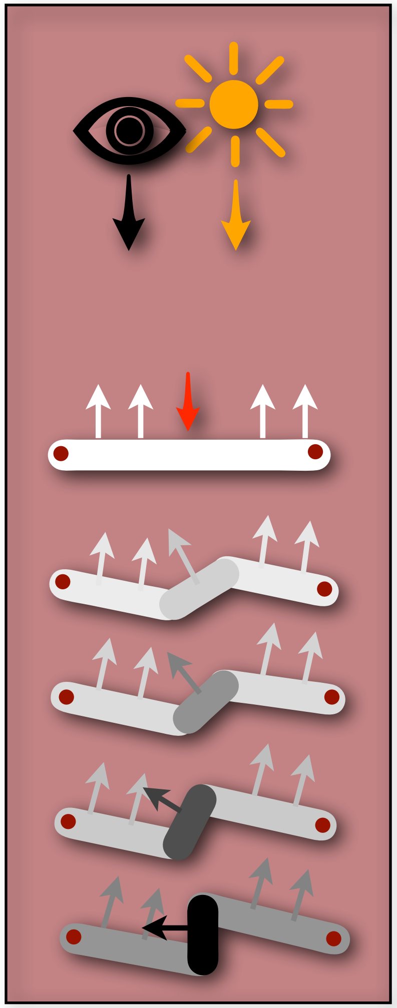


Why is this hard?

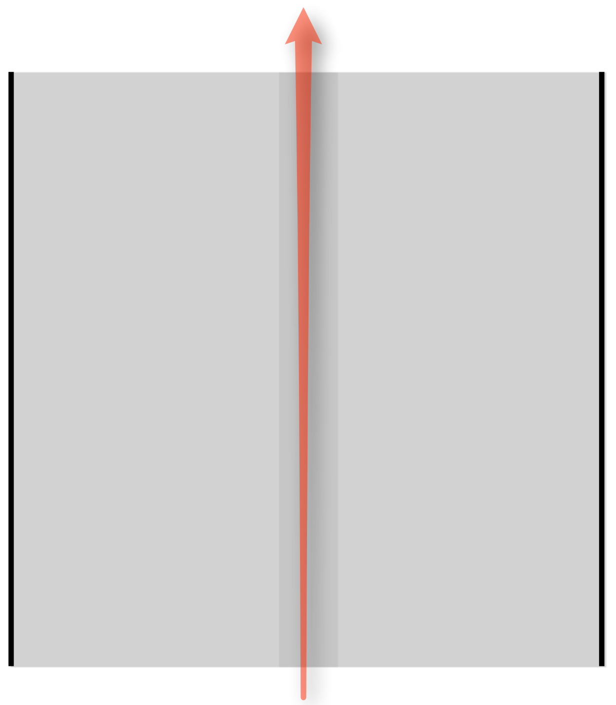


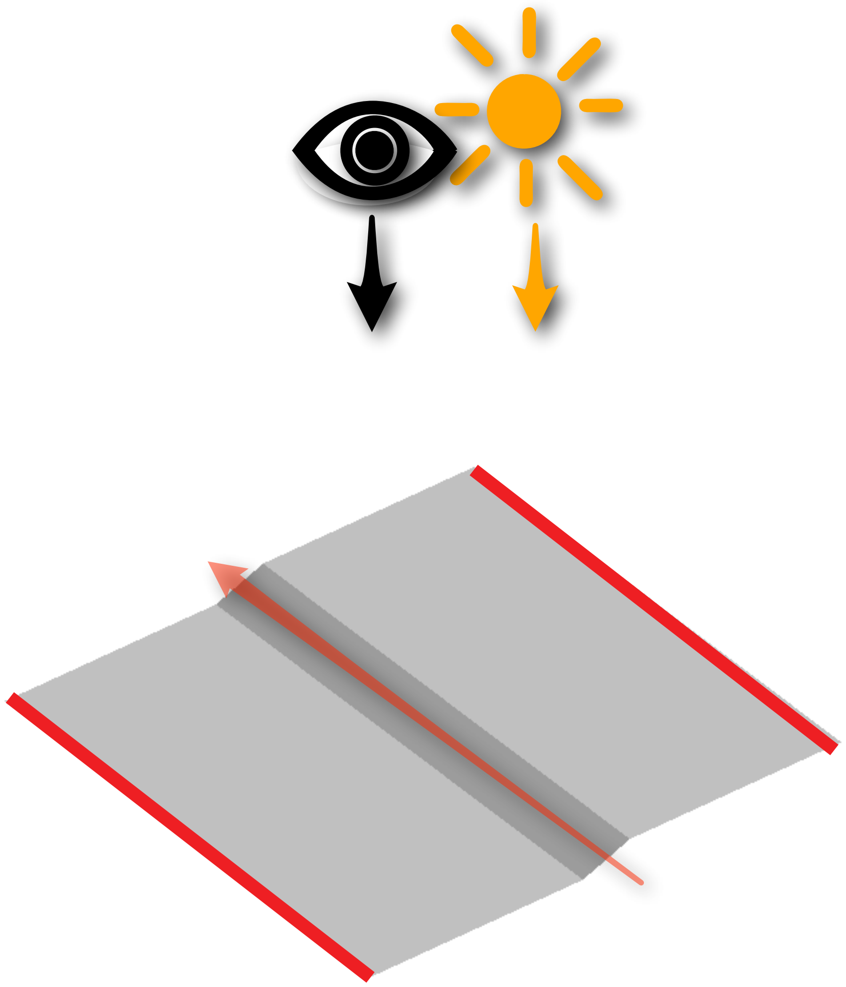
Why is this hard?

The change is global



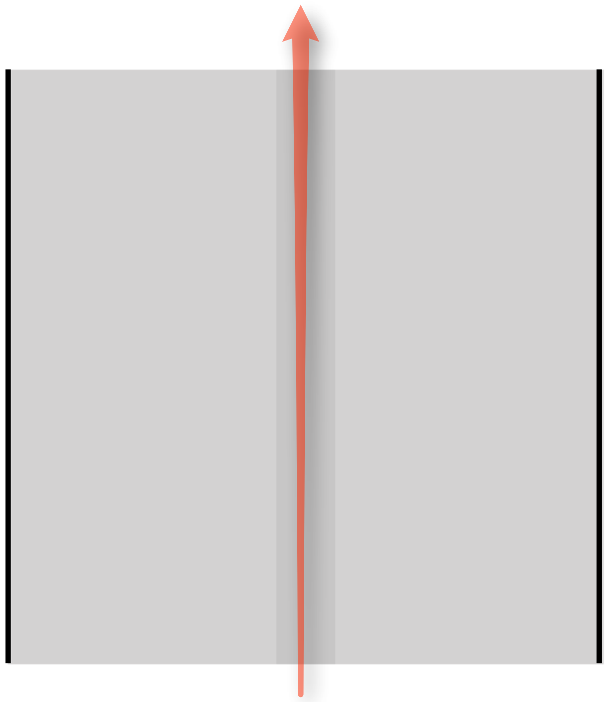
Why is this hard?

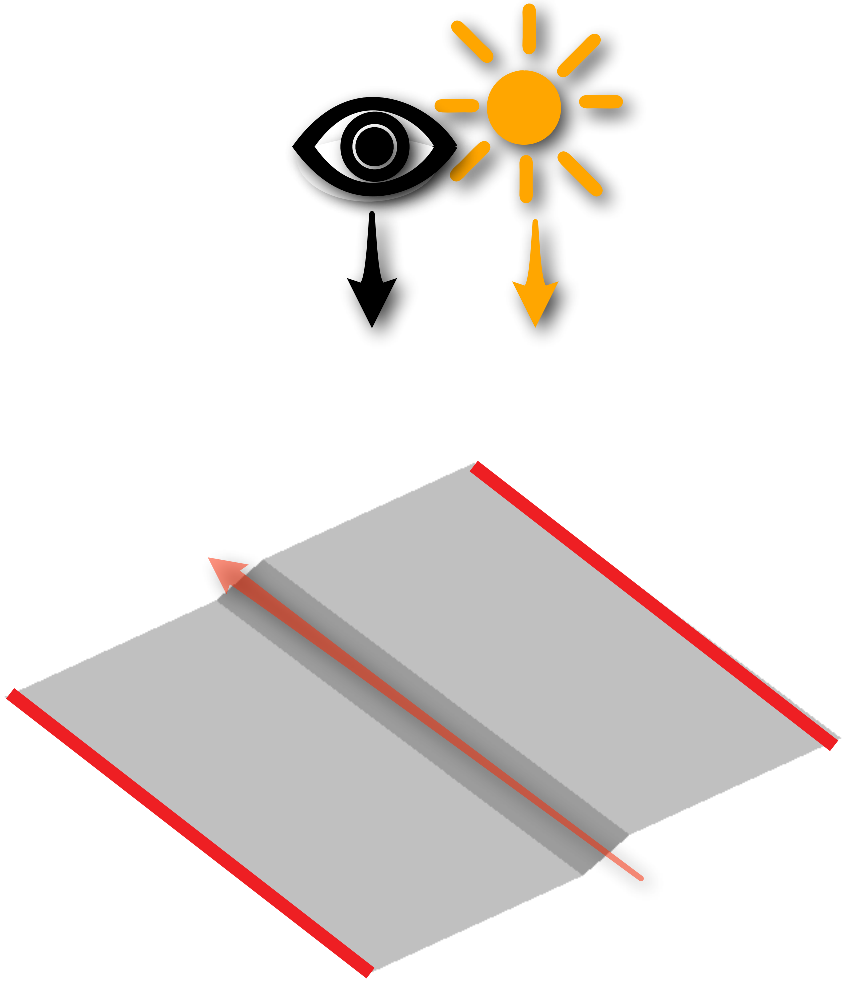




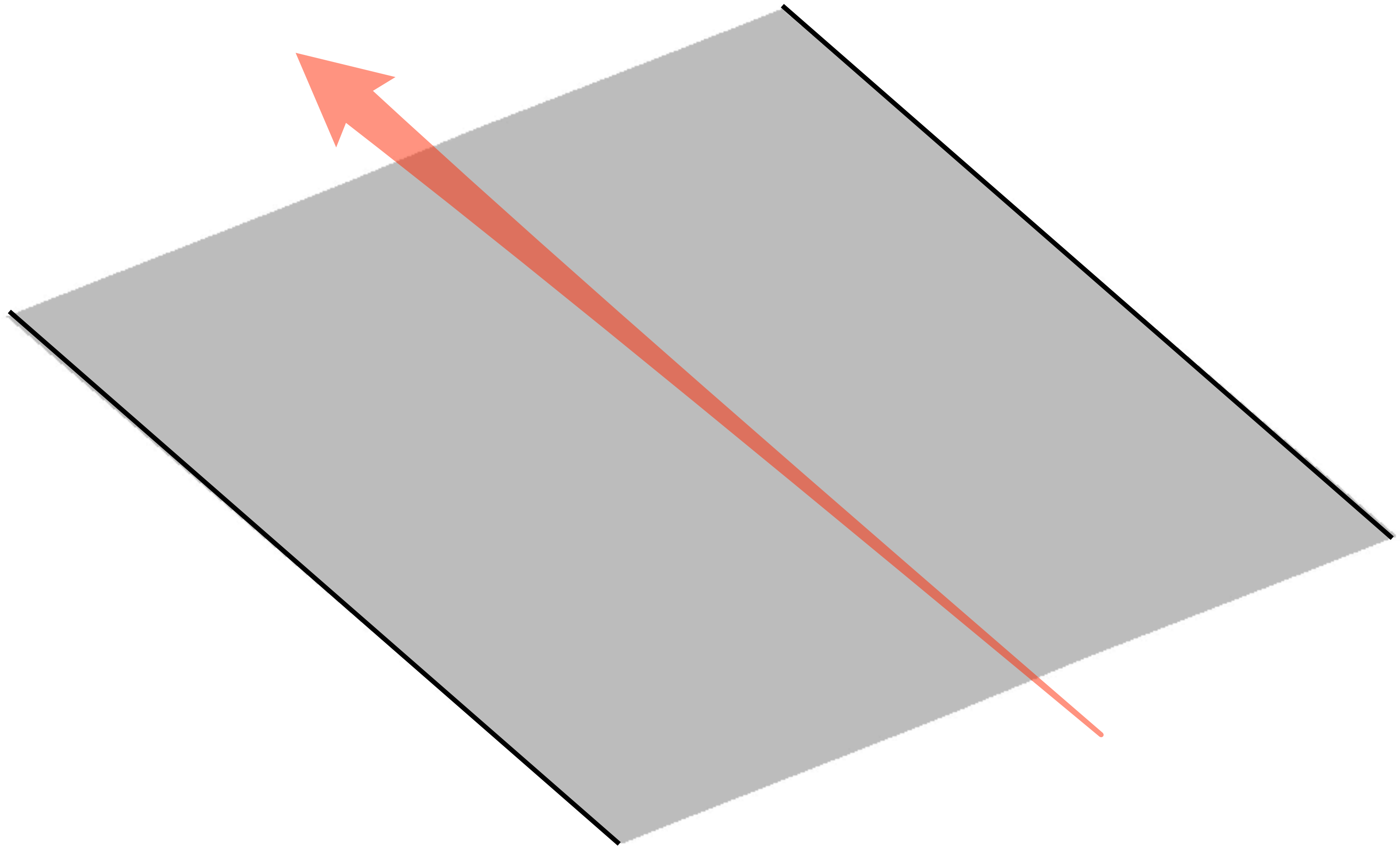
Why is this hard?

The change is global

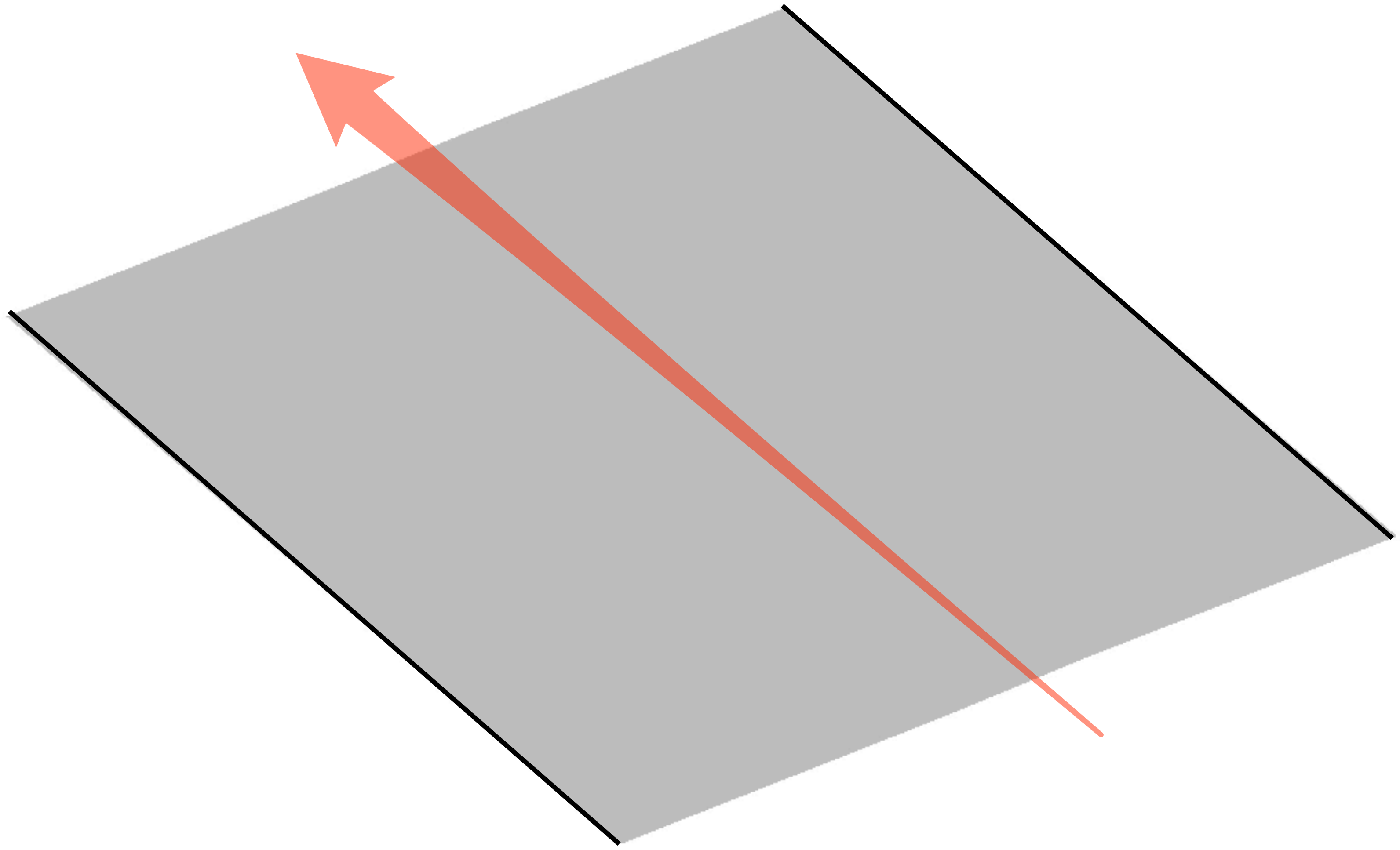




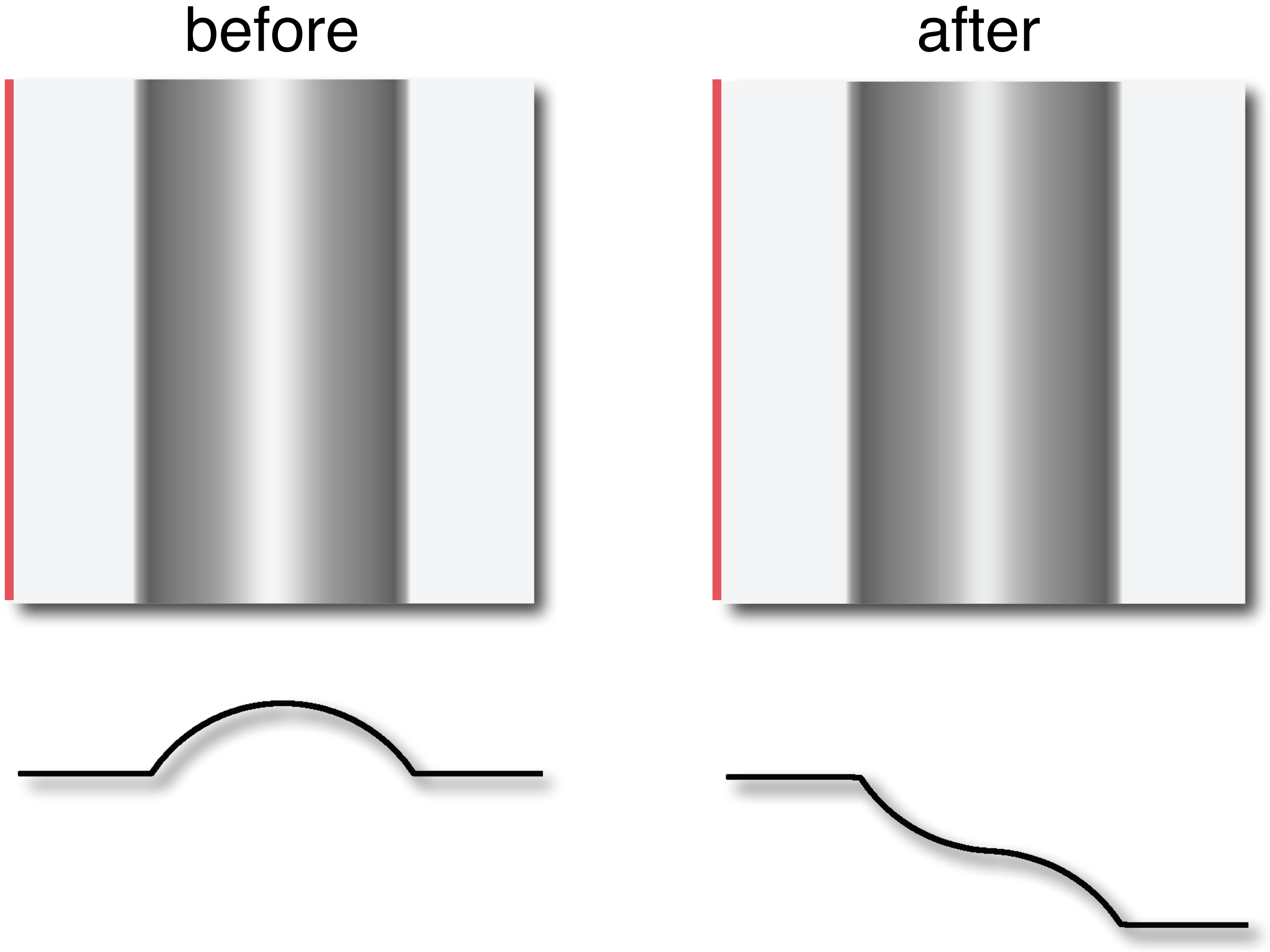
Rotation about the stroke



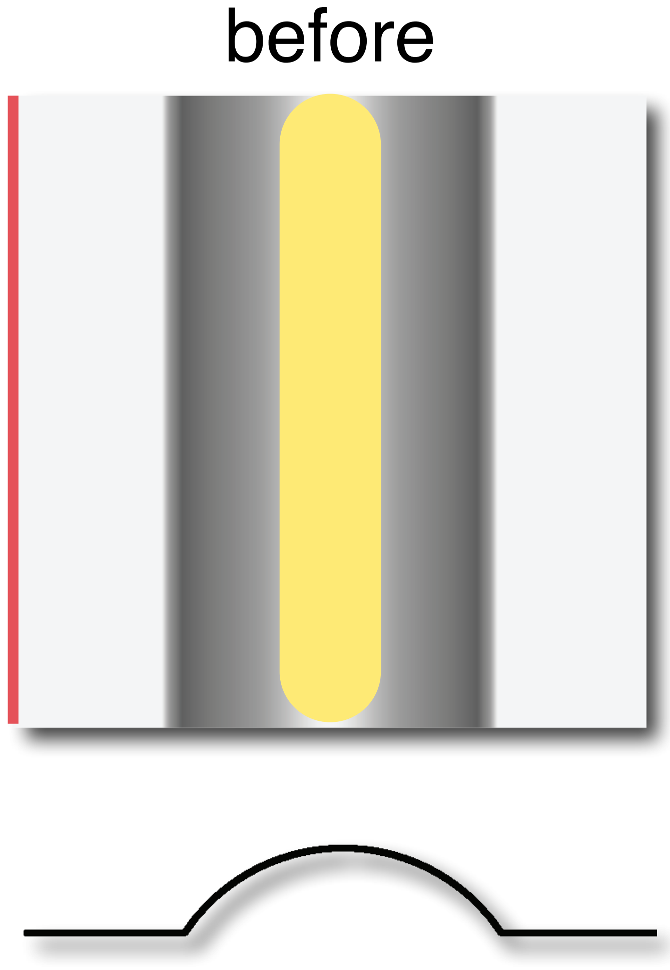
Rotation about the stroke

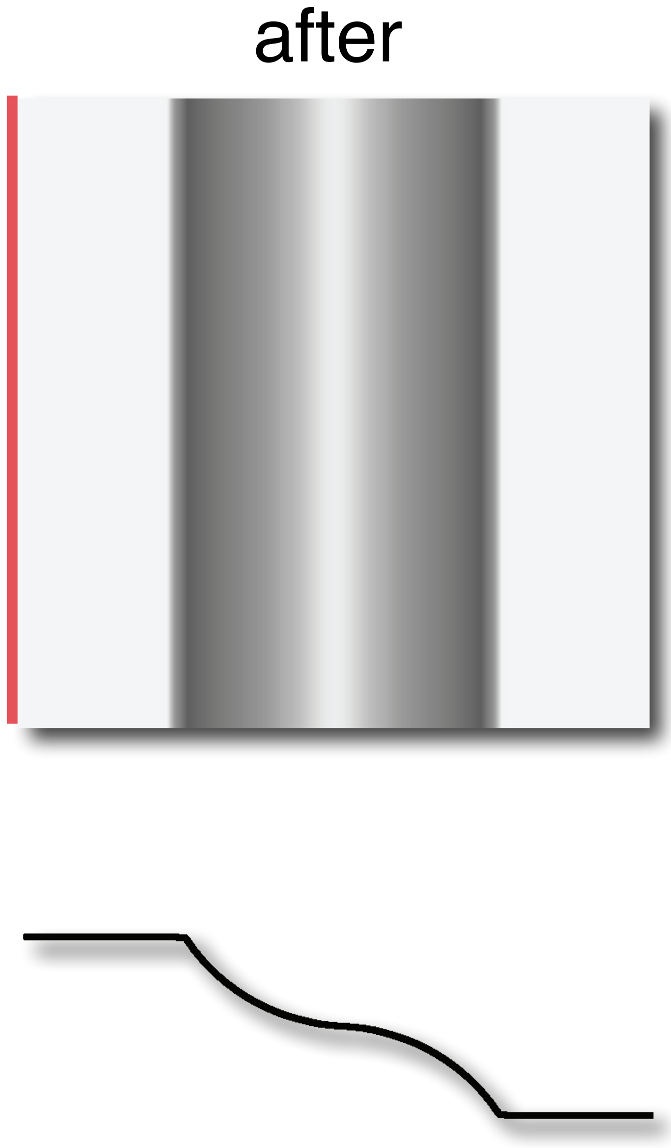


Highlight Darkening Instability

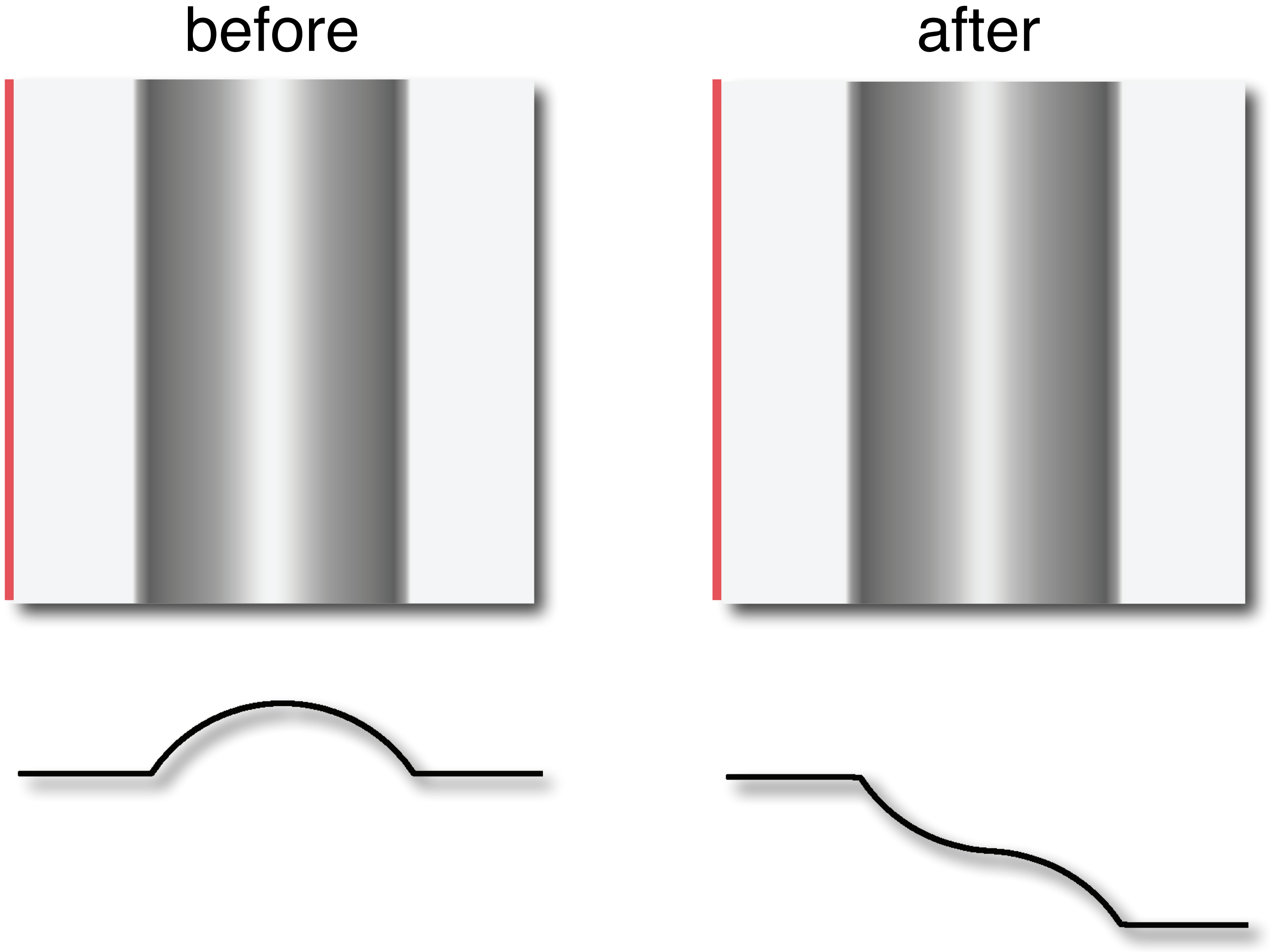


Highlight Darkening Instability

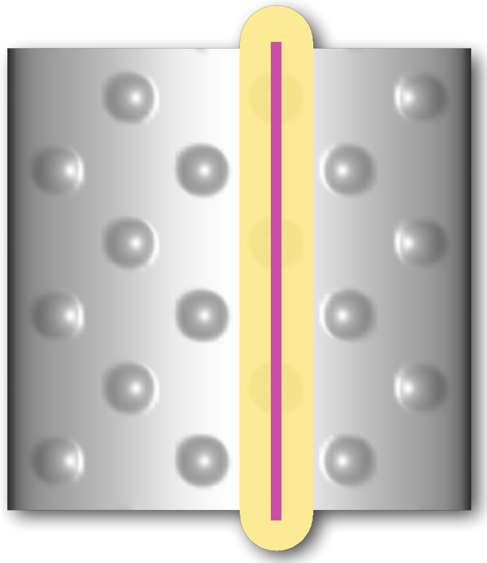




Highlight Darkening Instability



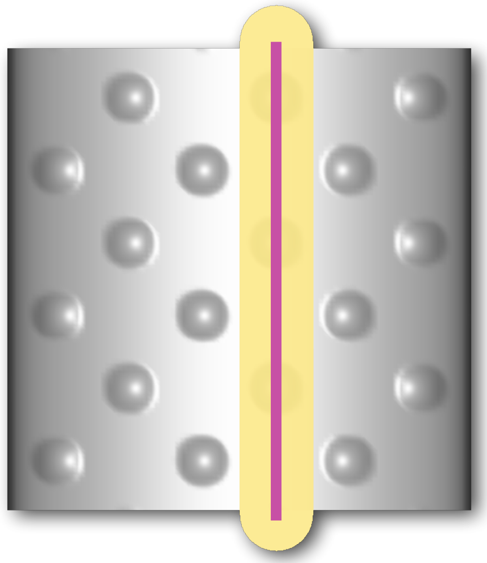
Approach



Approach

Centerline of stroke: rotate surface about the stroke

- stable, predictable



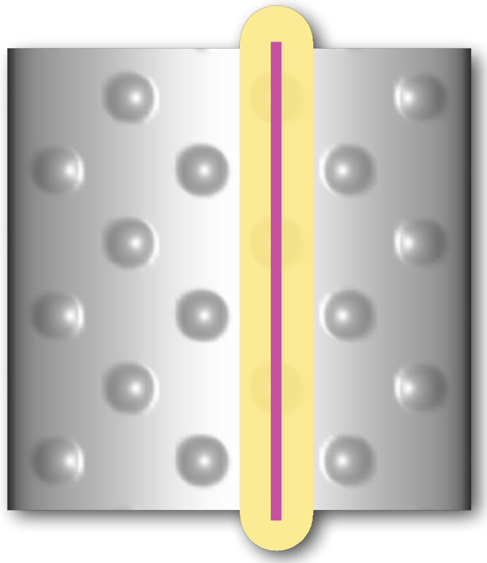
Approach

Centerline of stroke: rotate surface about the stroke

- stable, predictable

Elsewhere: Laplacian Editing Energy

- preserves appearance & shape



Approach

Centerline of stroke: rotate surface about the stroke

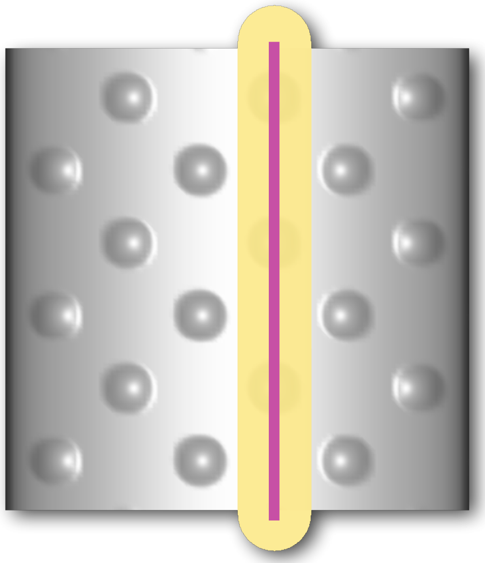
- stable, predictable

Elsewhere: Laplacian Editing Energy

- preserves appearance & shape

Variable vertex weights for our brush parameters

- controllable



Approach

Centerline of stroke: rotate surface about the stroke

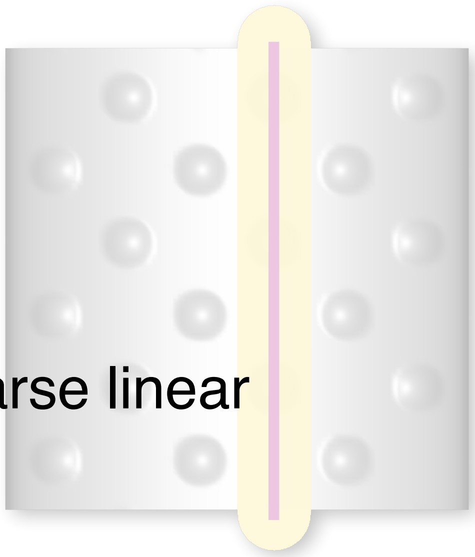
- stable, predictable

Elsewhere: Laplacian Editing Energy

- preserves appearance & shape

Variable vertex weights for our brush parameters

- controllable



Linear Constraints + Quadratic Energy = sparse linear system of equations

Quadratic Energy

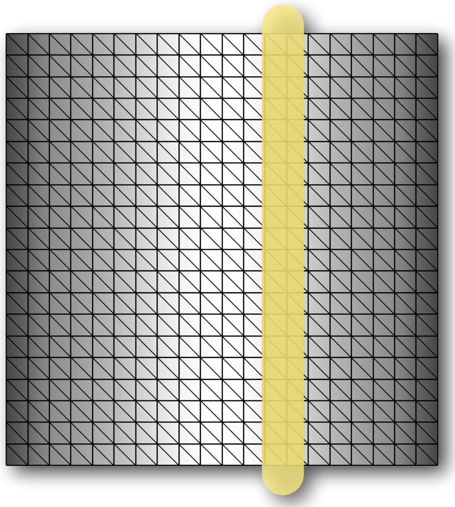
vertices
(degrees of freedom)

$$E(V') =$$

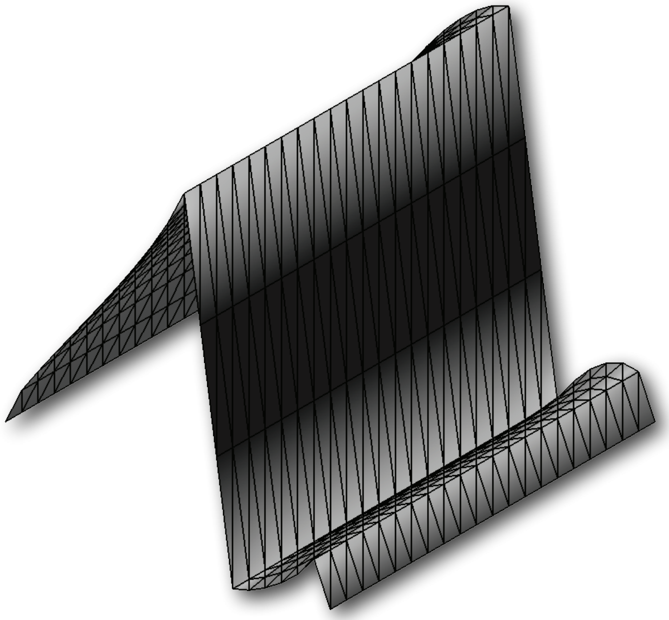
Quadratic Energy

vertices
(degrees of freedom)

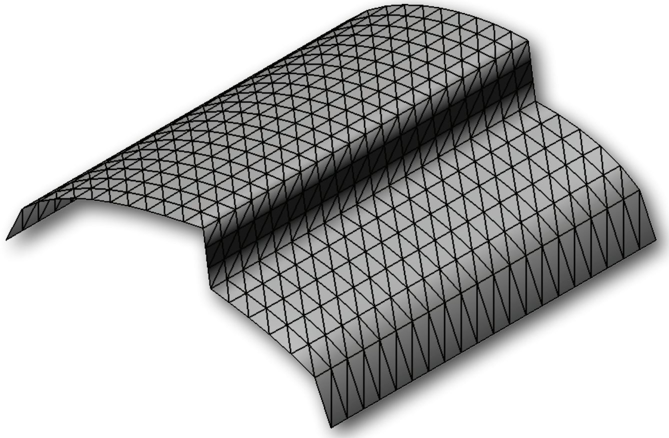
$$E(V') =$$



stroke



height field



xyz

Quadratic Energy

vertices
(degrees of freedom)

$$E(V') =$$

Quadratic Energy

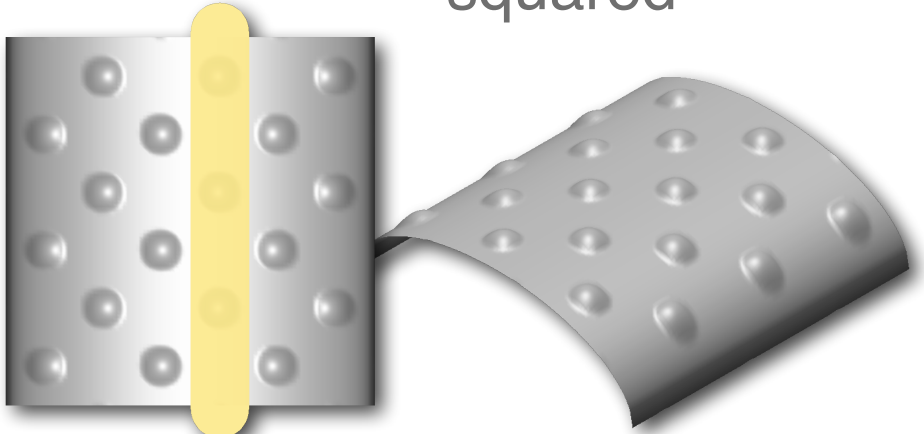
vertices
(degrees of freedom)

$$E(V') = \underbrace{\sum_{i=1}^{n} g(v_i)\|\Delta v_i - \Delta v_i'\|^2}_{\text{difference in laplacian, squared}}$$

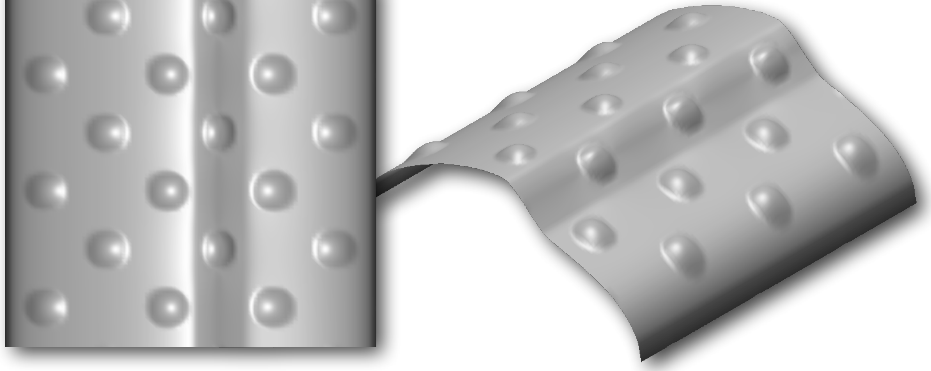
Quadratic Energy

vertices
(degrees of freedom)

$$E(V') = \underbrace{\sum_{i=1}^{n} g(v_i)\|\Delta v_i - \Delta v_i'\|^2}_{\text{difference in laplacian, squared}}$$



before



after

Quadratic Energy

vertices
(degrees of freedom)

$$E(V') = \underbrace{\sum_{i=1}^{n} g(v_i) \|\Delta v_i - \Delta v_i'\|^2}_{\text{difference in laplacian, squared}}$$

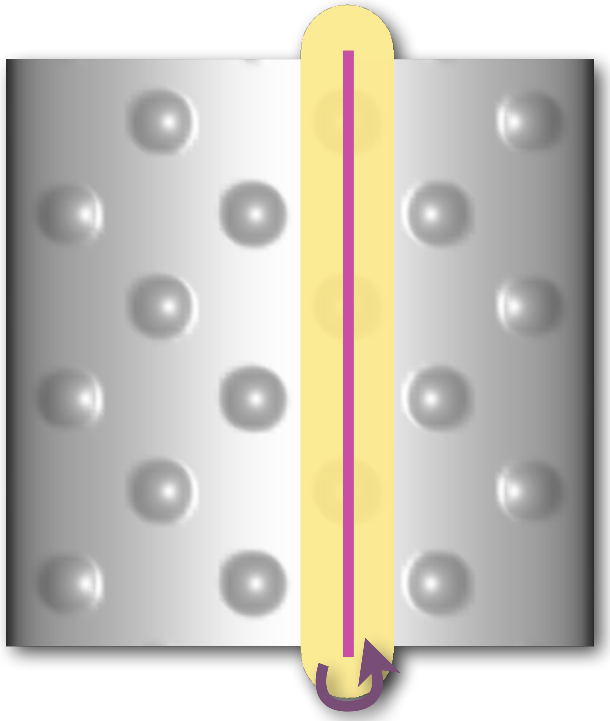
Quadratic Energy

vertices
(degrees of freedom)

constraint rotating the surface
under the stroke

$$E(V') = \underbrace{\sum_{i=1}^{n} g(v_i)\|\Delta v_i - \Delta v_i'\|^2}_{\text{difference in laplacian, squared}} + \overbrace{w_{lsq} \sum_{(i,j)=e\in C} \|(v_i' - v_j') - e^{trg}\|^2}^{\text{constraint rotating the surface under the stroke}}$$

difference in laplacian,
squared

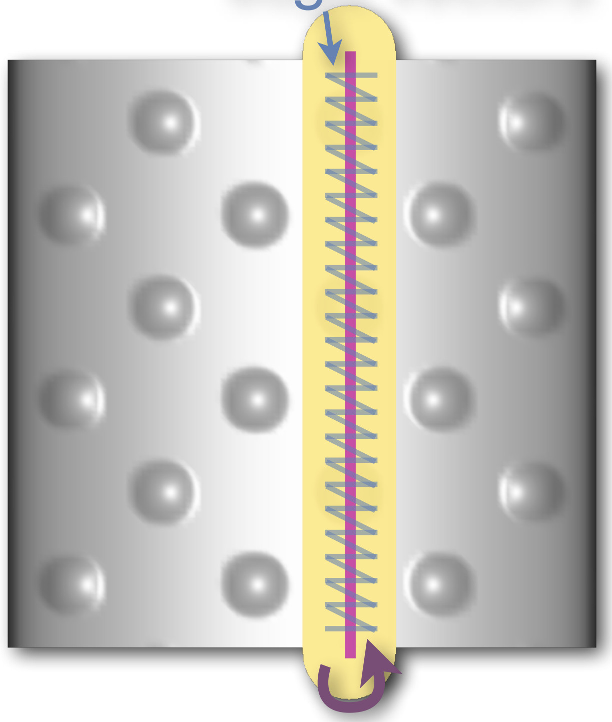


Quadratic Energy

vertices
(degrees of freedom)

$$E(V') = \underbrace{\sum_{i=1}^{n} g(v_i)\|\Delta v_i - \Delta v_i'\|^2}_{\text{difference in laplacian, squared}} + \overbrace{w_{lsq} \sum_{(i,j)=e\in C} \|(v_i' - v_j') - e^{trg}\|^2}^{\text{constraint rotating the surface under the stroke}}$$

edge vectors



Quadratic Energy

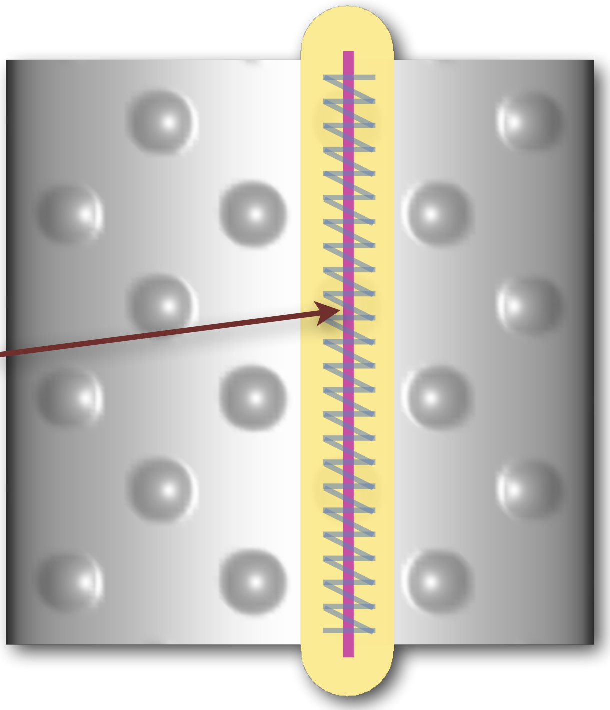
vertices
(degrees of freedom)

constraint rotating the surface
under the stroke

$$E(V') = \underbrace{\sum_{i=1}^{n} g(v_i)\|\Delta v_i - \Delta v'_i\|^2}_{\text{difference in laplacian, squared}} + \overbrace{w_{lsq} \sum_{(i,j)=e\in C} \|(v'_i - v'_j) - e^{trg}\|^2}^{\text{constraint rotating the surface under the stroke}}$$

difference in laplacian,
squared

image-plane constraints
(not shown in E)



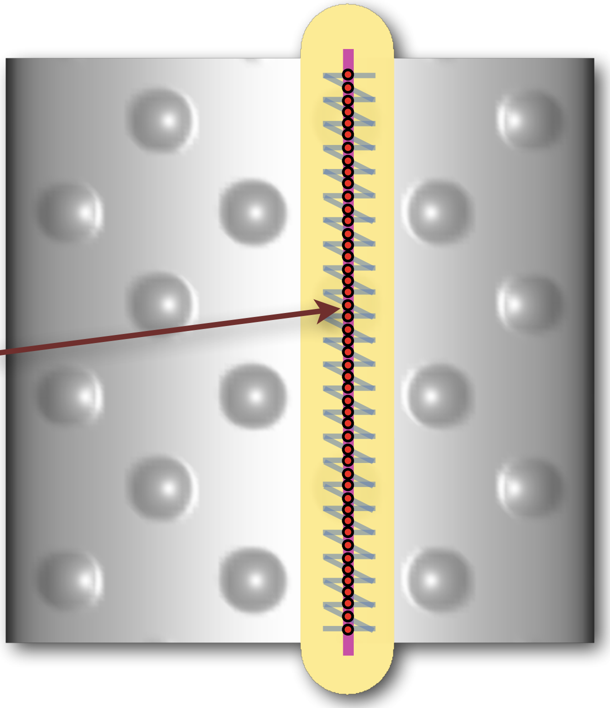
Quadratic Energy

vertices
(degrees of freedom)

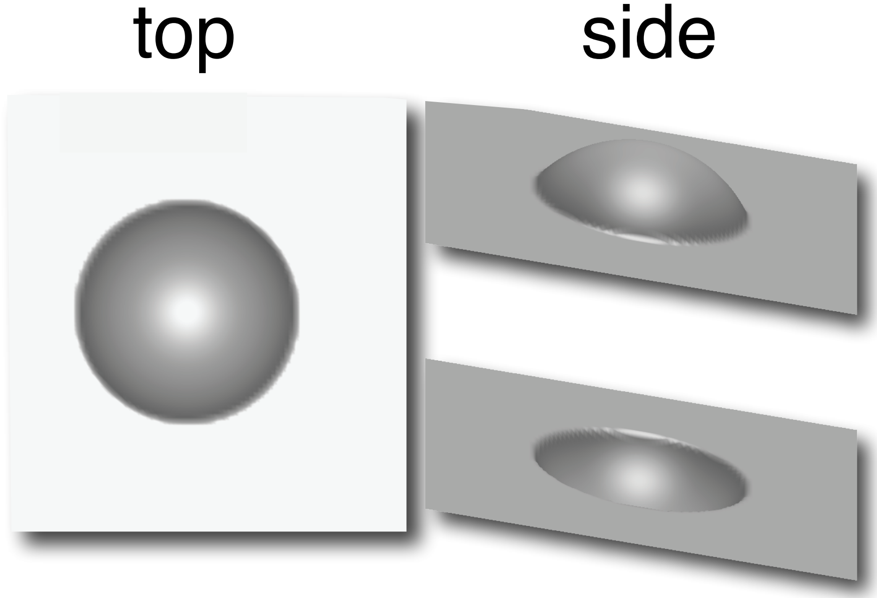
constraint rotating the surface
under the stroke

$$E(V') = \underbrace{\sum_{i=1}^{n} g(v_i)\|\Delta v_i - \Delta v_i'\|^2}_{\text{difference in laplacian, squared}} + w_{lsq} \overbrace{\sum_{(i,j)=e\in C} \|(v_i' - v_j') - e^{trg}\|^2}$$

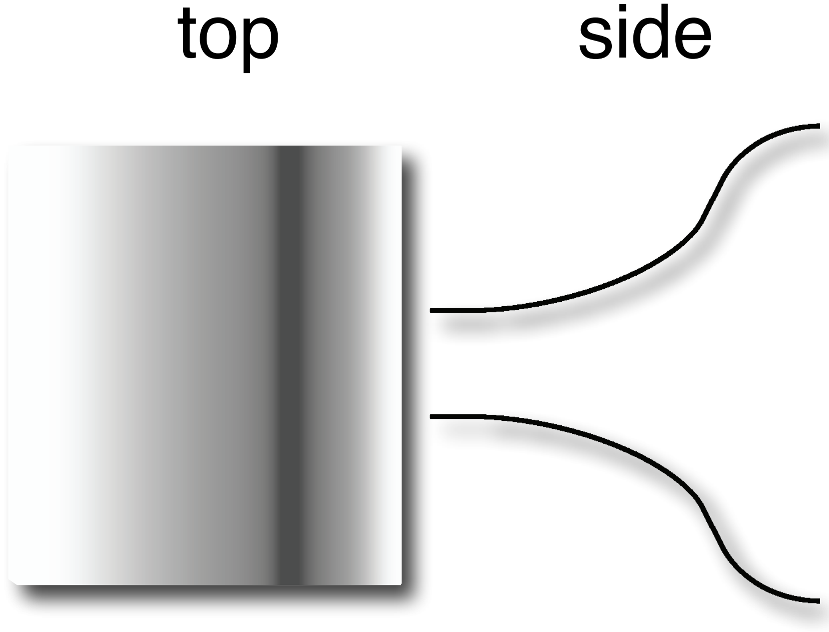
image-plane constraints
(not shown in E)



Flip Ambiguity

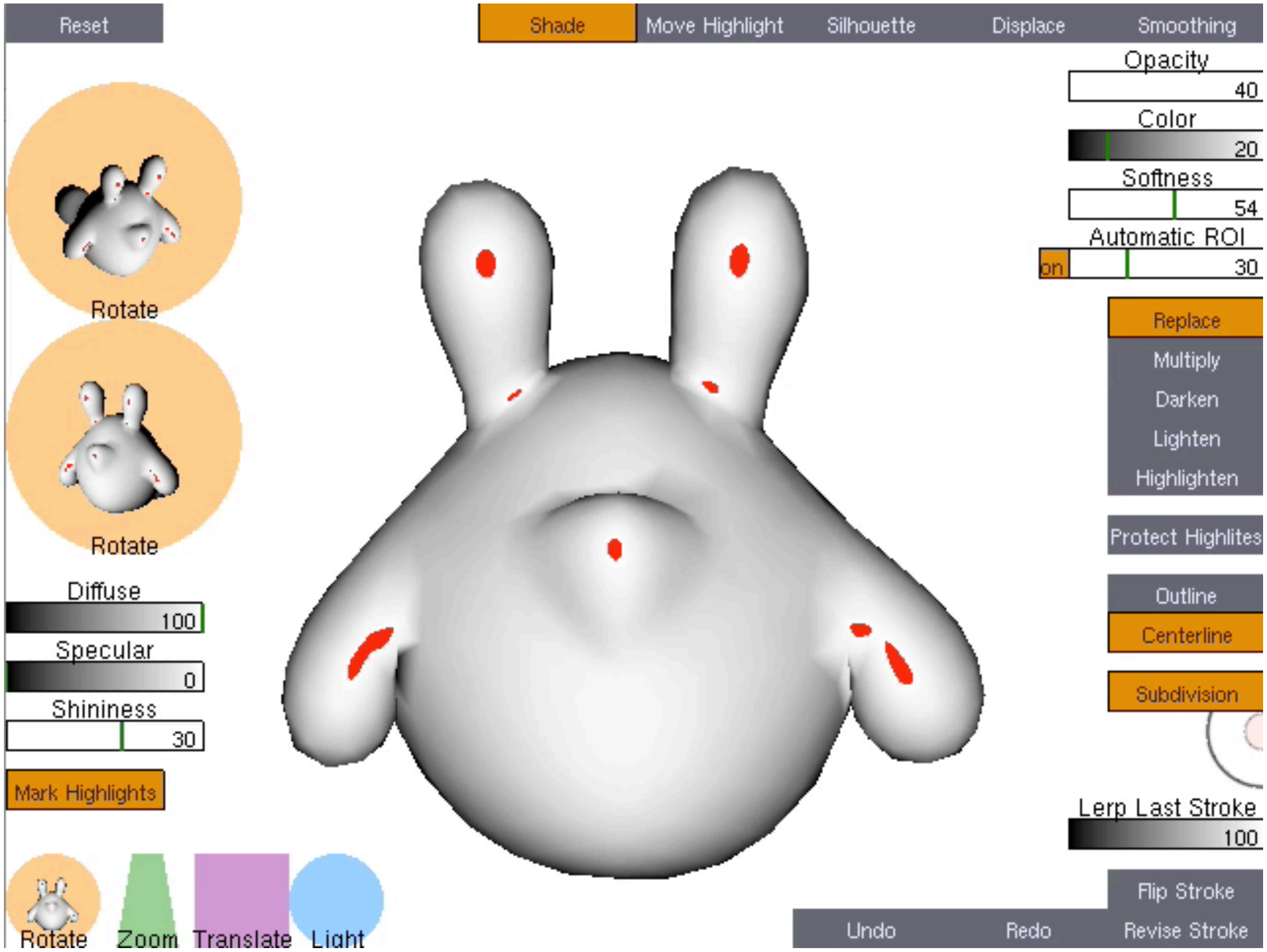


Concave/Convex ambiguity

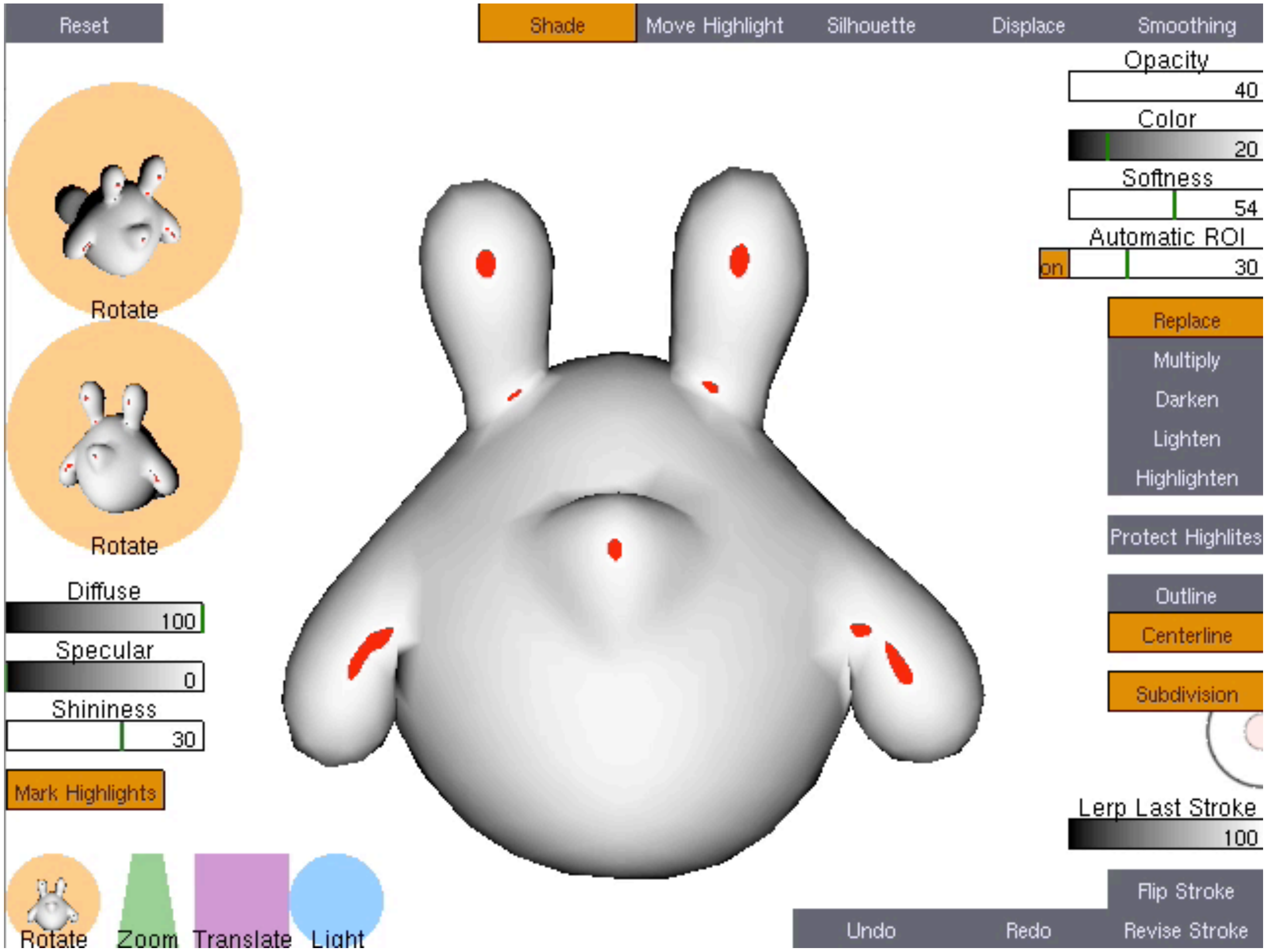


Slope ambiguity

Summary


Reset
Shade
Move Highlight
Silhouette
Displace
Smoothing
Opacity
40
Color
20
Softness
54
Automatic ROI
on
30
Replace
Multiply
Darken
Lighten
Highlighten
Protect Highlites
Outline
Centerline
Subdivision
Lerp Last Stroke
100
Flip Stroke
Undo
Redo
Revise Stroke
Rotate
Rotate
Diffuse
100
Specular
0
Shininess
30
Mark Highlights
Rotate
Zoom
Translate
Light

Summary


Reset
Shade
Move Highlight
Silhouette
Displace
Smoothing
Opacity 40
Color 20
Softness 54
Automatic ROI on 30
Replace
Multiply
Darken
Lighten
Highlighten
Protect Highlites
Outline
Centerline
Subdivision
Lerp Last Stroke 100
Flip Stroke
Undo
Redo
Revise Stroke
Rotate
Rotate
Diffuse 100
Specular 0
Shininess 30
Mark Highlights

Limitations

Shading requires expertise.

Speed: modifications aren’t local.

Lack of integration with sculpting tools and silhouette editing.

Highlight control is limited.