Double Oriented Sculptures

The Goal (or what do we want to see?)

We would like to automatically construct 3D sculptures that have two faces. Viewing them from a certain angle gives one 'flat' image while viewing from a perpendicular angle, gives another. Each of the two viewpoints, should 'fool' the eye into thinking it is the only image there. So every pixel, which is visible in the direction of view, should be part of the image or look like it belongs. Any viewpoint other than the two, can show a deformed set of pixels.

Example: https://www.youtube.com/watch?v=JGMEgDLZ4s0

The Concept (or why do we see it?)

The sculpture is formed by spreading the outlines of two 2D images, one into the space spanned by the width of the other. The objective is to 'hide' one image's profile behind the profile of another. Once all pixels of each image are occluded, from the viewpoint of the other, the wanted optical illusion hatches.

This trick works on the human eye because of the way our visual perception works. Since we cannot see behind objects or inside their interiors, we do not have full access to the 3-dimensional information in front of us. Thus, by viewing the sculpture from the correct viewpoint, we are fooled to think it is all that's there.

The pair of human eyes are designed to approximate 3D vision, a visual perception effect named 2.5D. The 3D environment is projected onto two 2D planes, the retinas. The images are used to construct the 3D environment by adding depth perception, from different depth cues. In our case, the sculpture's frame is condensed and thin enough to daze these cues so that the depth difference is not detected.

The Approach (or how would we do it?)

Theoretically, given two perpendicular 2D images, we would like to 'push' inwards lines of the one to be obscured by the other. The profile of these image parts should form shapes of the other image. The bent lines and the created shapes get stationed at some depth from their original place. Thanks to the mentioned characteristics of the human eyesight, the overall images should not be deformed.

Practically, this is sustained by projecting each image onto the cylinder created by the other cloned in 3D space. Parts of each cylinder not hit by the perpendicular image

are removed. In order for images to be projected fully, they must have the same height. Otherwise the parts in the difference will be projected onto nothing. Projecting from both directions gives the outlines of the images as points in space. Next step towards getting an actual sculpture is to connect the points, ideally giving a wire-like curvature.

We should traverse the points of each image, in an order approximating the shape, i.e. visit each point once, via the shortest path, giving two curvatures that can be united.

The Main Idea (or how did we do it?)

1. **Data as Bitmaps**

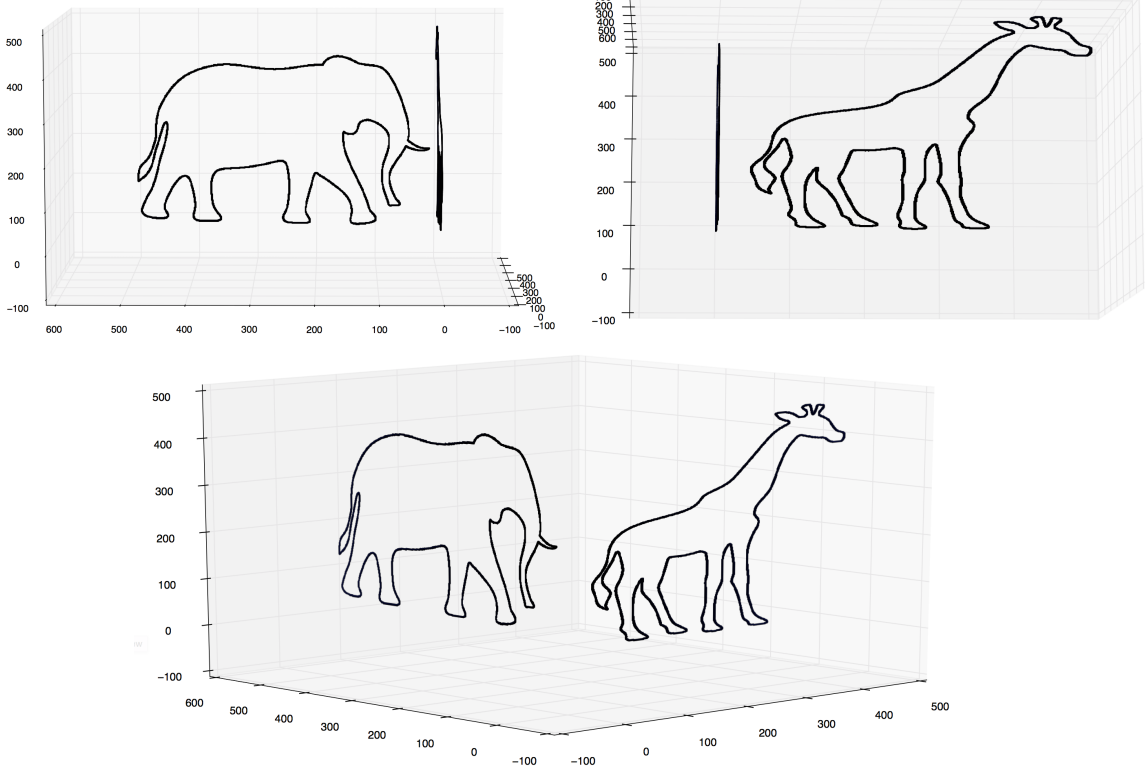
   A general image given as an input needs to be processed before it can be used for creating these unique sculptures. A general image should be converted into an outline only, to give best results.

   After rendering only the contour of the image, we define its bitmap:

$$\forall\, i,j\ pixel\ <\ \delta\colon\ I[i,j] = 1$$
$$\forall\, i,j\ pixel\ >= \delta\colon\ I[i,j] = 0$$

   For a gray-scale threshold δ, usually 0.5 is a good rule of thumb. Lower will filter pale parts (smudges or water marks) and height will be needed for poor quality images, where some of the lines are blurred.

   This rule of thumb gives the smoothest outline. Pixels slightly lighter than black, probably blurred angles are added. On the other hand, noise of very pale pixels around the outline is ignored. The two desired images are placed perpendicular to each other:

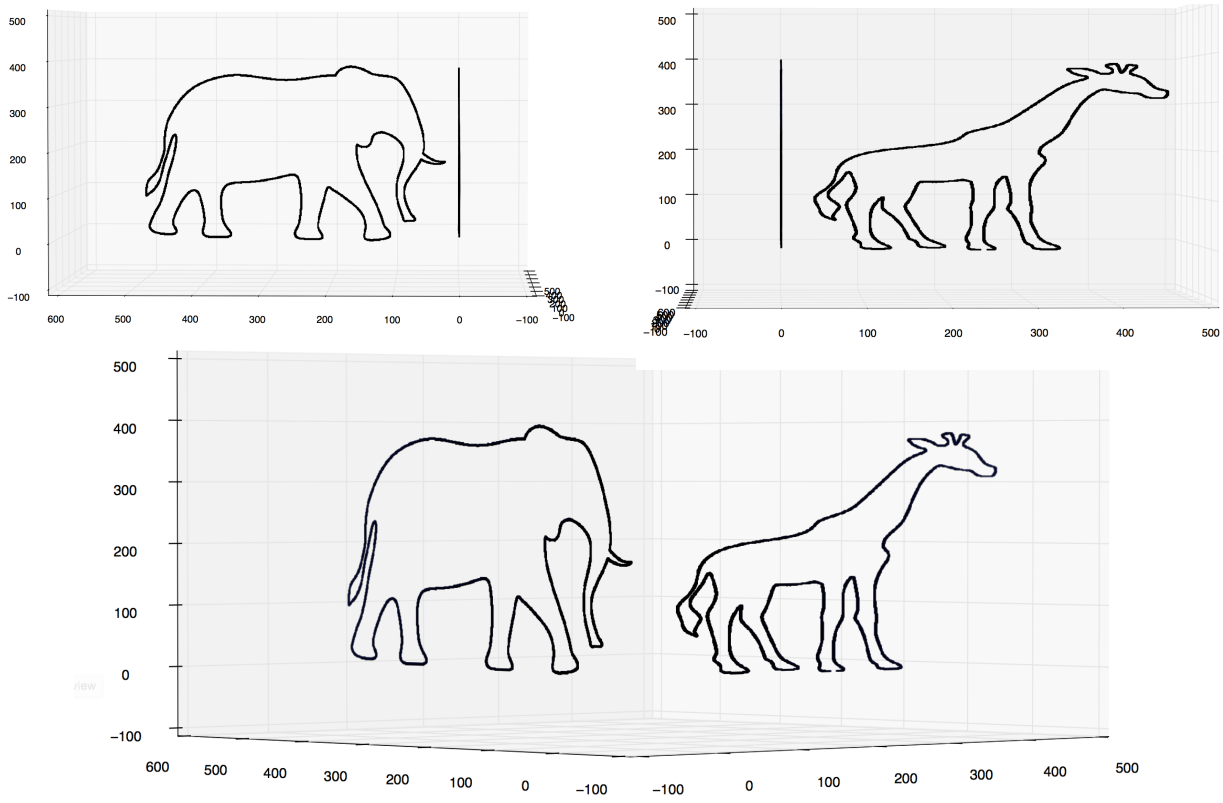


2. Alignment

It is quite prominent in the previous model, how the images have a difference in their position as well as their height (the number of rows actually containing data). As said, this will prevent the sculpture from showing the images fully.

Firstly, we would like both images to “stand on the ground”. So we start filling the rows from the first one containing any pixel. This means overriding each row from the bottom, starting at the first line of data.

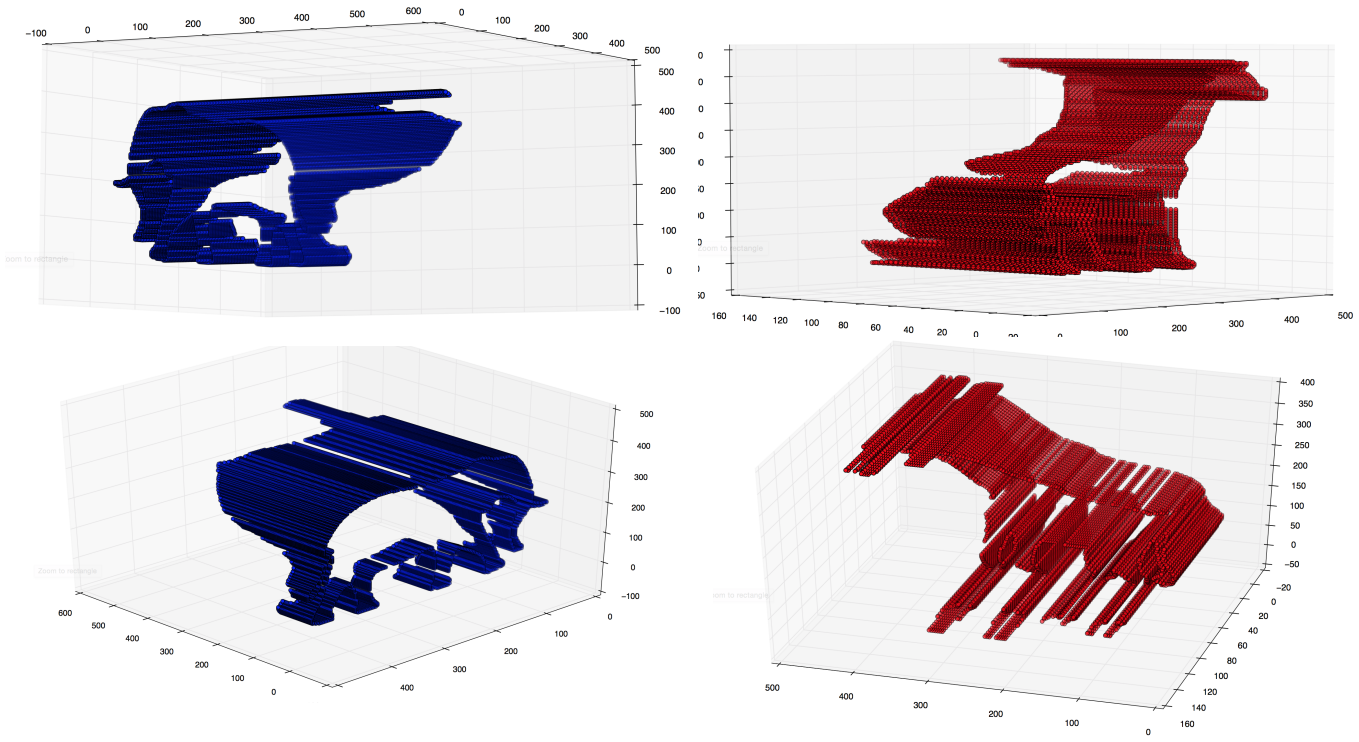
Secondly, we would like to scale the taller image down to the same height of the other. So we remove the difference of rows, dispersed evenly along the image. This means dividing the image height into as many groups as the difference and removing the first row of every group.



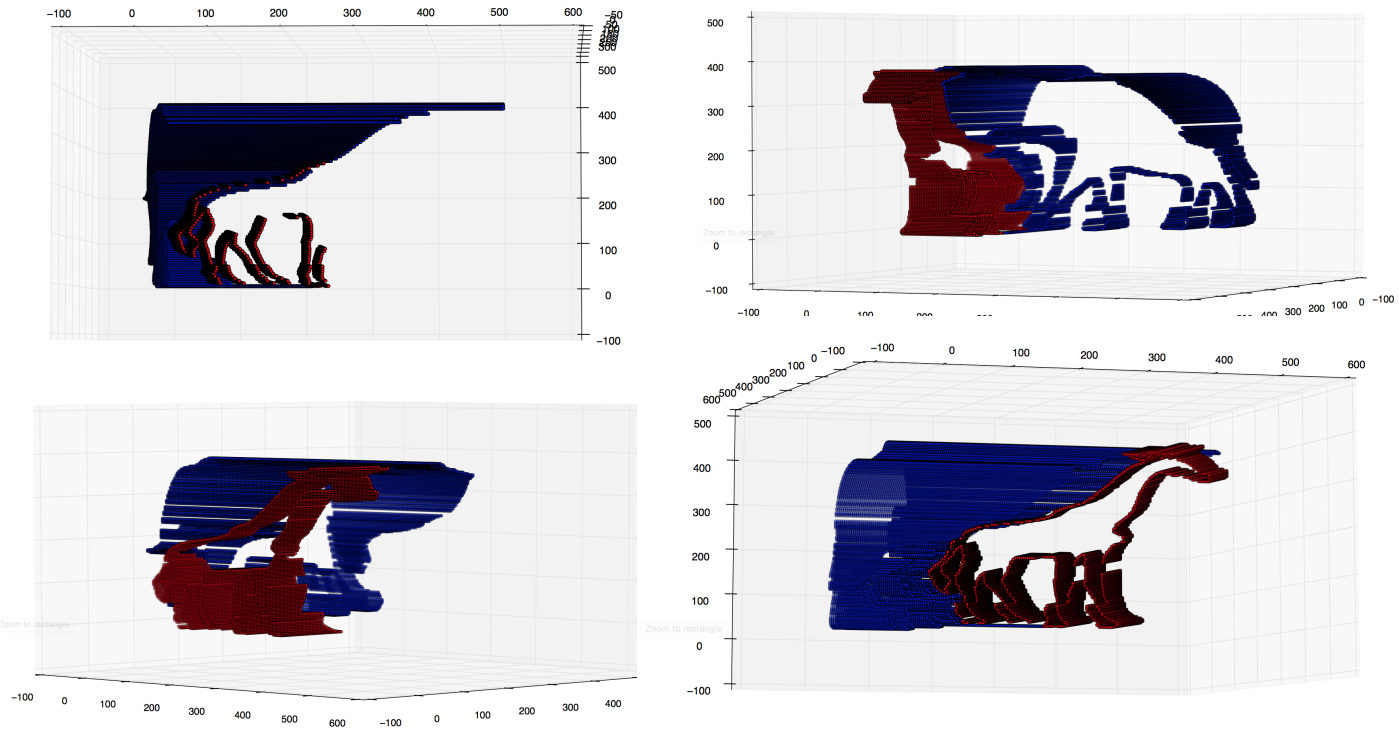
3. Cylinders

In order to illustrate the effect we are seeking, we look at the cylinders created by cloning each 2D image into the space spanned by the width of the other. Since they are perpendicular, the profile of the cylinder is the surface on which we ‘draw’ the other image, by their intersection.

Here are the layers of each 2D image, perpendicular to one another, up to the point of ‘hit’ with the other’s profile:



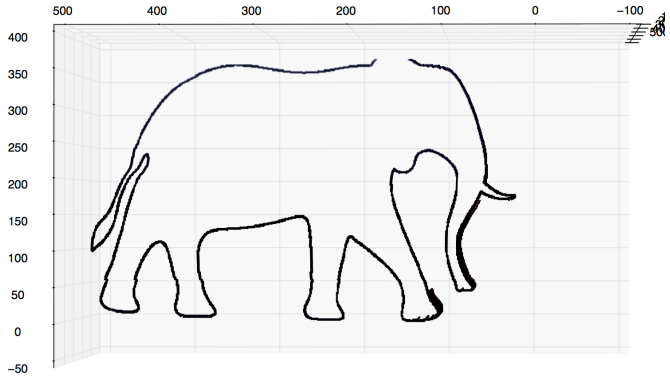
Cylinders combined, up to the surface of intersection:

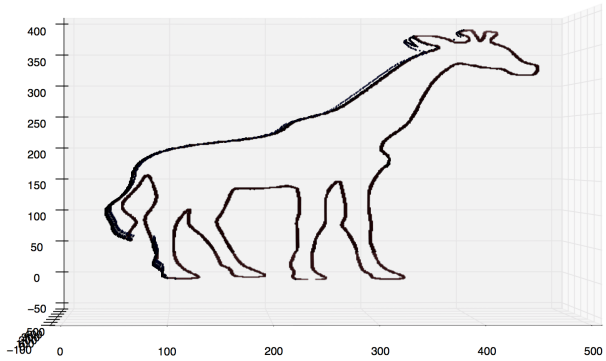


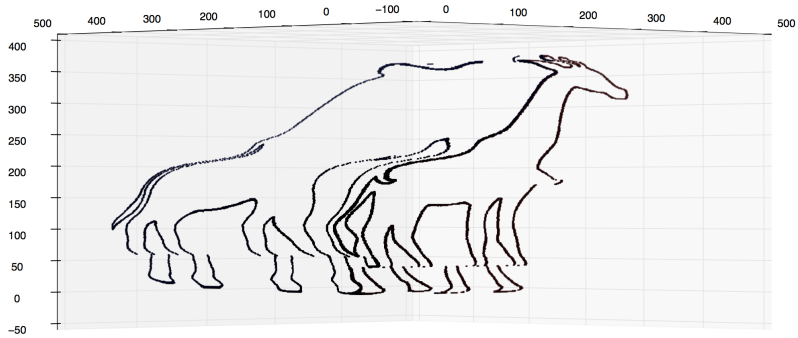
4. Projection

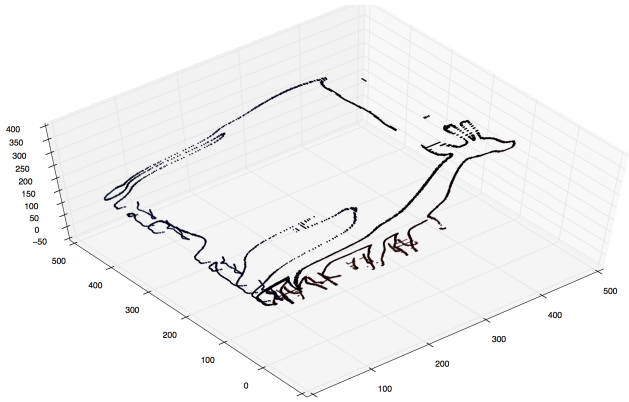
Finally, we get to the formation of one 3D image from the two 2D. As partly explained earlier, the projection means sending one image's pixels backwards until hitting the surface created by the cylinder of the other image. Doing the same from the perpendicular direction too, engraves both the images in the spanned 3D space.

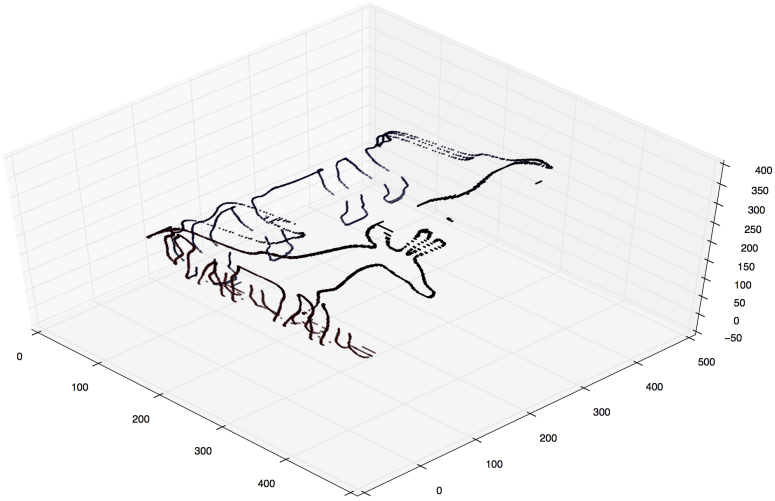
Results from the image viewpoints as well as the connection angle and above:

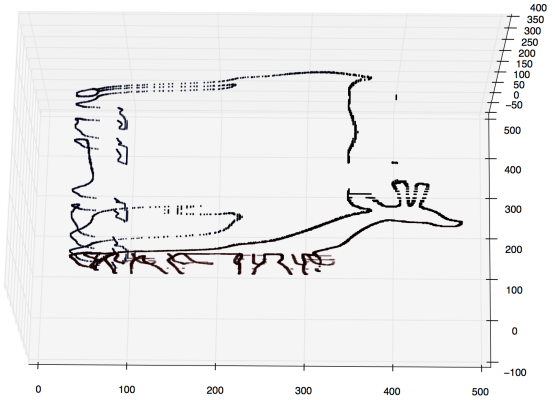












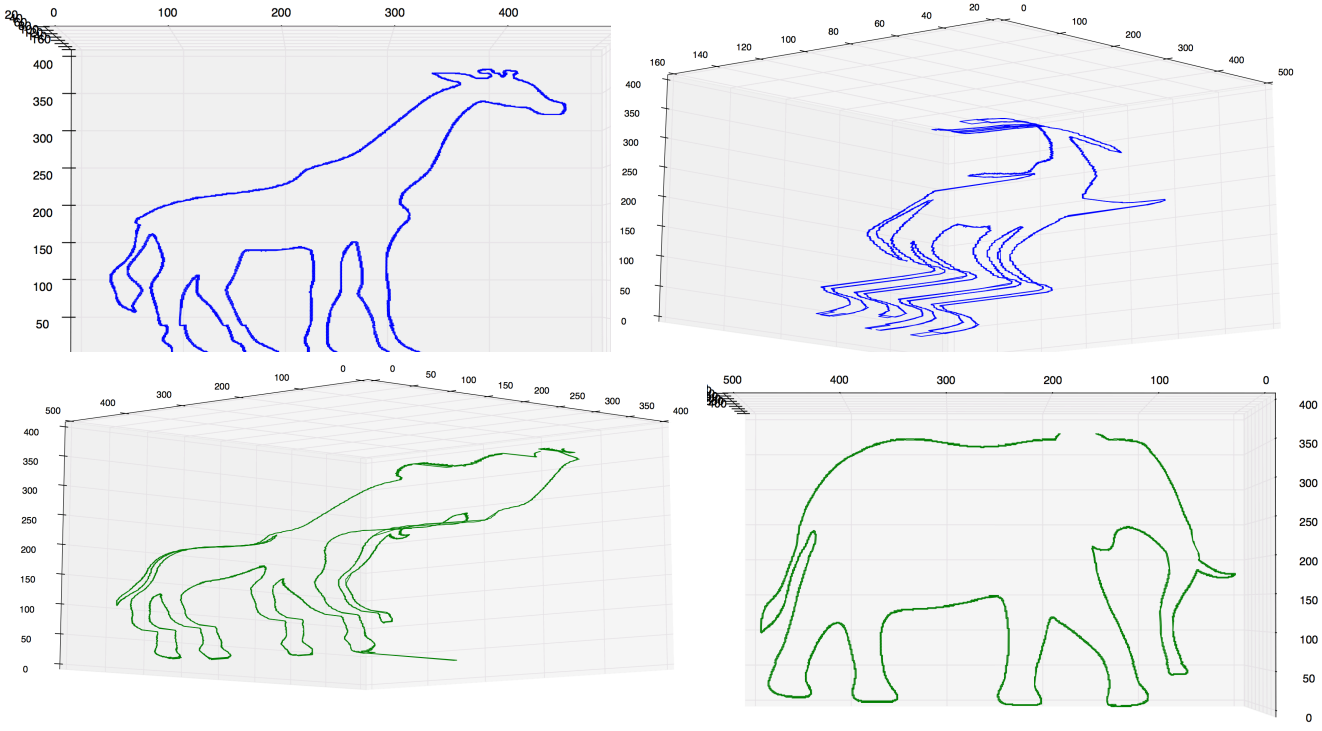
5. Curvature

As seen in the images, current model is satisfactory in showing perpendicular images in one 3D space, however the formation is not yet a sculpture. For that we need to tie all the parts together by connecting the points. The connection should be done sequentially, in the order of the image outlines.

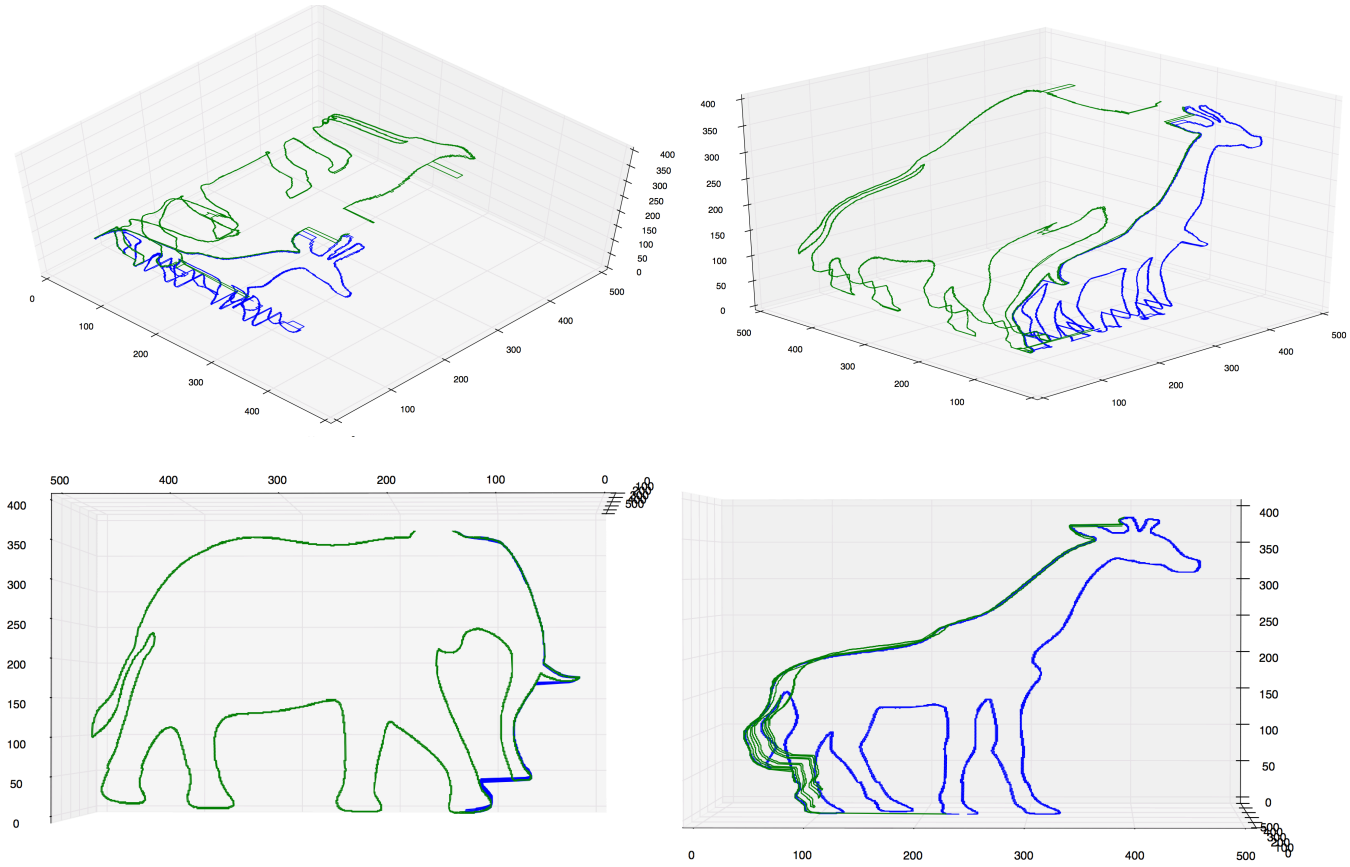
Finding this order resembles the travelling-salesman problem. We want to find the shortest path, passing exactly once in each pixel. Since the algorithm takes n! steps, number of pixels is too large for it to run in reasonable time. First attempt was to use heuristics. Starting at an arbitrary pixel, each next pixel for the path is chosen from those not yet used, having minimal distance from the previous. This showed poor results since the images are not exactly a cycle – they have protruding lines too (whiskers, eyebrows). So some points should not be further connected.

The prefect-results algorithm was found to be connection by minimal distance. Each iteration chooses a pair - one from connected and one from remaining, with smallest distance between them. While this approach runs at about $O(n^3)$ it is expensive in memory and still very slow on the large amount of pixels we have. An improvement was needed for the impatient. Again a heuristic, we find and sort the 30 closest remaining points to each connected point. This heavy part is done only once in general (again if the points all get connected). The minimal distance pairs are found in one dimension of points (each is always mapped to its closest one and the minimum of these is sought).

Now we have the traversal of pixels of the original image, the way to the curvature is taking the corresponding depth points. Connecting them gives the one-stroke curvature:



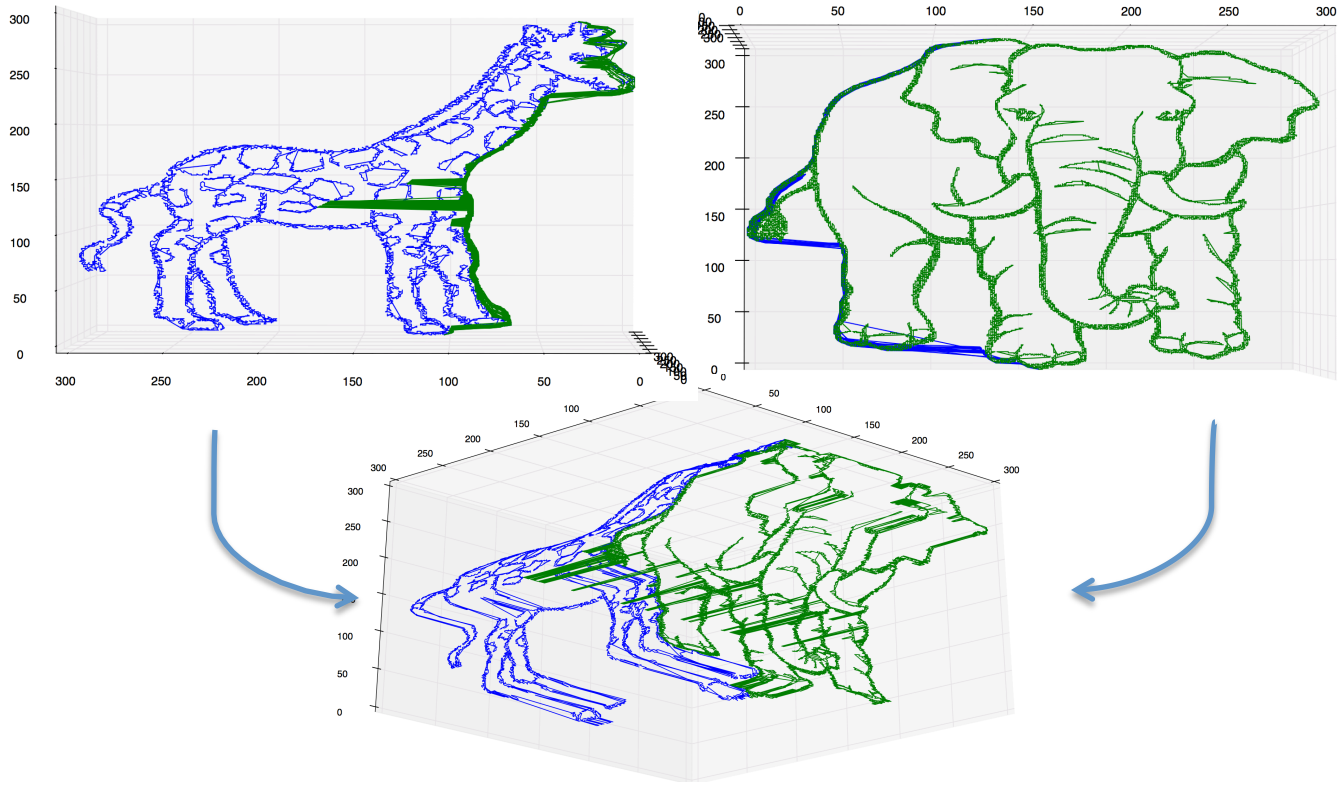
Combination of the two created curvatures gives:

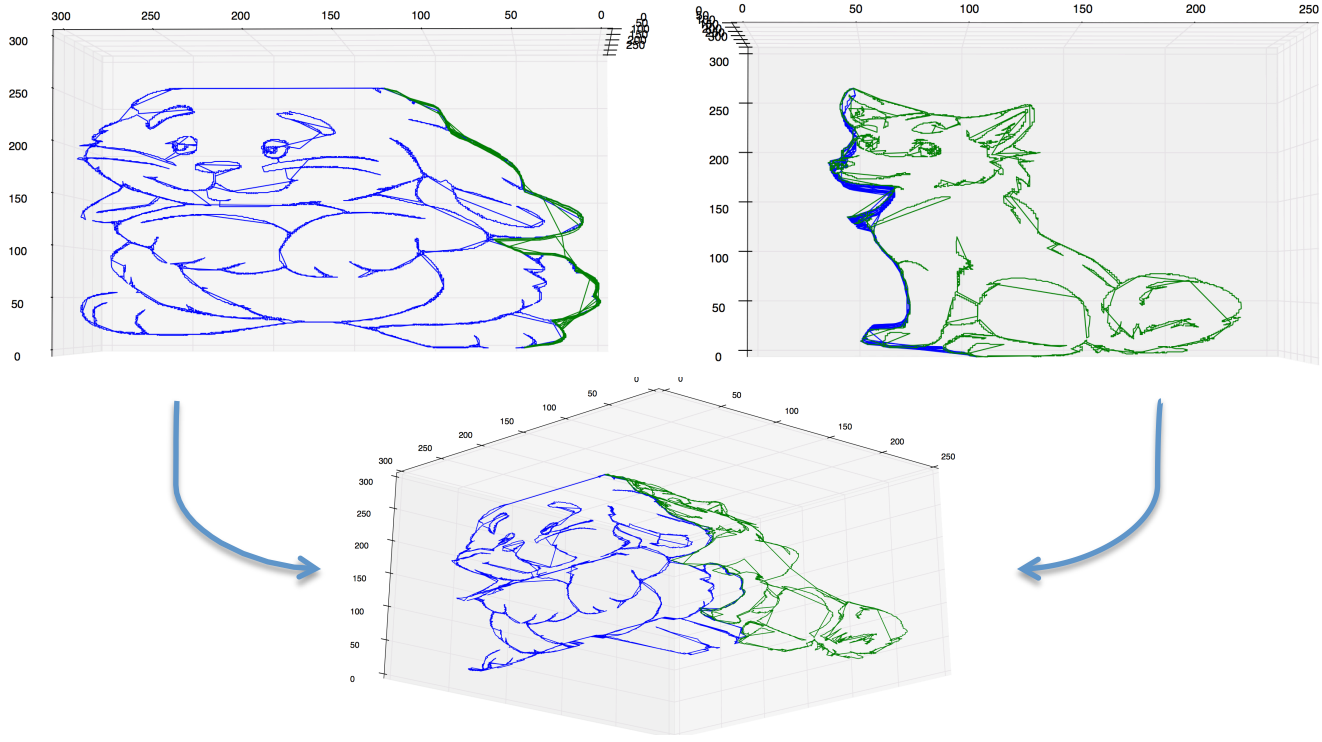


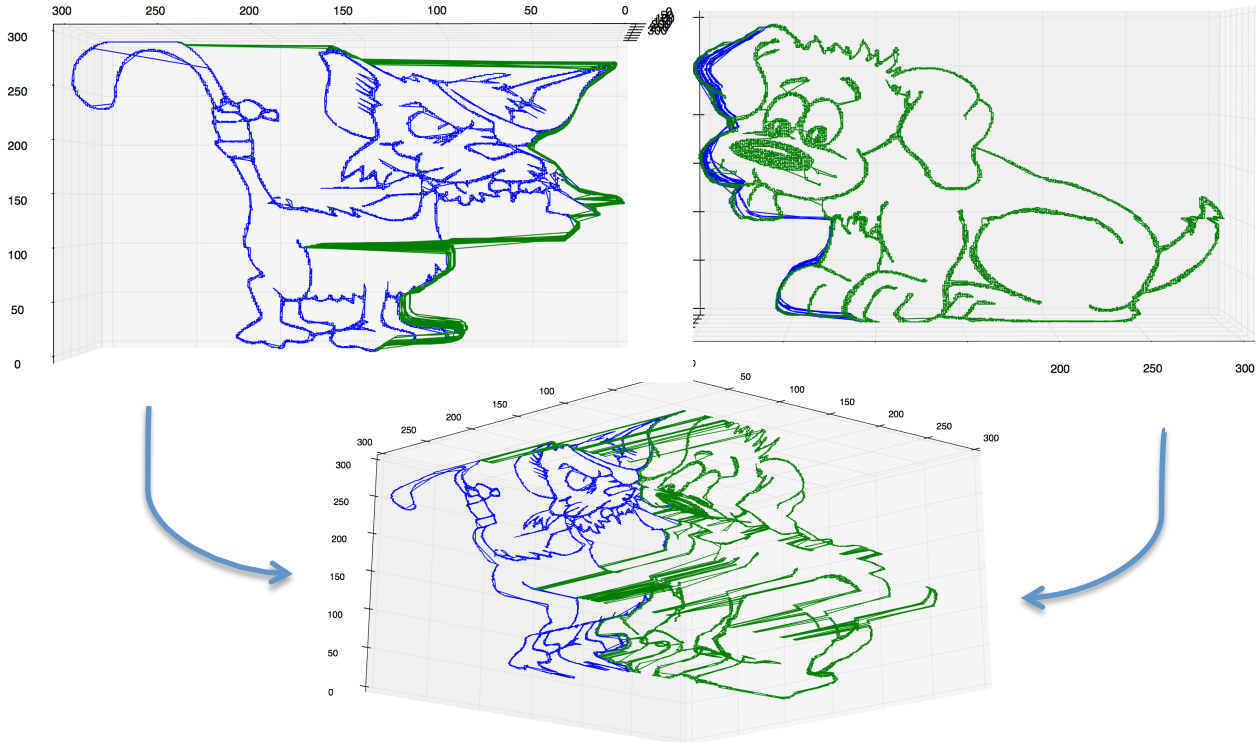
Challenge (or how did it go?)

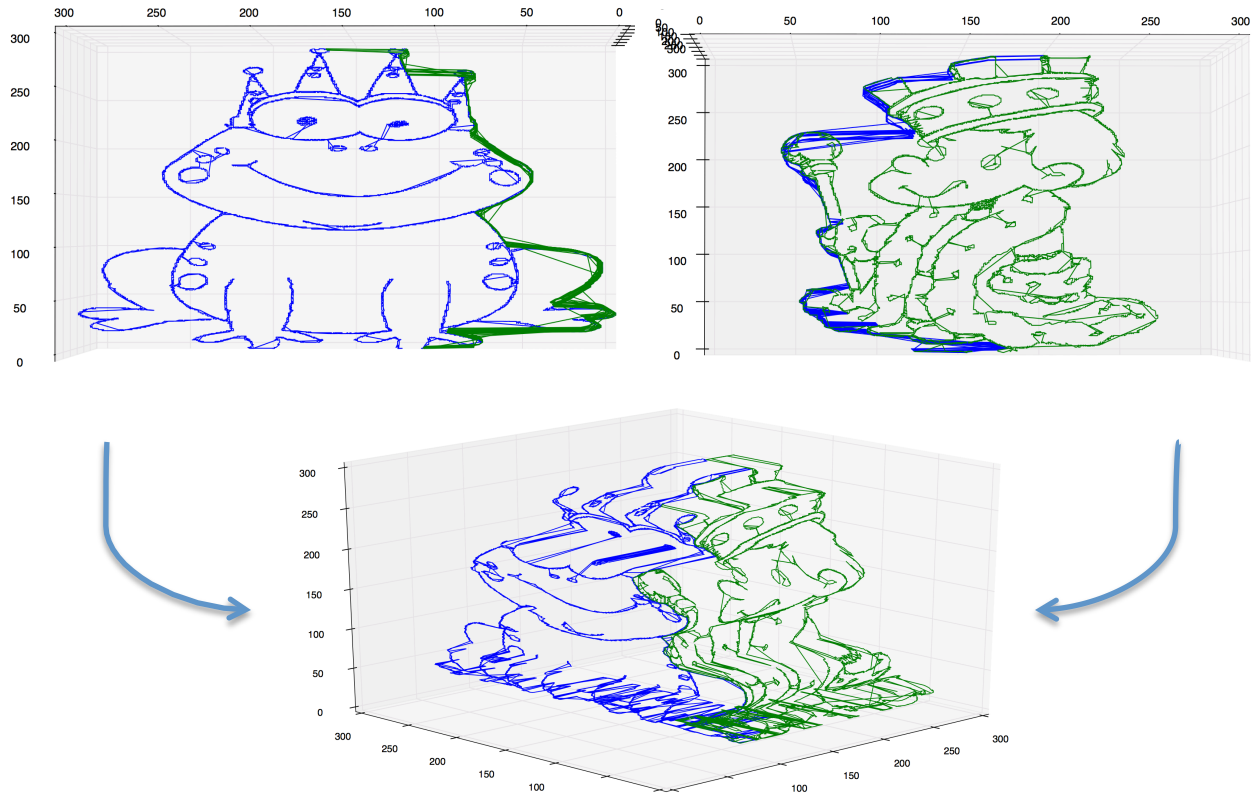
- At first, it seems like the problem is symmetrical and can be calculated via one direction. That it suffices to intersect one image, through the entire profile width of the other. This does give all pixels needed for both views but with overlapping layers that may be redundant. After many tryouts, we resolved to just project each pixel onto the profile of the other, until intersection. Doing this from both directions gives the exact necessary pixels for each view.
- The curvature was to be computed by traversing the pixels of the image in an order mimicking the path of drawing. This seemed like a problem of finding the next nearest neighbour for each pixel and continuing the curve towards it. This was not enough however, because of the specific requirement of catching all pixels while tracing the outline of the image. Considering the pixels as vertices of a graph brought the realization that this is a vertex connection problem.
- Making the projection “clasp” inner lines as oppose to the most outer line of the intersecting image. This will give a more astonishing effect when turning the sculpture around and discovering the surprise.

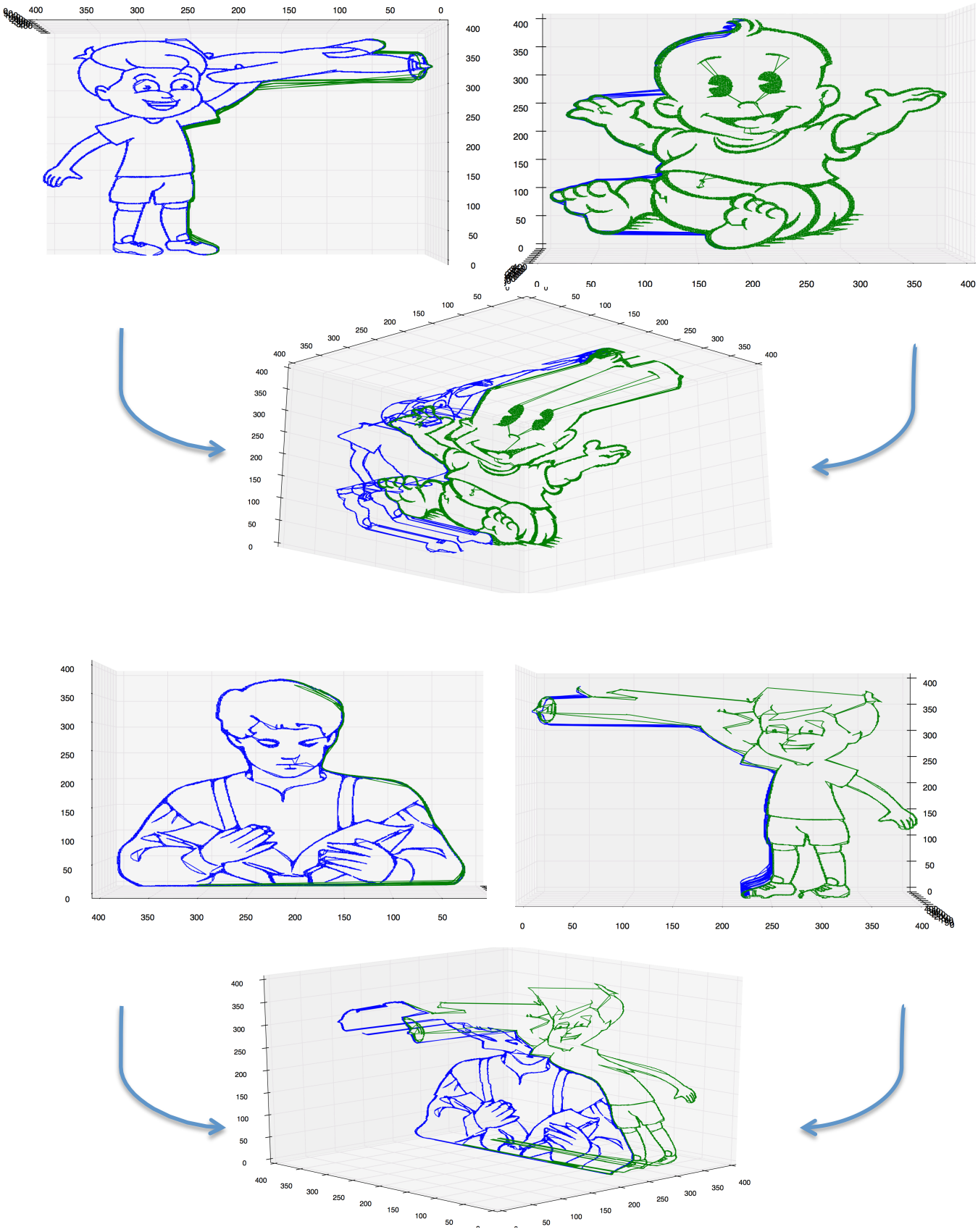
Results (or how does it look?)

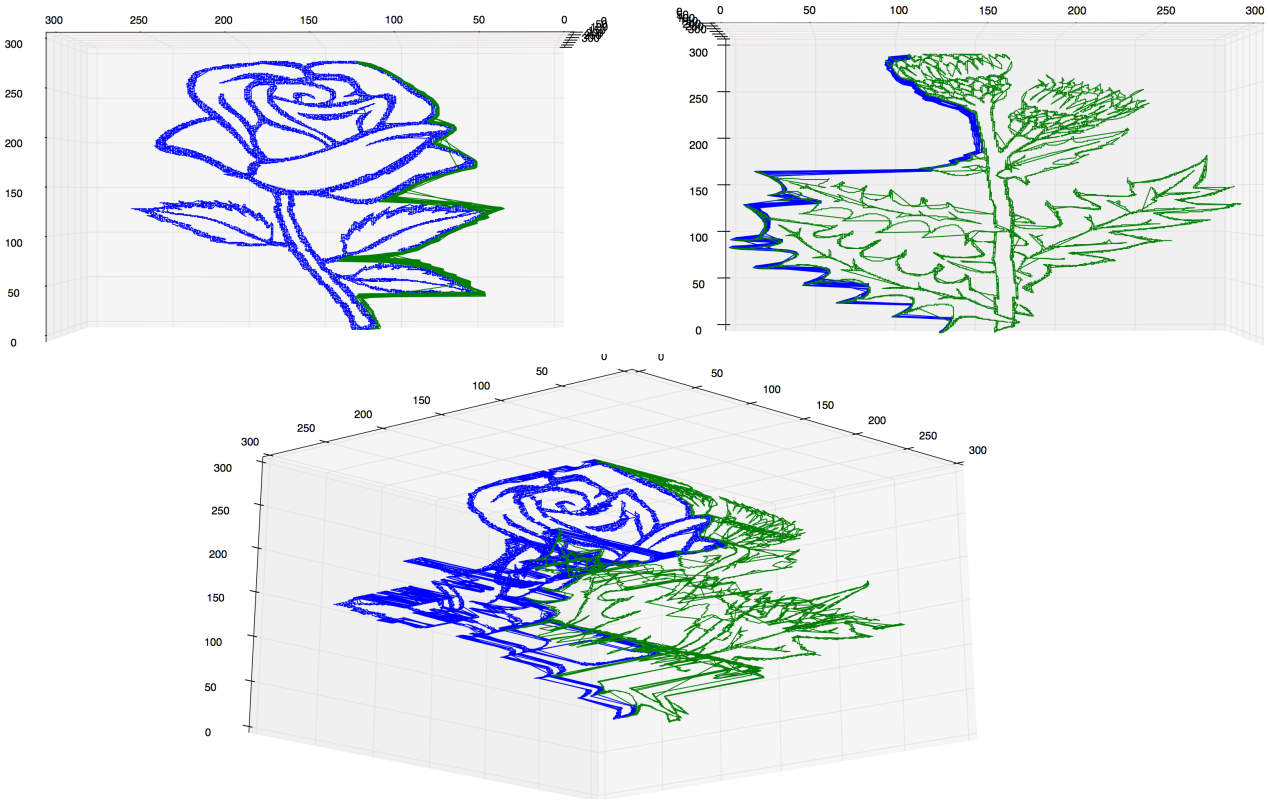





300 250 200 150 100 50 0
300 250 200 150 100 50 0
200 250 300
50 100 150 200 250 300
300 250 200 150 100 50 0


300 250 200 150 100 50 0
300 250 200 150 100 50 0
0 50 100 150 200 250 300
300 250 200 150 100 50 0
300 250 200 150 100 50 0
300 250 200 150 100 50 0





Sources

https://www.quora.com/How-are-we-able-to-see-3D-objects-when-our-retina-is-just-a-2D-screen
https://en.wikipedia.org/wiki/2.5D_(visual_perception)
http://cvcl.mit.edu/sunslides/suns-c10-l1depthcues.pdf
http://petrified.ucsd.edu/~ispg-adm/pubs/j_icga_89_1.pdf