Sketching in augmented reality

Nicole Feng

Spring 2019

Change of plans…

- Initially I planned to do a "then & now" re-photography app. The goal was to use augmented reality combined with other computer vision techniques to align photos and explore the results.


https://www.swiss-miss.com/wp-content/uploads/2010/02/4355405531_e9daccfdc6-480x326.jpg


http://www.bluedotmagazine.com/2016/05/18/london-during-the-blitz-then-and-now-photographs/


Wikipedia

- However, many technical difficulties, and way too many moving parts for <10 weeks.

Augmented reality basics

- Device has to build a map of the environment, then determine its location within that environment.
- Virtual images are determined by aligning the virtual camera's pose with the real camera's pose.
- Google ARCore SDK does this automatically using a process it has patented as "concurrent odometry and mapping"
  - At a high level, it detects feature points from multiple images and uses them to build a 3D visual representation of the environment.
  - Combined with information from the IMU (inertial measurement unit), camera pose is estimated.
- ARCore obscures how many functions work.

Sketch-based modeling

- In contrast to CAD, sketch-based modeling turns 2D sketches into 3D models.

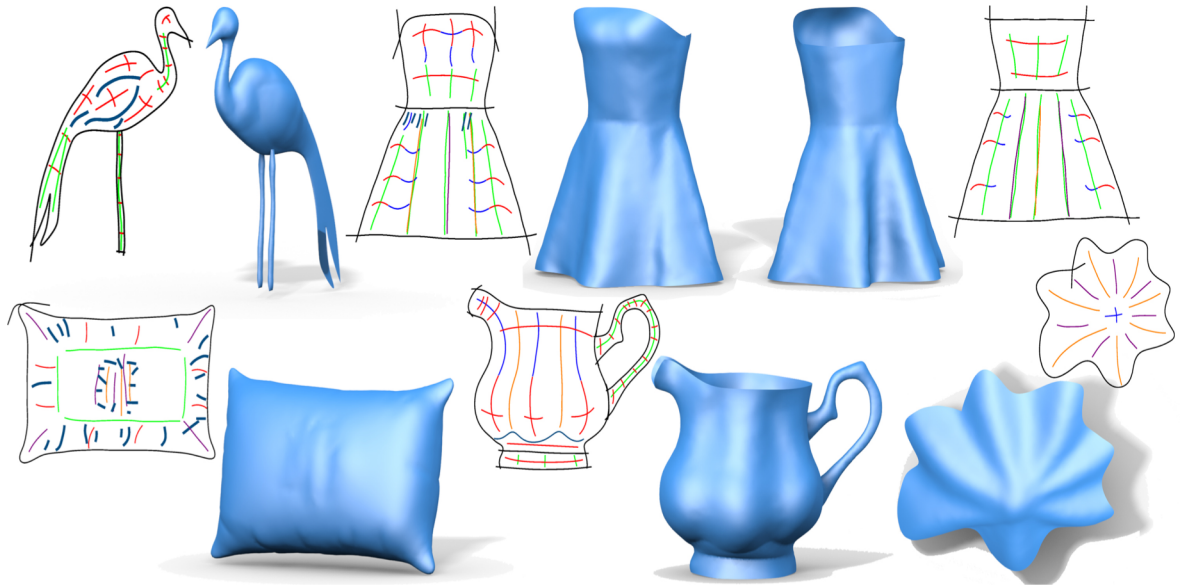
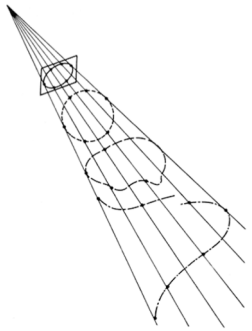


Image from Li et al, *BendSketch,* 2017.

- With AR, user can also view the resulting model within their current context.

- 2D ➔ 3D: infinite number of possibilities



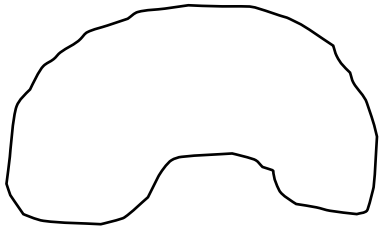
All 3D curves map to the ellipse
in the image plane.

- In VR, users can sculpt/draw in 3D, and ideally in AR as well (although difficulties with depth and hand-tracking.)
- But user may not know exactly where in 3D space they want points to go, or they want something quick & convenient with little overhead.
- Follow *Teddy: A Sketching Interface for 3D Freeform Design* by Igarashi, Matsuoka, and Tanaka (1999), which uses an inflation-based approach.

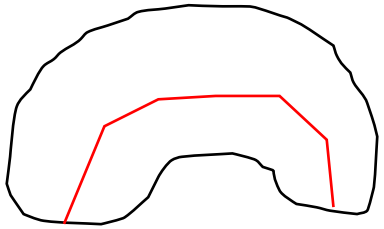
Teddy (1999) overview

Simple idea:

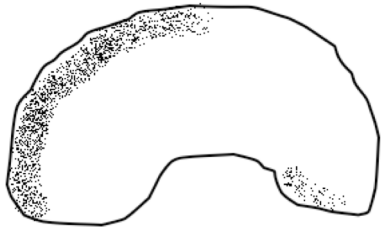
We have a 2D shape:



Determine its "skeleton":

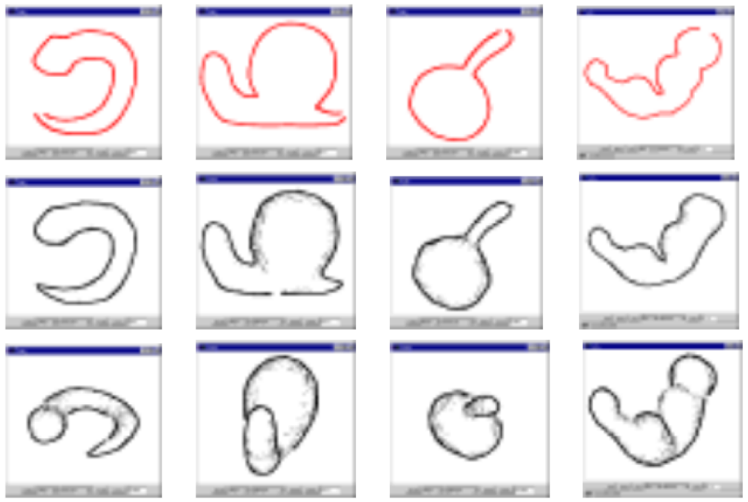


Inflate:



Teddy (1999) overview

- User draws 2D closed curves (curves are automatically closed otherwise.)
- The closed region is "inflated" in both directions. Wide areas become fat, and narrow areas become thin. A free-form shape is generated.



a) snake b) snail c) cherry d) muscular arm

Figure 5: Examples of creation operation (top: input stroke, middle: result of creation, bottom: rotated view).

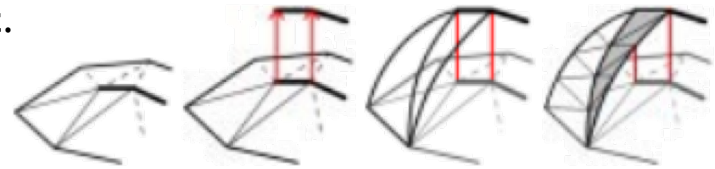
- Cons: models must have spherical topology, no self-intersecting curves.

Algorithm overview

a) initial 2D polygon b) result of CDT c) chordal axis

d) fan triangles e) resulting spine f) final triangulation

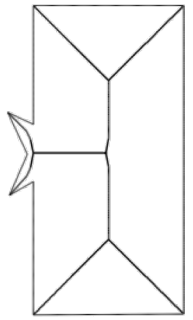
- Start with a closed 2D polygon.
  - Resample and make all edges some unit length.
- Determine the "spine" the polygon using the *chordal axis*.
  - Perform constrained Delaunay triangulation (CDT) on the polygon.
  - Obtain the chordal axis by connecting the midpoints of internal edges.
  - Prune insignificant branches.
  - Re-triangulate the mesh.
- Elevate the vertices of the spine by an amount proportional to their distance from the polygon.
  - Elevate each vertex on the spine by a distance proportional to the average distance between the vertex and the external vertices directly connected to it.
  - Convert each internal non-spinal edge to a quarter oval, and elevate along with the spine.
  - Sew together the neighboring elevated edges.
  - Copy the elevated mesh to the other side to make the mesh closed and symmetric.
  - Remove short edges and small triangles.



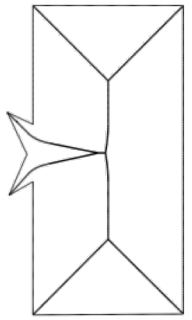
a) before b) elevate spines c) elevate edges d) sew elevated edges

Skeleton construction - CDT

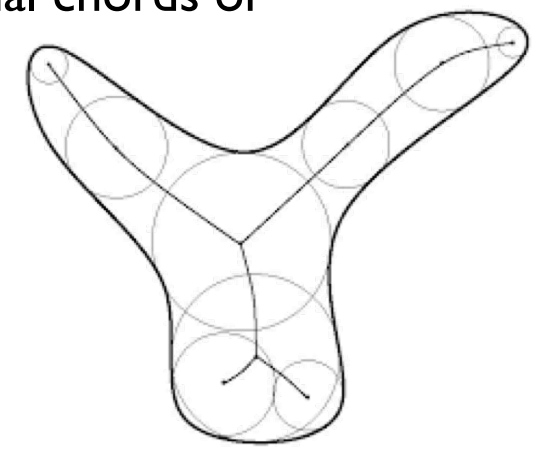
- A *maximal disc* is a circle contained in the interior of the shape, that is tangent to its boundary at at least 2 points.
- A *maximal chord of tangency* of a maximal disc is a chord that
  - (i) connects two points of tangency (of the disc with the shape boundary)
  - (ii) at least one of the two arcs subtended by the chord has no points of tangency.
- The *chordal axis transform* is the set of all pairs (p, d), where either:
  - (i) p and d are the midpoint and half the length of a maximal chord of tangency, resp.
  - or, (ii) p and d are the center and radius of a maximal disc with three maximal chords of tangency that form an acute triangle.



(d) CAT skeleton



(e) MAT



Skeleton construction - CDT

- In the discrete case, *maximal discs* are replaced by *empty circles* that pass through >= 3 polygon vertices.
  - *Empty circles* do not contain any vertex of the polygon that are *visible* to the any 2 vertices on the circle (visible = the line segment joining the 2 vertices is entirely in the interior of the polygon.)
- *Maximal chords of tangency* are replaced by the line that joins 2 non-neighboring vertices of the polygon iff an *empty circle* passes through both these vertices.
  - This is equivalent to a Constrained Delaunay Triangulation (CDT.)
  - A Delaunay Triangulation is a decomposition of a polygon into triangles s.t. the circumcircle of each triangle is empty.
  - Results in 3 types of triangles: *terminal, sleeve,* and *junction*

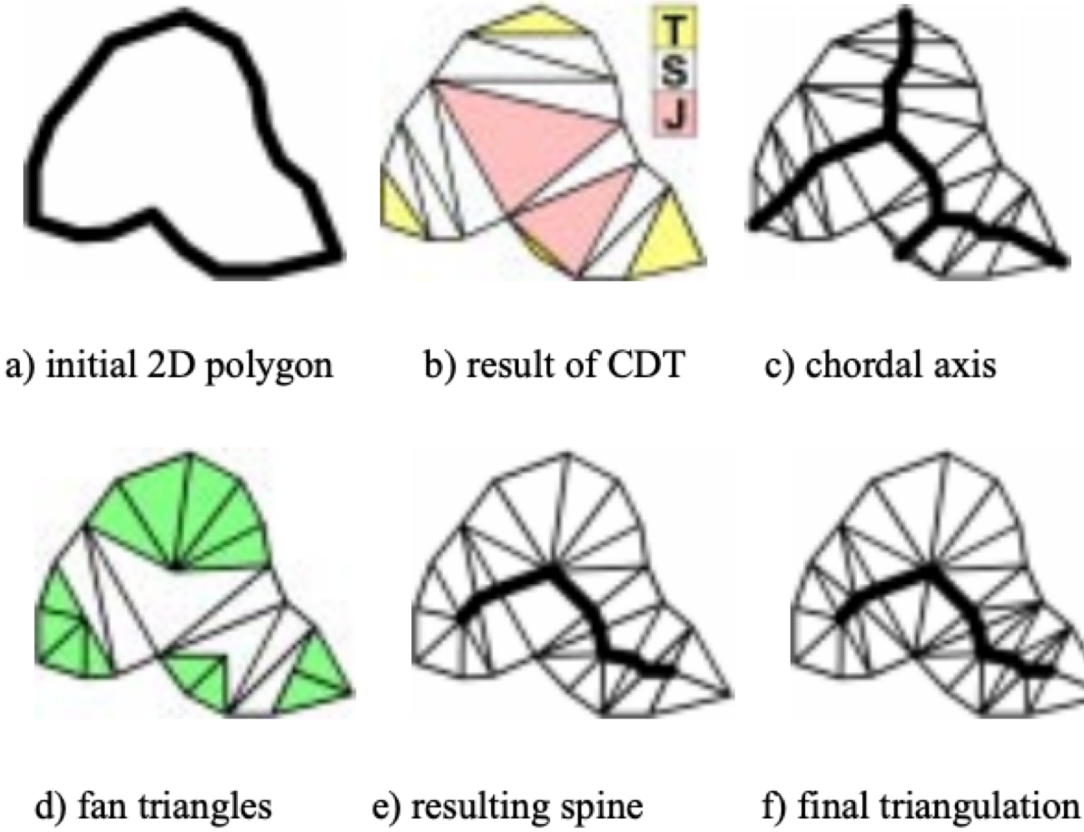
Skeleton construction - CDT

- After the CDT, we obtain "terminal" (2 boundary edges), "sleeve" (1) and "junction"(0) triangles.

- The chordal axis is obtained by connecting the midpoints of the interior edges.

- However, we then want to obtain an "inflatable-friendly" triangulation.

- We remove insignificant branches and round the extremities via a "fanning" process.



Skeleton construction – fanning & pruning

- We want the final mesh to be as smooth as possible.
- We want to round the extremities of each branch (which are indicated by terminal triangles.)
- We do this by fitting a hemispherical cap at the end of each branch, and determining the center around which to fan the triangles.

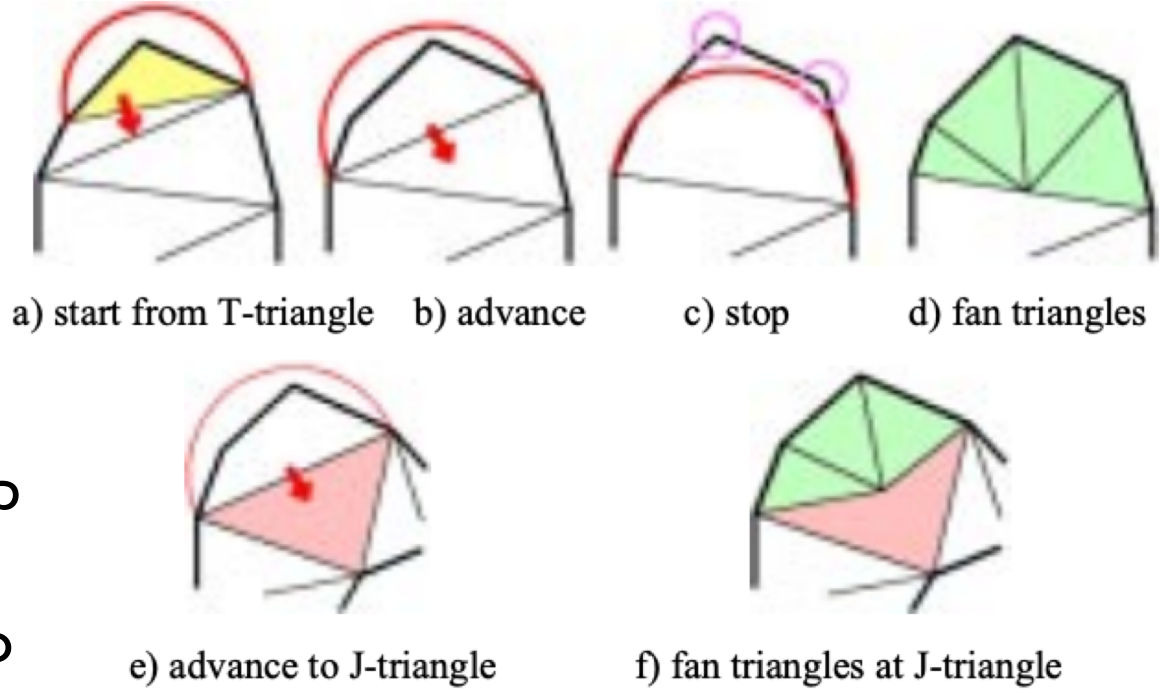
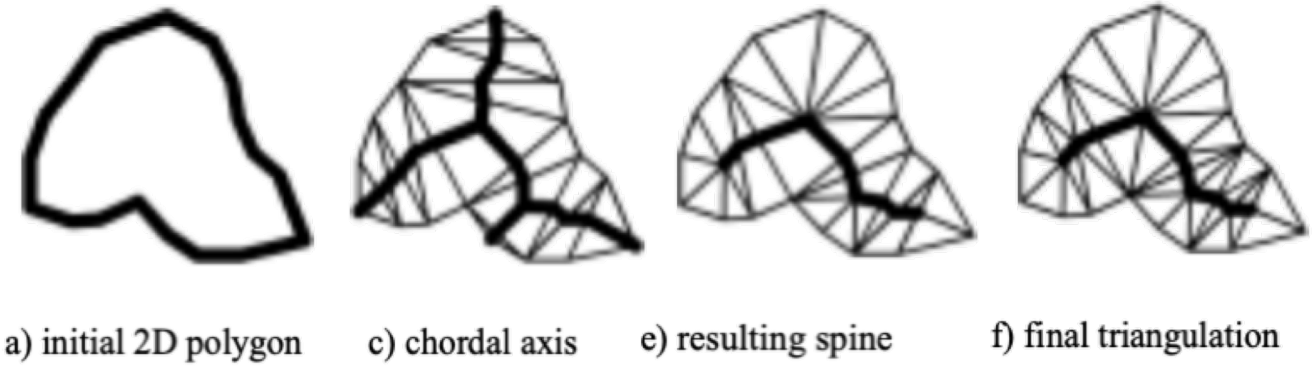


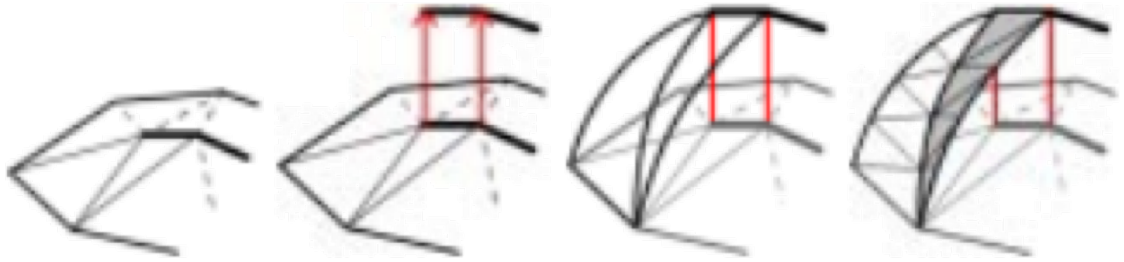
Figure 14: Pruning.

Mesh construction – re-triangulation

- After fanning and pruning, we connect the resulting internal points and midpoints of internal edges to obtain the spine we want.
- We then construct the final triangulation such that all interior edges are between the spine and the boundary.
- First we elevate the spine a distance proportional to the average length of its adjacent edges.
- Then we sew together each triangle with several smaller triangles, and elevate them such that each edge approximate a quarter-ellipse.

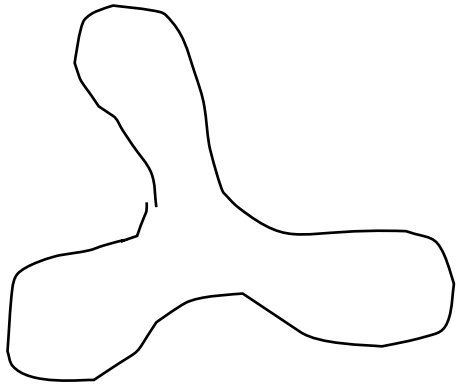


a) initial 2D polygon c) chordal axis e) resulting spine f) final triangulation

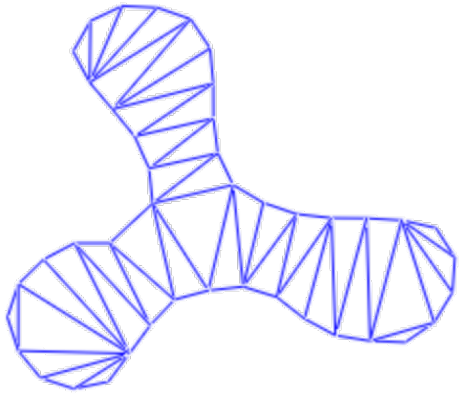


a) before b) elevate spines c) elevate edges d) sew elevated edges

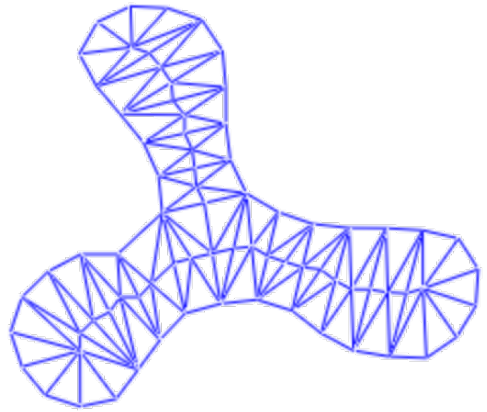
Draw in 2D:



CDT:



After pruning and re-triangulation:



Final mesh:

Implementation

- Implement halfedge data structure for performing manipulations on mesh
- Implement constrained Delaunay triangulation
- Implement drawing interface, which:
  - Takes in 2D drawing input
  - Resamples drawn curve
  - Performs CDT
  - Prunes insignificant edges from the chordal axis
  - Re-triangulates
  - Elevates the spine
  - Elevates each of the interior edges, and stitches them together as a mesh
- Have ARCore detect planes on which to place the newly created object.
- Implement mesh manipulation (drag and drop)
- Optionally support other mesh operations.

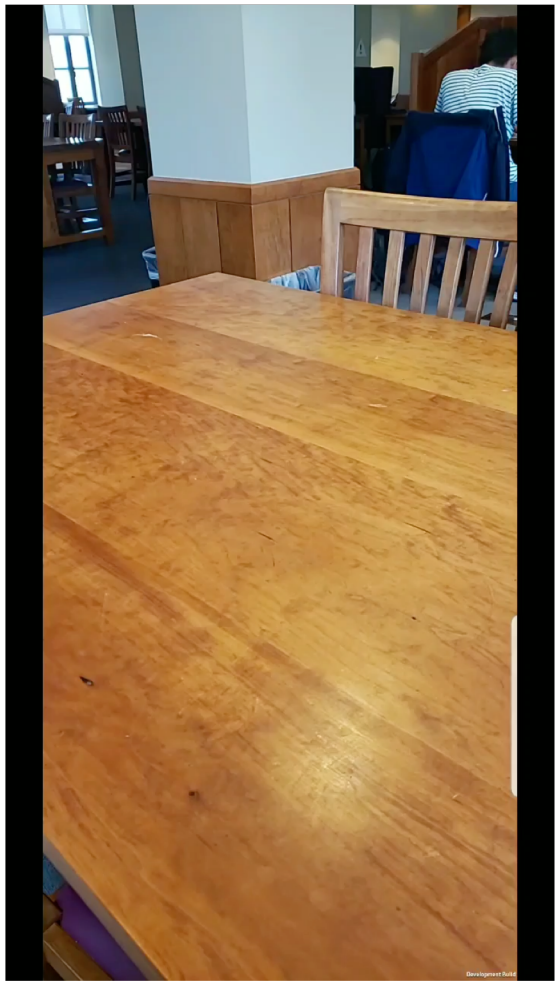
Other implementation details

- Google ARCore comes with environmental light estimation, so we use this in our shaders to adjust the lighting of the mesh.
  - Source code is not available, but my guess is that it computes some average of pixel intensities across frames to provide the light estimates.

- ARCore also comes with a plane detection feature, which we use to anchor our mesh.

Implementation

Code available at: https://github.com/nzfeng/CS174



Currently supported operations: mesh creation, pinch enlarge / shrink

Cons: too slow on smartphone

Goals of project

- Implement a sketch-based modeling algorithm
- Develop app that demonstrates using AR to extend an existing application
- Develop app using newly-released AR SDK for smartphone, where there is little existing work and sparse documentation / example usage.

Future directions

- Support other types of drawing and modeling.
    - Other types of inference, for example from orthographic projections
- Allow further manipulation of existing objects, such as cutting, extrusion, etc. although operations are too slow on my smartphone.

References

- H.G. Barrow and J.M. Tenenbaum, *Interpreting Line Drawings as Three-Dimensional Surfaces,* 1980.
- Igarashi, Matsuoka, Tanaka, *Teddy: A Sketching Interface for 3D Freeform Design,* 1999.
- Lakshman Prasad, *Morphological Analysis of Sha*pes, 1997.
- Google patent: https://patentimages.storage.googleapis.com/4d/af/08/b9351479700bcf/US20170336511A1.pdf
- https://developer.android.com