Spiral Based Point Cloud Representations of Sculptures

Douglas Dwyer

New York City; douglas.dwyer@me.com

Abstract

Boundary representations of sculptures and other 3D objects are now widely available and useful for both 3D printing and digital rendering. I present an approach to transform such representations into a point cloud representation where the points are organized using a 3D spiral. This representation is then used to create a function that returns a point on the surface of the object using height and angle as the inputs. Such a function facilitates making drawings of 3D objects, transforming them and placing a two dimensional design onto their surface.

Introduction

In this paper, I transform files designed to make 3D prints into drawings using mathematics. The images in this paper do not make use of rendering programs, but rather convey a sense of depth by drawing dots and lines directly using the statistical programming language called R as in Figure 1.

With photographs of a sculpture from multiple angles and an algorithm one can create a 'point cloud representation.' A point cloud representation of a sculpture is a set of points in three dimensional space that are all on the surface of the sculpture. One can directly plot the points mapped to a picture plane, but whether or not the resulting plot conveys a sense of depth depends on the number of points, their location and the size of the dots used to represent the points. Transparent dots can be useful. Different lighting schemes can be deployed to convey a sense of depth with rendering software such as CloudCompare.

Figure 1: *Picasso's Dove drawn on Venus*

For 3D printing and for rendering by ray tracing algorithms, it is useful to convert a point cloud representation into a 'boundary representation' that uses a set of polygons to define the surface of the sculpture. There are many algorithms available to make such transformations in both directions.

You can plot the vertices of a boundary representation directly, but the results may not be pleasing. Connecting the vertices may not yield an attractive set of curves. Consequently, visualizing boundary representations is typically done with rendering software such as Blender.

I seek to draw points and lines directly. I want to see if I can convey a sense of depth and orientation, transform sculptures, and draw designs on them. I also

want to create images that are pleasing to look at and capable of conveying a certain emotion. In pursuit of these objectives, I have found it useful to convert boundary representations of sculptures into point cloud representations where the points are structured with the use of a 3D spiral. Such a representation allows for the creation of a function in which one inputs a height and an angle and the corresponding point on the surface of the sculpture is returned.

The approach works well for a torso or a portrait — when for each horizontal plane the points of intersection of the sculpture with the plane form a single contour. The approach struggles with something that has multiple legs (a horse) or something with a hole (a teapot or Rodin's *The Thinker*) or a surface with more than one peak (a mountain range), because there will be multiple contours in a single horizontal plane and information will be lost when representing them using a single spiral based mapping. A relief sculpture (a sculpture carved from a block of stone with limited depth) could be more efficiently represented by associating a z coordinate (depth) with each x (left to right) and y (top to bottom) coordinate rather than a spiral representation.

The remainder of the paper is organized as follows. In the next section we discuss what is possible using this approach and what are the challenges of directly drawing from the vertices in a boundary representation. The third section discusses how I create a spiral based point cloud representation and derive a function from it that returns a point on the surface of a sculpture for any height and angle. The fourth section presents some examples of the approach. Possible extensions are discussed in the conclusion.

Challenges and Possibilities

The challenges with plotting directly from boundary representations are illustrated in Figure 2. I started with an *.stl* file that was intended to support 3D printing and I extracted the 52,280 vertices. This set of vertices can be viewed as a *point cloud representation*. The upper right image in the figure plots a white dot over every point using 2 of the 3 coordinates. The upper left uses a dot that is one-fourth the size. The lower left assigns a 20% transparency to each dot and the lower right only plots 10,000 randomly chosen vertices. In general, one sees more points where the surface is more complex (e.g., in the folds of the statue's clothing). I do not find the figures effective at conveying a sense of depth in the sculpture, and it is not clear if you are looking at the front or the back of the sculpture. Directly drawing a set of vertices designed for 3D printing is challenging.

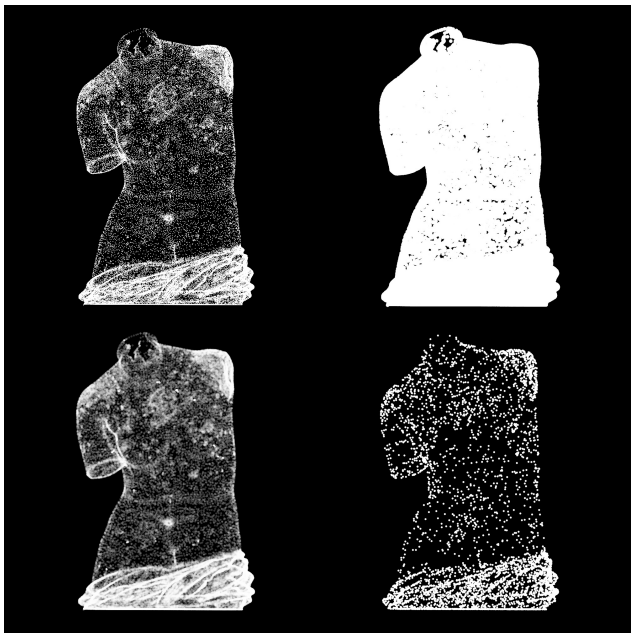

Figure 2: *Four scatter plots of a set of vertices intended for 3D printing of Venus de Milo*

Figure 1 is a demonstration of what can be accomplished with a spiral based approach. The form of Venus de Milo is shown by using 4 ribbons that spiral around her inspired by what Escher did in Rind, which is a portrait of his wife represented as a spiral winding around her. In Figure 1, the ribbons are actually made up of four sets of 20 diagonal lines that spiral around the sculpture twice. The spirals are of equal width with space between them. The location of the points on the spiral are computed by creating a set of diagonal line segments of 1,000 points each and calling the spiral function to map each point to the

surface of the sculpture. One can see the other side of the sculpture in the empty spaces between the ribbons conveying a sense of depth.

The concept of atmospheric perspective is used to further enhance the sense of depth. Specifically, each line is subdivided into a pair of points and the distance from the viewpoint to the pair of points is computed. The pairs that are furthest away are drawn first using a darker tone, and the pairs that are closest are drawn last using a lighter tone.

To draw Picasso's Dove on the statue, I first created a digital representation of this drawing by bringing it into RStudio, *locating* key points on the drawing and then fitting cubic splines through the coordinates of the key points to capture each line. With cubic splines, one can convert the relatively small number of located points into many points for each line. The design is represented as six curves that have between 100 and 500 points each.

The dove is placed on the sculpture by using the x and y coordinates as the angle and height coordinates of the spiral function. In order to achieve a pleasing placement of the drawing on the sculpture, I transformed it before calling the spiral function. I first inverted the x-axis, rotated it by 45 degrees (counter clockwise), shifted it upwards, and scaled it down. The rotation is intended to make it look like the dove is flying down the ribbons.

Developing a Spiral Representation of a Sculpture

In this section, I describe the three steps that I use to transform a boundary representation into a spiral based point cloud representation, and then how I derive a spiral based function that returns a point on the sculpture for any given height and angle.

Step 1 converts the boundary representation into a set of horizontal slices of the sculpture that form contour lines — a point cloud representation in which the points are in the correct order so that they can be joined to form a curve. The next step is to find a central axis for the sculpture, which is important when the sculpture is not *standing up straight* and does no harm otherwise. The third step takes each height and angle from a 3D spiral and computes the distance from the central axis to the surface of the sculpture on the ray corresponding to that height and angle. The fourth step is to create a function that will output the location of a point on the surface of the sculpture for any combination of height and angle.

Step 1 deploys Konturno ([3]) which is an application built by Theo Armour in 2019 (Figure 3). The app processes any *.obj* file. Of the set of triangles that make up the sculpture, it finds the ones that intersect with a given plane. When there is such an intersection, it finds the points where the edges of the triangles intersect the plane. The points are then joined together into a set of interconnected line segments in the correct order — a contour line. There can be multiple contour lines per plane. The process can be repeated for many planes and the contours are downloaded to a *.csv* file. The program makes effective use of three.js (a popular JavaScript library for building 3D apps). A sculpture with a million vertices is processed in about a minute.

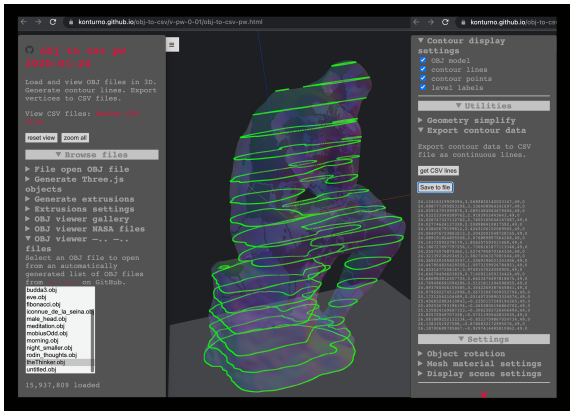

Figure 3: *Step 1: The Thinker in Konturno*

In Step 2, for each layer I find the average of each coordinate and then plot a smooth line through it. Specifically, I use three cubic splines (one for each dimension) with six knot points. Figure 4 shows the front view and the side view of this line with 10,000 randomly chosen points from the data file. The line has sufficient curvature to capture the posture of *The*

Thinker and at the same time does not exhibit regions of excessive curvature. The next step computes the radius of the sculpture with respect to the central axis.

Step 3 computes a radius for a set of chosen heights and angles. This step loops over N points on a 3D spiral that goes around the sculpture M times. For each point on the spiral, I find the layer just below the point on the spiral and the layer just above. I then rotate *The Thinker* by the negative of the angle. I find the pair of points on *The Thinker* that is furthest to the right and crosses the x-axis (where is x is the dimension of *left to right*). For each layer, I use linear interpolation to solve for the point on which the line defined by the chosen pair of points crosses the x-axis. If there is more than one such pair, I use the one with the largest x coordinate. This value is an estimate of the radius for that point on the sculpture. As I have an estimate for each layer, and I combine them with linear interpolation and store the result.

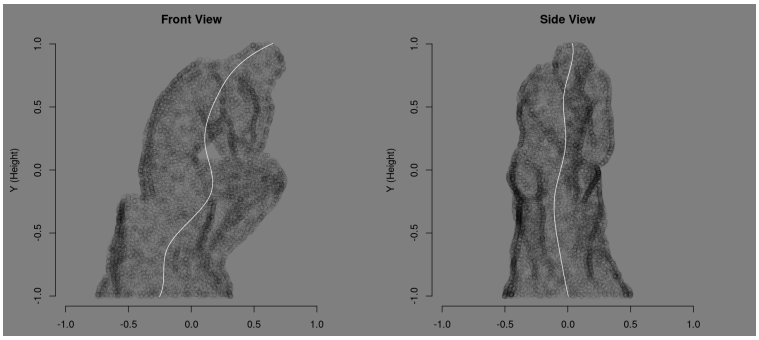

Figure 4: *Step 2: Finding the Central Axis*

In Figure 5, the right-hand panel shows a floor plan view of one layer in the middle of the sculpture and the left-hand panel shows the front view of *Rodin's The Thinker* after being rotated according to the chosen point on the spiral. The chosen layer appears as two white lines on the left. In both panels, the red point is the point that the algorithm is identifying as the point for the spiral representation. Note that for the chosen layer the point is inside the silhouette of *The Thinker* in the left-hand panel because the pair of points that cross the x-axis are not the points with the largest x in the layer as seen in the floor plan in the right panel. This process repeats N times and stores the resulting radius each time.

There are well developed algorithms for determining whether or not a ray intersects a polygon that could be used as an alternative to what is presented here and could be faster. To extract 100,0000 radii takes about 7 minutes, which is sufficient for my purposes as I only have to do this step once per sculpture.

Step 4, the last step, defines a function for which one can input a height and an angle and it returns the 3D Euclidean coordinates of the corresponding point on the surface of the sculpture. The core of this step is to find a function, $r(\theta)$, such that the set of points defined by:

$$\left\{ r(\theta),\ \theta,\ \frac{\theta}{M\pi} - 1 \right\}$$

are on the surface of the sculpture for any $\theta \in [0, 2M\pi]$. Where $r(\theta)$ is the radius that corresponds to the angle θ, and θ is the angle, and $y = \theta/(M\pi) - 1$ is the height that corresponds to the angle. The sculpture has been scaled so that its height (the y-) has a maximum of 1 and a minimum of -1. Consequently, when $y = -1$, $\theta = 0$ and when $y = 1$, $\theta = 2M\pi$, so the spiral goes around the sculpture M times.

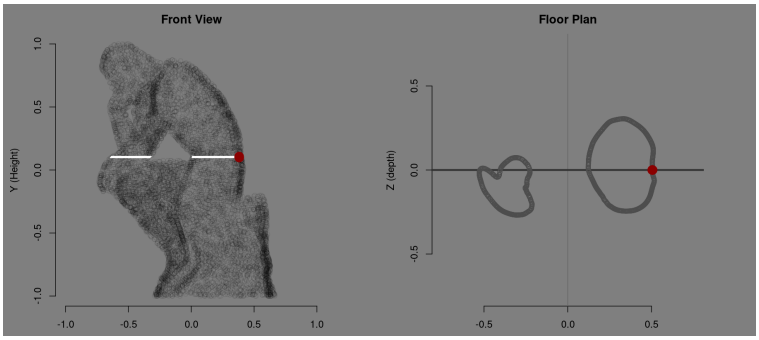

Figure 5: *Step 3: Determining the radii of the Spiral*

Figure 6 plots the radius of a sculpture as a function of θ as measured using the technique depicted in Figure 5. In this representation M is 128 implying that the horizontal runs from 0 to 256π. Represented this

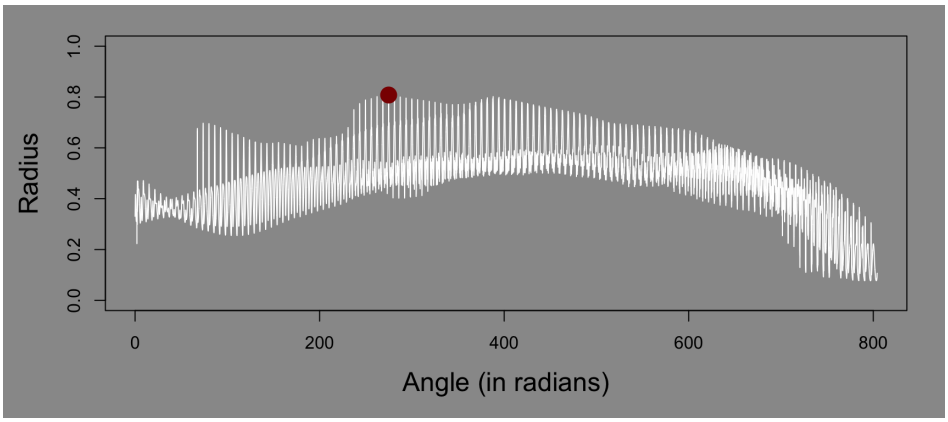

Figure 6: *Spiral Representation of a Portrait.*

way, the single curve is very jagged as the radius of the sculpture changes quickly as the spiral wraps around it. A dark red dot has been placed on the maximum radius, which turns out to be the tip of the portrait's nose for this sculpture. This portrait is based on a slice of an $.obj$ file that is derived from many simultaneous photographs of an actual person by Ten24. The file was purchased under a *Personal Use License*.

I wish to estimate the 3D coordinates of the intersection of a ray with an angle, θ_0, and a height, y_0, with the sculpture. For any given height, I can find the point on the spiral with that height by inverting the equation for height as a function of θ:

$$\theta_1 = (y_0 + 1)M\pi.$$

This point is on the spiral, but would typically not match the desired angle and consequently the radius at this point could be very different from the desired radius. My process is to *walk up* the spiral until I find a point on the spiral that matches the desired angle but corresponds to a height that is at most $2/M$ too big, and *walk down* the spiral until I find a point on the spiral that matches the desired angle but corresponds to a height that is at most $2/M$ too small. We can use linear interpolation to compute the radius of the spiral at each of these two points and then combine the two estimates to form a single estimate using linear interpolation again.

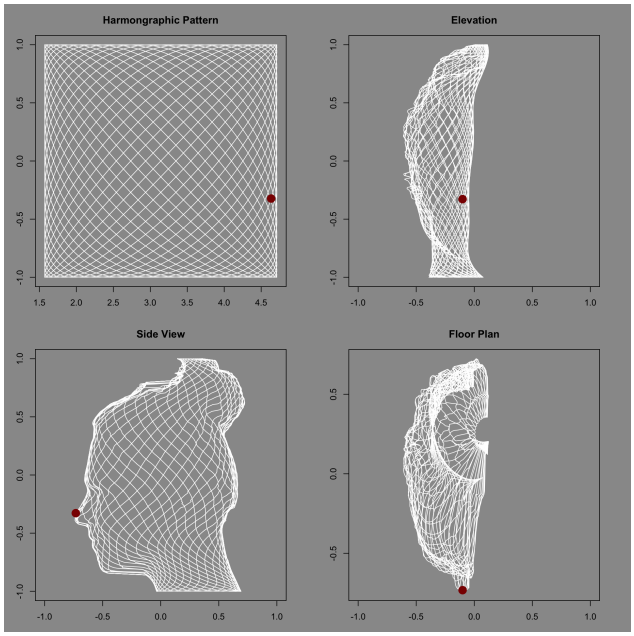

Figure 7: *Pattern mapped to the side of a Portrait*

To *walk up* the spiral, I need to know by how much, which I can accomplish by converting from radians to degrees rounding to the nearest 360 using modular arithmetic, and then converting back to radians:

$$\Delta\theta = \pi\left(\left(\frac{\theta_1 180}{\pi}\right) \bmod(360)\right)/180 - \theta_0,$$

where $\Delta\theta$ is how much θ_1 has overshot the desired angle, θ_0. If positive, the points on the spiral for which we retrieve the radii are $\{\theta_1 - \Delta\theta, \theta_1 - \Delta\theta + 2\pi\}$ and $\{\theta_1 + \Delta\theta, \theta_1 + \Delta\theta + 2\pi\}$ otherwise. Once we have the

desired radius, we convert from cylindrical coordinates to centered Euclidean coordinates and then add back the central to get the desired point on the surface of the sculpture in 3D coordinates.

Figure 7 shows how a pattern can be mapped to the side of a portrait using this function. The pattern is defined by the set of points:

$$\left\{\sin(\phi), \frac{1}{2}\,\pi\,\cos\left(\frac{23\phi}{24}\right) + \pi\right\},$$

where ϕ is any real number. This pattern is formed by the first coordinate having a wave length that is 23/24 of the second coordinate so they go in and out of sync over the course of the interval $[0, 48\pi]$ generating a *fishnet* like pattern (for more background on such patterns see [1]). The first coordinate will be mapped to the height of the sculpture and ranges from $[-1, 1]$ and the second coordinate will be used for the angle and has the range of $[\pi/2, 3\pi/2]$. So the pattern will cover the full top to bottom of the sculpture but only goes one-half of the way around the sculpture. Figure 7 presents four views. In the upper left is the pattern itself. In the upper right is the front view, the lower left is the side view and the lower right is the floor plan. The most successful is the side view as the pattern does not overlap itself. Each view has a red dot that corresponds to the red dot in Figure 6 and is the tip of the portrait's nose.

The pattern in the upper left corner is represented with 100,000 points. Mapping this pattern to the surface of the sculpture takes about 0.1 seconds in R on a laptop (a 2018 vintage MacBook Pro). The function is designed to allow the processing of 100,000 points all at once.

The images in this paper use 128 and 100,000 for M and N, respectively. Experimentation reveals that this value of N is more than sufficient for Figure 1. The differences between this figure and one based on N of 10,000 were difficult to identify, but a version with an N of 500 yielded a strange image with ripples going up and down the sculpture.

Applications

This section presents two additional examples of artwork generated using the spiral representation of sculptures derived from *.stl* files. I use Rodin's *La Pensée* (Thought) to show how the approach can capture fold and cusp catastrophes on the profile of a portrait. I use Alan Bridgwater's *Spring* as an example of Moiré patterns revealing where the surface of the sculpture is elliptic versus hyperbolic.

Rodin's Sculpture La Pensée (Figure 8) is drawn with transparent dots on 189 vertical curves that are evenly spread around 240 degrees of the sculpture. As the surface *folds* away from the picture plane the transparent dots overlap and become solid white lines. In the language of catastrophe theory, 'fold catastrophes' produce white lines in the image. We see such lines on the edges of the sculpture but also on the model's cheek and eye cavity in the interior of the sculpture. Where these lines disappear is where the folds disappear which is an abrupt change in the number of relative extreme points on the surface. Such behavior is consistent with a 'cusp catastrophe' or perhaps a 'swallowtail' or a 'butterfly' (see chapters 1–4 of [2]). In the figure, we see the overhang of the eye cavity disappear, which is indicative of a 'cusp catastrophe'.

Figure 9 presents two similar representations of *Bridgwater's Spring*. In each, the vertical curves on the sculpture are drawn by calling the spiral function using a sequence of vertical lines that go around one-half of the sculpture. In the right-hand version, a pattern of white vertical lines have been overlaid as well.

McRobie describes what he calls *Moiré Interferometry*. McRobie places evenly spaced black stripes on a transparent material and then shines direct light through them. He photographs both the black stripes and the shadows that they create on a curved surface ([2], pages 47-48). Moiré interference patterns will emerge. He demonstrates that concentric circles indicate an elliptic surface and hyperbolic curves indicate a hyperbolic surface. Figure 9 is a *virtual replication* of McRobie's photographic technique. On the sculpture's hip concentric circles emerge indicative of an elliptic surface, and on the sculpture's side a hyperbola emerges

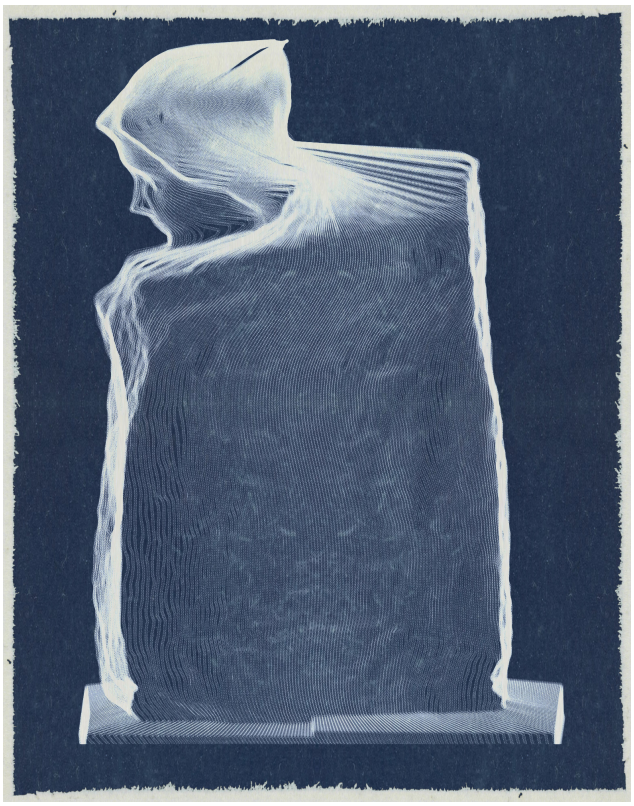

Figure 8: *Rodin's Sculpture La Pensée (Thought) with fold and cusp catastrophies*

indicative of a hyperbolic surface.[1]

Conclusion

In this paper, I have presented a mathematical technique to represent sculptures and shown applications. A gallery supplement provides additional examples, with links to animations on the instagram account @dashdotdotdashdotdot. Future work could experiment with using multiple spirals to capture sculptures that are not easily represented with a signal spiral. One can also experiment with methods of representing depth in an image. The picture plane need not be a plane and depth could be conveyed using lines that become thinner and less distinct as they recede.

Potters often use a glaze to place a design onto handmade ceramic pots. Before committing to the design, potters will consider how it will look from multiple perspectives. One could photograph a pot from multiple perspectives, create a point cloud representation, convert it to a boundary representation, create a spiral representation and then experiment with how different designs will look from multiple perspectives and test alternative placements. One may want to convert back to a boundary representation to get a photographic rendering of what the design would look like.

This approach could be extended to create boundary representations that could then be used for making 3D prints. For example, Eunwha Kim-Kilian made a bronze sculpture version of Escher's Rind in 1998.

[1]When Figure 9 is viewed on a printed page, a Moiré interference pattern only appears on the right-hand image. When viewed on a screen, however, Moiré interference patterns may emerge on the left-hand image as well due to how the nearly vertical curved lines are being represented on the screen's pixels. When viewing Figure 9 on a screen, if one zooms in the Moiré interference patterns should disappear on the left-hand image, but not on the right-hand image.

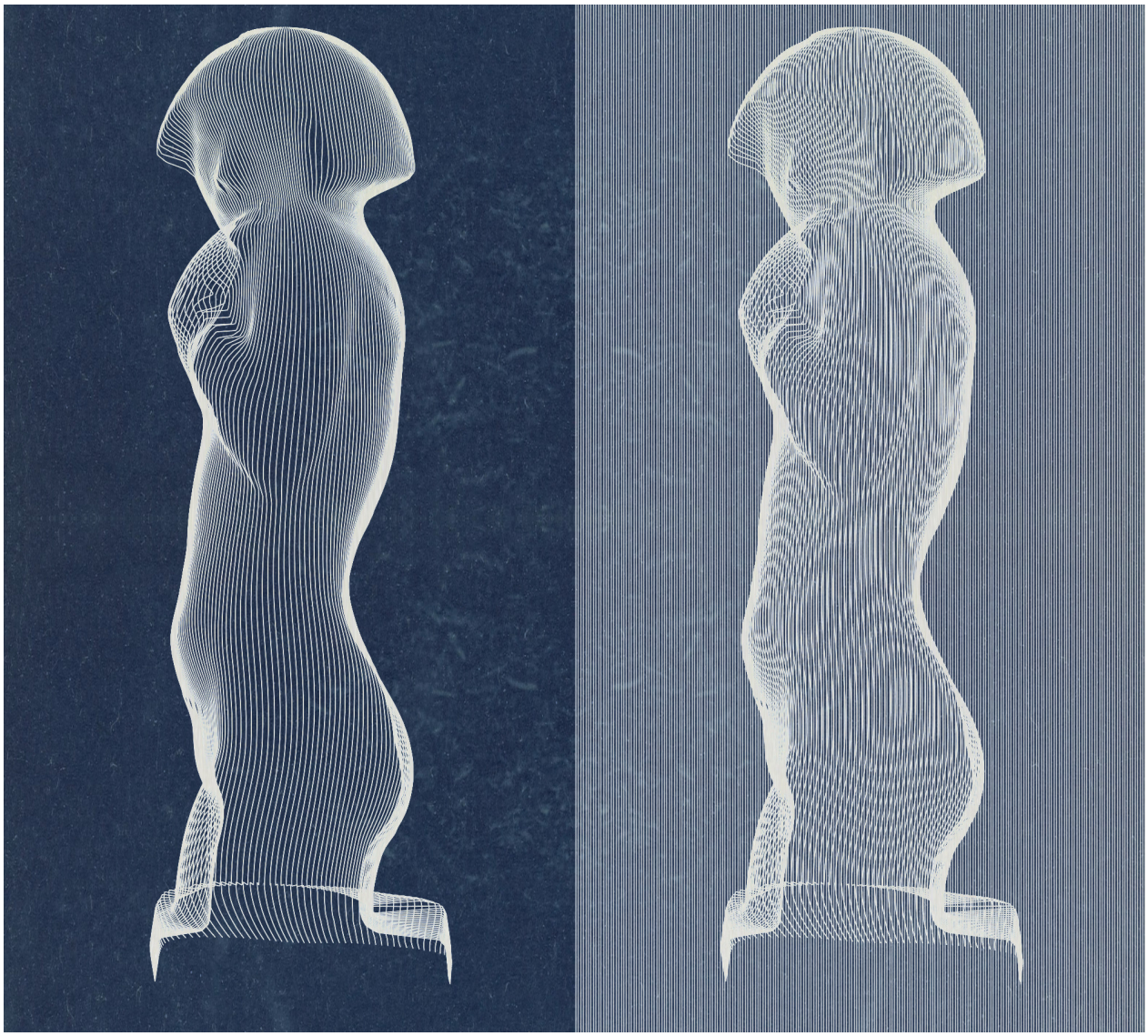

Figure 9: *Alan Bridgwater's Spring with and without Moiré Interference Patterns*

One could make a boundary representation of a spiral version of a sculpture of choice and make a 3D print. Another application would be to make 3D prints that look like silhouettes of different sculptures from different viewpoints. When holding the print in your hand you would see one silhouette transform into another as you rotated it.

I would like to acknowledge the helpful comments of two anonymous referees and Theo Armour for his help with three.js and his encouragement. I would also like to express my appreciation of myminifactory.com, *Scan the World* and *@3DLirious* for making many wonderful sculptures available in *.stl* format.

References

[1] A. Ashton. *Harmonograph: A Visual Guide to the Mathematics of Music*. Wooden Books, 2003.

[2] A. McRobie. *The Seduction of Curves*, Princeton University Press, 2017.

[3] T. Armour. *Konturno.* https://konturno.github.io