Dithering by Wires of Orthogonal Images

Firas Habib[1], Sagit Asman[1], Gershon Elber[1]

[1]Computer Science Department, Technion, Israel

Abstract

In this article, we present a method for dithering by wires (DBW) a pair of images by carefully positioning a selected set of 3D line segments or linear wires in 3-space, in the interior of an axis-aligned cube, while the images are on two orthogonal faces of the cube. Wires can either be of arbitrary length or start and end on the faces of the cube, possibly between predefined anchor points. The projections of the aforementioned wires on the XZ and the YZ planes reconstruct dithered versions of the two original images.

The presented approach can operate either with black wires, over a white background, for grey scale input images, or with colored wires, toward the reconstruction of colored images. The output is a set of 3D line segments that represent (linear) wires in space. Any assembly approach for wires can be employed to reconstruct a real tangible representation of the results, while, in this work, we only demonstrate computer-based results, ready for such assemblies.

Keywords: Line-art, Random algorithm, Color and B&W dithering

1 Introduction

The idea of dithering by wires (DBW) is not new. Quite a few artists have employed these ideas over the years. Figure 1 shows one example, created using the scheme presented in this work. In (a), the input image is shown. In (b), the DBW is shown, where the extent of the wires varies. In other words, the wires float in the plane. To ease possible tangible assembly, we also offer the option of having wires that stretch throughout the square domain, as can be seen in (c). More importantly, we seek to place the wires in 3-space, either floating inside an axis-aligned cube or spanning from one face of the cube to another (as in Figure 1 (c)), in such a way that the wires project into two different directions and yield two different DBWs of two independent images.

The common solution to DBW, solves a 2D problem of one given image. As stated, in this work, we seek to simultaneously dither two images by 3D lines. Consider two square independent images, $\mathcal{I}_i,\ i = 1, 2$ of the same size, and assume the two input images are on the XZ and YZ planes of a cube. We like to find a set of 3D lines of finite thickness (i.e., linear wires, or simply *wires* hence after) that when projected on the XZ and YZ planes, will dither-approximate the two input images. For now, and unless otherwise stated, we assume black wires over a white background.

A naive approach to solve this problem can be combinatorial. Select n^2 anchor points in a 2D grid over each of the six faces of the cube. Then, exhaustively search for a possible optimal wire between the $O(n^4)$ pairs of possible locations on two different faces, in each step of the algorithm. Considering that typically, $O(n)$ operations are required to evaluate the contribution (or score) of a wire, this approach is going to be prohibitively expensive, for any reasonable n.

Hence, we consider a stochastic algorithm that randomly generates spatial wires that are either between points on different faces of the cube, or of arbitrary lengths inside it. The algorithm controls how many wires are created and allows some fairness control between the two input images (possibly giving some priority to one of the images).

The rest of this work is organized as follows. In Section 2, some related previous work is discussed. In Section 3, we present the stochastic algorithm, whereas in Section 4, results are presented. Finally, we conclude in Section 5.

(a) (b) (c)

Figure 1: *An example of dithering by wires (DBW) the picture of the Mona Lisa, in (a). 4500 lines are employed in this dithered (converted to grey level input) image, in (b) and (c). As a 2D problem, (b) shows a 2D result, for arbitrary length wires. However, the wires can also be in 3-space, stretched between two faces of an axis-aligned cube, as is shown in (c) in a general view, only to be projected onto 2D planes (faces of the cube) in multiple directions, resulting in (independently) dithered images by wires. This is where this work begins...*

.

2 Previous Work

While academically little can be found in dithering and rendering by wires or string art, in the arts there are some relevant results, all planar. Petros Vrellis [14] creates a way to knit and dither portraits using threads and the placement of the threads is based on 2D computer simulation. Linify [2] shows another 2D example of line art computer simulation for dithering planar images. In [11], Roberto Reif also shows an attempt at a computer-based simulation of line art.

Knitter [12] by Christian Siegel is a software package for creating knitting, with circular/square frames, from an image and is publicly available. Several approaches are presented to solving the 2D problem, adjusting the locations of the pins (randomized, equally spaced and other patterns of spatially spreading the pins).

KnitReport [5] by Malo Malcom Drougard studied previous results like [12] and implemented 2D line art simulation, that is made public. This report also displays quantitative comparisons between the different variants.

This work aims to employ wires in space so they project into independent dithered images, in different directions. In this sense, it is related to similar previous work, such as [6, 10], in which the synthesized 3D geometry projects into independent 2D shapes, in different directions. For examples, Voxel based synthesized geometries, deformed spherical volumetric cells, and deformations of silhouette areas from different directions, were exploited.

In [3], a multi-axis robotic arm is employed toward the automation of the assembly process of planar string arts. A similar multi-axis robotic approach can possibly be used to fabricate 3-space cuboid string arts, as we proposed in this work.

In summary, solutions to the planar problem are pretty much known and some previous results have used a similar stochastic approach as in here. However, in this work, we seek to take the problem into a higher dimension, placing lines in 3-space and have their projection onto orthogonal planes yield different, independent, dithered images.

3 Algorithm

Before we present the stochastic algorithm in Section 3.1, we need to introduce some preliminaries. Consider an intermediate step of the algorithm, having the current state of the dithering process and a new wire to select next. The wire that will be selected is the one that maximizes some optimization function, for example, covering most uncovered pixels so far. The algorithm is greedy in the sense that it chooses the best wire at each step and updates the stored data to that effect. As a result and given a budget of n wires, the algorithm ensures no global optimum in any sense. Not only that different locations of n wires might yield a superior result, but clearly, it can very well be that a less optimal local choice of a wire at some step will yield a more optimal answer, globally, at the end.

During the execution of the algorithm, we hold two auxiliary versions of each of the input images:

1. **StateImage** that is initialized with the original input image. In each iteration, we subtract the new selected wire from **StateImage**, so that in the following iteration, that same wire will score little.
2. **DitheredImage** - Initialized as a background image, and in each iteration, the selected wire with the maximized score is added to this image.

The score of a wire (and possibly the selected color of the wire) is calculated by iterating over all the pixels that are covered in the image by the wire. A variation of the Bresenham algorithm [8] is employed toward this end. We further assume that no selected wire will be parallel to the cubes' faces. I.e., given $wire(P_0, P_1)$, with end points $P_0(p_0^x, p_0^y)$ and $P_1(p_1^x, p_1^y)$, we enforce $p_0^x \neq p_1^x, p_0^y \neq p_1^y$ and $p_0^z \neq p_1^z$. This restriction inhibits the possibilities that the algorithm will independently dither the two images with wires that project to points in the other image.

Given the two input images, where one image is dominantly darker (considering the average intensity of the pixels in the given image), the scores of the wires will be dominated by the darker image, leading to wires that neglect the lighter image. To alleviate this difficulty, we introduce a user-controlled multiplicative fairness score for each of the images, effectively controlling the average intensity.

The number of wires we use should be limited. Too few will clearly fail to approximate and dither the input images to a satisfactory level. Too many wires and we will end up with data sets that will be difficult to fabricate. Hence, striving to minimize the number of wires while achieving dithering results that are appealing, we introduce a heuristic in our algorithm to elevate the importance of critical zones in the input images. We assign pixels that lay on important features in the input images a higher score compared to pixels that lay inside a monotone zone. In essence, we will force the algorithm to focus on edges, in a similar way artists typically work and sketch. While the non-photo-realistic rendering (NPR) community has invested a significant effort into the computation of saliency of visual features [9], herein the salient edges are derived by using the simple Sobel edge detection operator [13]. See, for example, Figure 2.

3.1 The Stochastic Algorithm

Consider two square independent images, $\mathcal{I}_i,\ i = 1, 2$ of the same size and relative fairness $\mathcal{F}_i$, to be dithered with N_{wires} 3D wires. Algorithm 1 presents the top level steps. Algorithm 1 receives $\mathcal{I}_i,\ \mathcal{F}_i,\ i = 1, 2$ and N_{wires}, but also the number of random trials, *NumRndm*, that will be conducted for each selection of a new wire. Algorithm 1 iterates N_{wires} times and fetches the best wire it can find at each step, in Line 4, by calling **FindNextWire**, in Algorithm 2. Then, the best wire found is subtracted from the two **StateImage** buffers, in Lines 7 and 8 of Algorithm 1, and is added to the respective **DitheredImage** buffers, in Lines 9 and 10.

Algorithm 2 randomly generates a new wire, in Line 4: for wires that end on the faces of a cube, **GenRandomWire** generates random wires between two random points on two different random faces of the cube, while ensuring the created wire is co-planar with no face of the cube. For wires interior to the cube, **GenRandomWire** simply generates wires between two random locations inside the cube. For each such new

Figure 2: *An image of Michelle Obama is presented on the left. Automatic features extraction using a Sobel edge detection filter is presented on the right.*

Algorithm 1 StochasticDither - a randomized wire selection algorithm.

Input:

$\mathcal{I}_0$, $\mathcal{I}_1$: Two square images of similar size. Holding both the **StateImage** and **DitheredImage**;

N_{wires}: Number of wires to dither $\mathcal{I}_0$ and $\mathcal{I}_1$ with;

$\mathcal{F}_0$, $\mathcal{F}_1$: Fairness weights for $\mathcal{I}_0$ and $\mathcal{I}_1$;

$NumRndm$: Number of random wire trials, per new wire;

Output:

$\mathcal{W}$: A list of N_{wires} wires in 3-space dithering $\mathcal{I}_0$ and $\mathcal{I}_1$;

Algorithm

$N := 0$;
$\mathcal{W} := \emptyset$;
while $N \neq N_{wires}$ **do**
    $\{P_0, P_1\}$:= **FindNextWire**($\mathcal{I}_0$, $\mathcal{I}_1$, $\mathcal{F}_0$, $\mathcal{F}_1$, $NumRndm$); ▷ Algorithm 2
    P_0^{xz}, P_1^{xz} := **ProjectXZ**(P_0, P_1); ▷ Project points P_0, P_1 onto the XZ plane
    P_0^{yz}, P_1^{yz} := **ProjectYZ**(P_0, P_1); ▷ Project points P_0, P_1 onto the YZ plane
    SubtractWire($\mathcal{I}_0$, P_0^{xz}, P_1^{xz}); ▷ Subtr. the pixels in line (P_0^{xz}, P_1^{xz}) from $\mathcal{I}_0$.**StateImage**
    SubtractWire($\mathcal{I}_1$, P_0^{yz}, P_1^{yz}); ▷ Subtr. the pixels in line (P_0^{yz}, P_1^{yz}) from $\mathcal{I}_1$.**StateImage**
    AddWire($\mathcal{I}_0$, P_0, P_1); ▷ Add the pixels in line (P_0^{xz}, P_1^{xz}) to $\mathcal{I}_0$.**DitherImage**
    AddWire($\mathcal{I}_1$, P_0, P_1); ▷ Add the pixels in line (P_0^{yz}, P_1^{yz}) to $\mathcal{I}_1$.**DitherImage**
    $\mathcal{W} := \mathcal{W} \cup \{P_0, P_1\}$; ▷ Add wire to returned set
    $N := N + 1$;
Return $\mathcal{W}$;

random wire, the algorithm computes its score with respect to the two input images, in Lines 7 and 8, by calling **CalculateWireScore**, in Algorithm 3. The weighted score is then compared to the best score so far and kept if better.

Algorithm 2 FindNextWire - find the next best wire to use in the DBW.

Input:
$\mathcal{I}_0$, $\mathcal{I}_1$: Two square images of similar size;
$\mathcal{F}_0$, $\mathcal{F}_1$: Fairness weights for $\mathcal{I}_0$ and $\mathcal{I}_1$;
$NumRndm$: Number of random wire trials, per new wire;
Output:
W: The best wire found in the random search for the next wire;
Algorithm

```
1:  BestScore := 0;
2:  BestPins := ∅;
3:  for i := 0; i < NumRndm; i++ do
4:      P0, P1 := GenRandomWire();
5:      Score0 := CalculateWireScore(I0, "XZ", P0, P1);          ▷ Algorithm 3
6:      Score1 := CalculateWireScore(I1, "YZ", P0, P1);          ▷ Algorithm 3
7:      ScoreSum := Score0 * F0 + Score1 * F1;
8:      if ScoreSum > BestScore then
9:          BestScore := ScoreSum;
10:         BestPins := {P0, P1};
11: return BestPins;
12:
```

Finally, **CalculateWireScore**, in Algorithm 3, computes the score of a give wire, in the current dithering state. For every pixel in that wire, it computes the difference, in Line 5, between the desired grey level of that pixel and the black color of the wire. Line 5 also takes into consideration the weight of that pixel, following the saliency of the pixel. The function **Diff** in Algorithm 3, Line 5, is the function the defines the optimum criteria. For grey level images, we compute the level in which a candidate wire darkens a specific pixel with respect to the desired level of the pixel and its level in the current state image. A wire will be chosen, if its average contribution per pixel is the highest. This, so longer wires will not necessarily be favored over shorter ones, due to their length.

4 Results

All the examples presented in this work were created using an implementation of the presented algorithms in the IRIT geometric modeling kernel and are now available as part of [7]. All presented examples employed input images of around 300×300 pixels. The number of random trial, in the stochastic algorithm, per new wire ($NumRndm$ in Algorithm 2) was typically 1500, unless stated otherwise. Computation was conducted on an i7 3.4GHz machine with 32G of RAM, and a single thread.

As stated at the beginning, the combinatorial exhaustive approach is feasible but extremely expensive. We provide one example that employs the exhaustive combinatorial search, in Figure 3. Note the equally spaced pins, on the left and right sides of Figure 3 (b) and (d), where the wires end. The computation time of this specific result for 35^2 pins on each face was 52 hours! Figure 4 shows a similar example using the stochastic algorithm that was executed in about 10 seconds. Interestingly and while subjective, the quality of the stochastic result is not visually inferior to the combinatorial exhaustive search.

Algorithm 3 CalculateWireScore - compute the score of the given wire with respect to image $\mathcal{I}$.

Input:

$\mathcal{I}$:= Input image;

$\mathcal{P}$:= Plane to project the wire on;

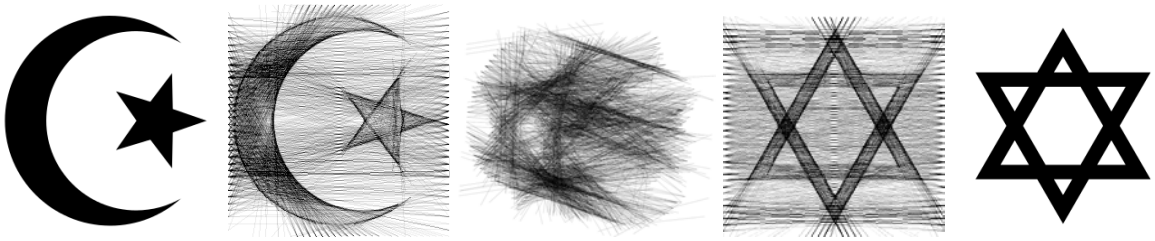
P_0, P_1: End points of a wire to consider its score;

Output:

$Score$: The score of this specific wire in $\mathcal{I}$;

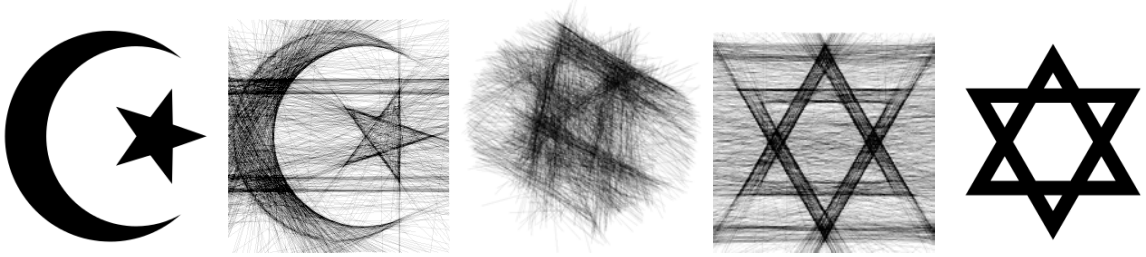
Algorithm

```
1: Score := 0;
2: Len := 0;
3: P_0^P, P_1^P := Project(P_0, P_1, P);
4: for every Pixel ∈ Line(P_0^P, P_1^P, I) do          ▷ E.g. a Bresenham-like algorithm
5:     Score := Score + Diff(I[Pixel].StateImage.Color, BlackColor) * I[Pixel].Weight;
6:     Len := Len + 1;
7: return Score/Len;
8:
```



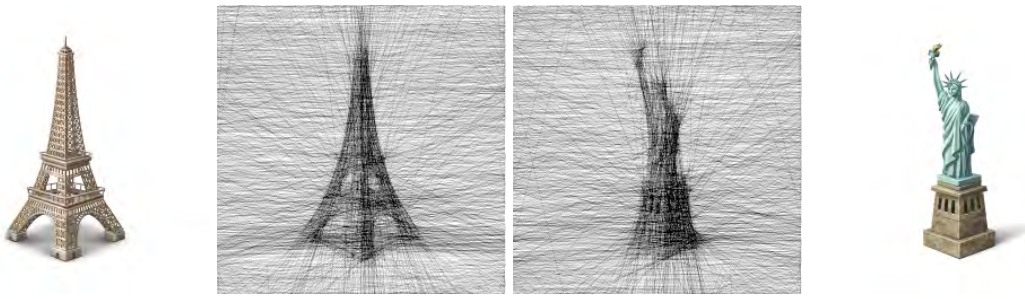
(a) (b) (c) (d) (e)

Figure 3: *The two icons, in (a) of the Islamic crescent moon and the Jewish David Star in (e), were simultaneously DBW with 3D wires that terminate on the faces of a cube and using a combinatorial approach, as can be seen in (b) and (d), while (c) shows a general view of the 3D wires. 2000 wires, 1 : 1 Fairness, 35^2 pins on each face of the cube. Over 52 hours of computation time! Icons (a) and (e) are from https://commons.wikimedia.org/wiki/File:Symmetric_religious_symbols.svg. Compare with Figure 4.*



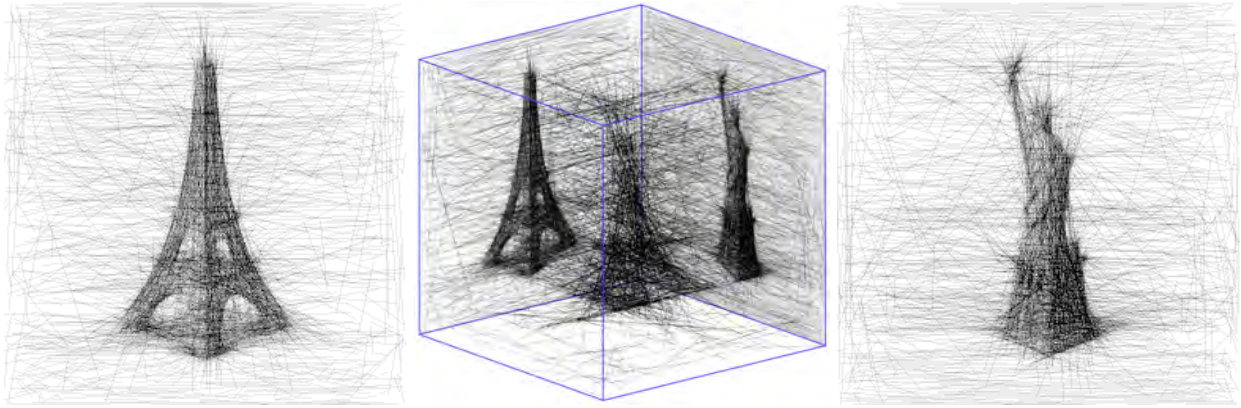
(a) (b) (c) (d) (e)

Figure 4: *The two icons, in (a) of the Islamic crescent moon and the Jewish David Star in (e), were simultaneously DBW with 3D wires that terminate on the faces of a cube and using a stochastic approach, as can be seen in (b) and (d), while (c) shows a general view of the 3D wires. 2000 wires, 1 : 1 Fairness. About 10 seconds of computation time. Icons (a) and (e) are from https://commons.wikimedia.org/wiki/File:Symmetric_religious_symbols.svg. Compare with Figure 3.*



(a) (b) (c) (d)

Figure 5: *The two icons, in (a) of the Eiffel tower and in (d) of the statue of liberty, were simultaneously DBW with wires that terminate on the faces of a cube and using a stochastic approach. 2000 wires, 1 : 1 Fairness. Input icons are from https://iconarchive.com (http://www.iconka.com).*



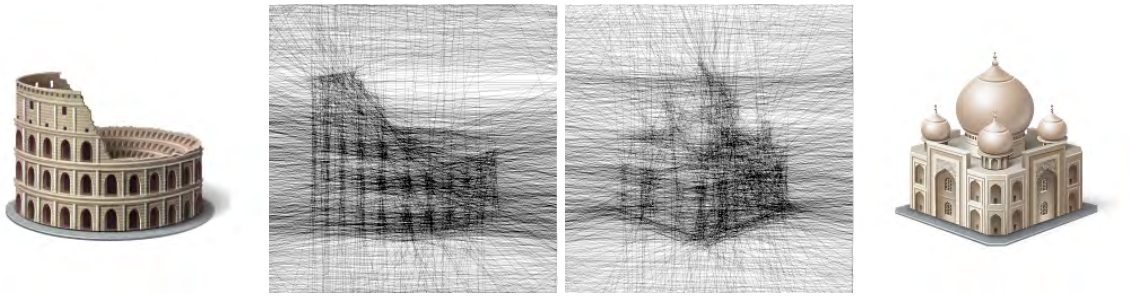
(a) (b) (c)

Figure 6: *The two icons From Figure 5 were simultaneously DBW with wires that float inside a cube, in (a) and (c), using a stochastic approach. (b) shows the same set of wires of (a) and (c) but in a general 3D view along with the two dithered projections, on two different faces of the cube. 2000 wires, 1 : 1 Fairness.*

The rest of the results presented in this section were created using the stochastic approach. Figures 5 to 8 show examples of iconic monuments that are simultaneously DBW. Figure 6 shows a similar result to Figure 5 using wires that float inside the cube, and also shows the wires in 3-space and how they project into two dithered images, in two different faces of the cube. Figure 7 shows a second DBW example for a set of two icons, while Figure 8 employs the same icons and setup as in Figure 7, to exemplify the effect of modifying the number of random trials per wire, that as stated, is 1500 trials by default. Clearly, the more trials the better the expected result will be, but with a direct cost over the computation times. Figure 9 depicts this more systematically, showing (as expected) that the more trials we perform, the better the score of the selected wire, throughout the selection process, of 2000 wires. More interesting are the slopes of these plots in Figure 9. A steep plot denotes the quick loss of contribution, for the n'th wire, whereas a shallow plot means the better preservation of the contribution of the n'th wire.

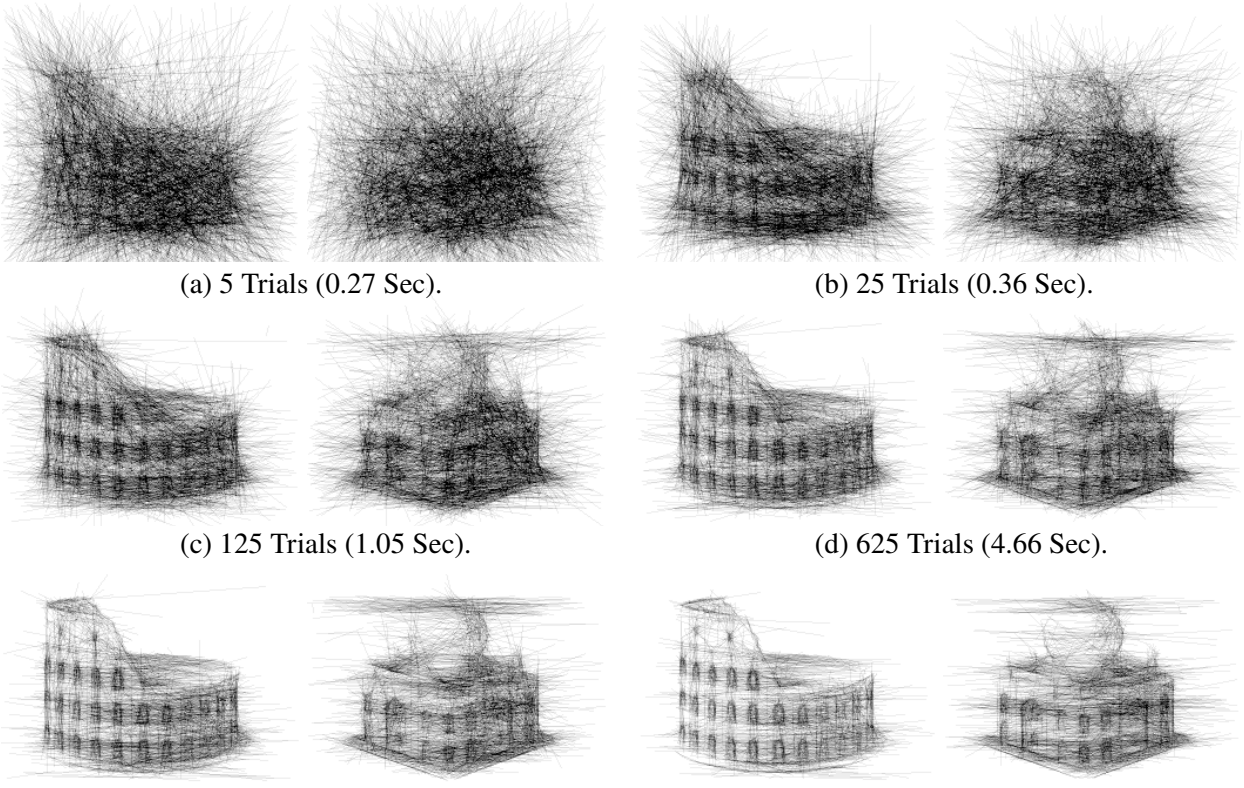
Figure 10 shows a simultaneous DBW of the pictures of Albert Einstein and Stephen Hawking, using wires that span between faces of a cube. 5500 wires were employed. Clearly, the number of wires has an immediate affect over the result and in Figure 11, we show the same result as in Figure 10, while varying the number of dithered wires.

Figure 12 shows the Obamas simultaneously DBW, using wires that span between faces of a cube. 4000 wires are employed with fairness that is very weakly biased toward Barack. In Figure 13, we experiment



(a) (b) (c) (d)

Figure 7: *The two icons, in (a) of the Colosseum and in (d) of the Taj Mahal, were simultaneously DBW with wires that terminate on the faces of a cube and using a stochastic approach. 2000 wires, 1 : 1 Fairness. Input icons are from https://iconarchive.com (http://www.iconka.com).*



(a) 5 Trials (0.27 Sec). (b) 25 Trials (0.36 Sec).

(c) 125 Trials (1.05 Sec). (d) 625 Trials (4.66 Sec).

(e) 3125 Trials (21.0 Sec). (f) 15625 Trials (102.7 Sec).

Figure 8: *The NumRndm trials per wire, in the stochastic Algorithm 2, should be a balance between computation times and the quality of the results. Here we show several results similar to Figure 7, using 3D wires that float in the cube, where only NumRndm trials is being modified. 2000 wires, 1 : 1.5 Fairness. See also Figure 9.*

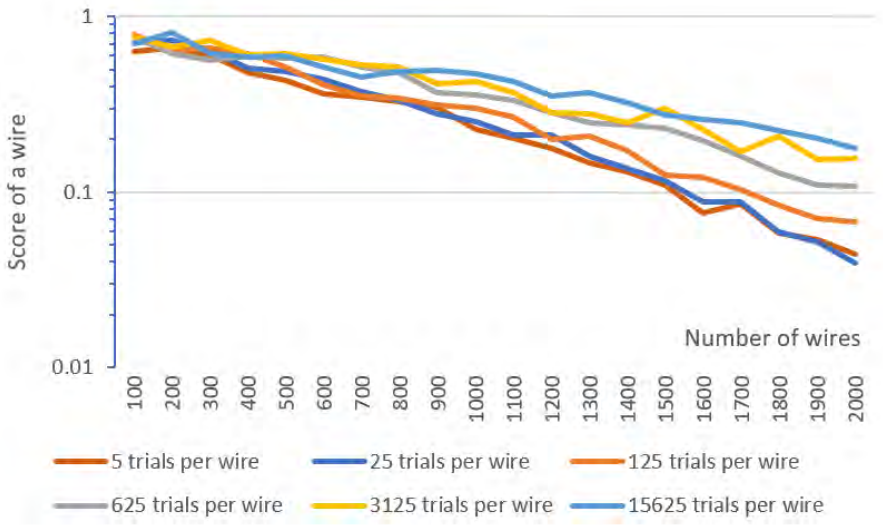
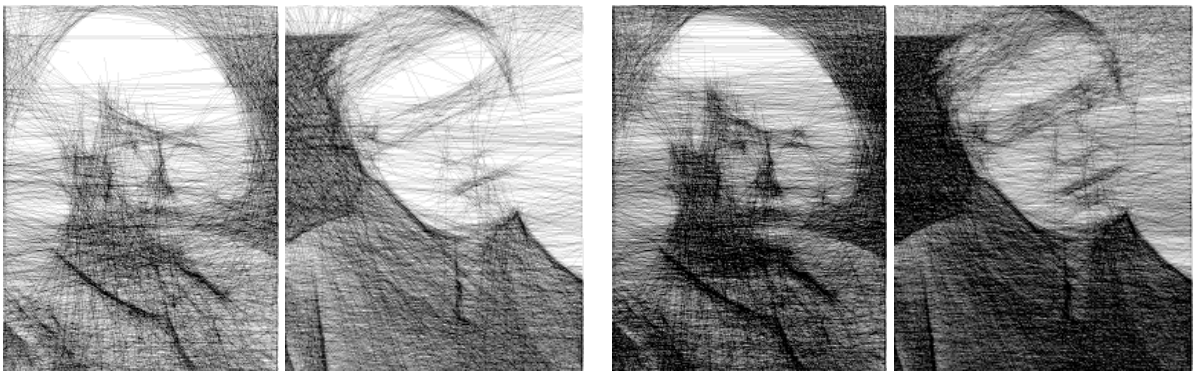


Figure 9: *Some statistics on the scores of the 2000 selected wires (horizonal axis), using data from Figure 8. Shown is the relative score (larger is better, vertical axis) of the selected wire, for the six examples in Figure 8, with different values of NumRndm trials per wire. Note the better scores, for more random trials, as well as the preservation of the scores, as the selection process progresses.*

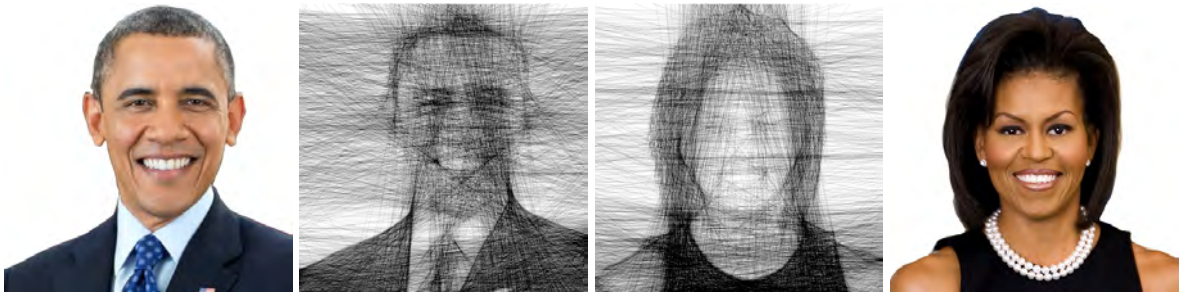
(a) (b) (c) (d)

Figure 10: *The image of Albert Einstein in (a) and Stephen Hawking in (d), are simultaneously DBW, with wires that terminate on the faces of a cube and the results are shown in (b) and (c). 5500 wires, 1 : 1 Fairness.*



(a) 2500 wires. (b) 4500 wires.

Figure 11: *The number of wires has an immediate affect over the results. Here we employ all the parameters as in Figure 10 except the number of wires. Compare with Figure 10.*



(a) (b) (c) (d)

Figure 12: *The image of Barack Obama on the left and a picture of Michelle Obama on the right are simultaneously DBW, with wires that terminate on the faces of a cube and the results are shown at the center. 4000 wires, 1.5 : 1 Fairness.*

with the fairness degree of freedom, in which high fairness for an image means that this image is considered more important. Finally, in Figure 14, we examine the influence of the features (saliency) in the image (recall Figure 2).

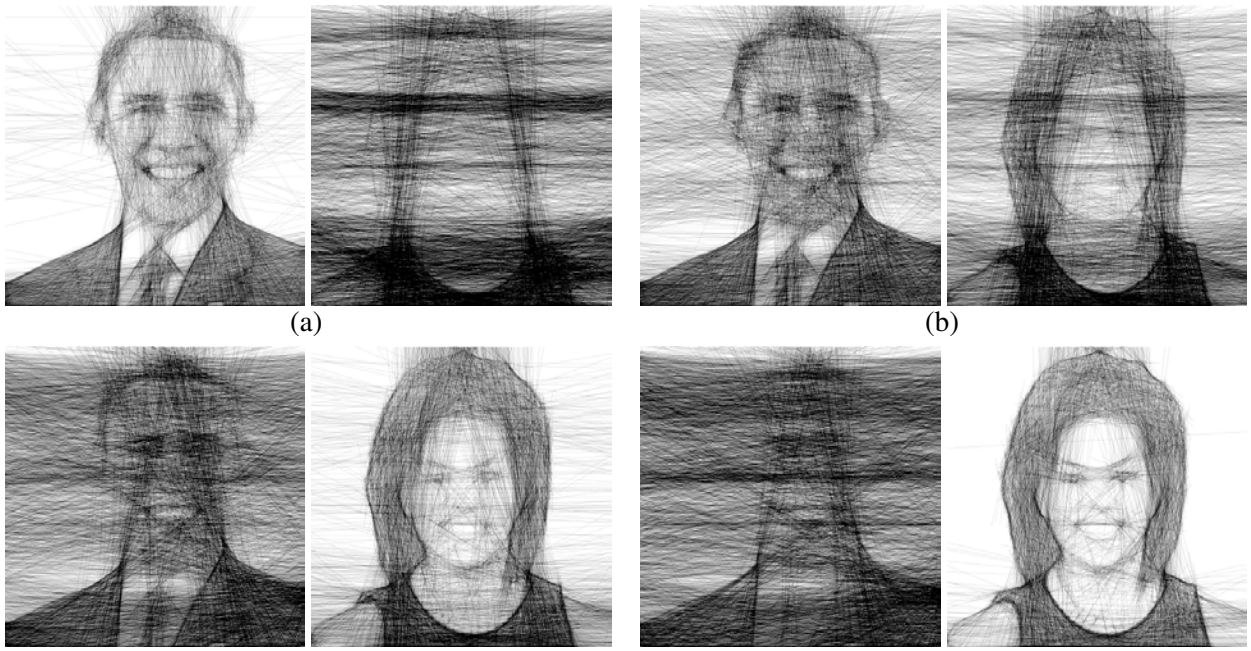
We complete our results section with a few colored ditherings. So far we have assumed the wires are black. Specifically, in Algorithm 3, Line 5, we compared against a black wire. However, one can perform two traversals over the pixels of a wire. The first pass will accumulate the current colors in the input image along the wire, only to compute an average color that is the best for this wire. Then, the second pass will be similar to Algorithm 3, Line 5, only this time the comparison will not be to a black color, but to the computed average color from the first pass.

In Figure 15, two independent examples are dithered from outdoor sceneries, using colored wires of arbitrary length. Figure 16 simultaneously DBW two sunflower drawings of Van Gogh, with wires of arbitrary length, and Figure 17 shows an example of a simultaneous DBW of two outdoor desert sceneries, with wires that span and ends on different faces of a cube.

Finally, Table 1 provides some statistics on several of the examples presented in this work. Timing information, input image sizes and number of wires are portrayed.

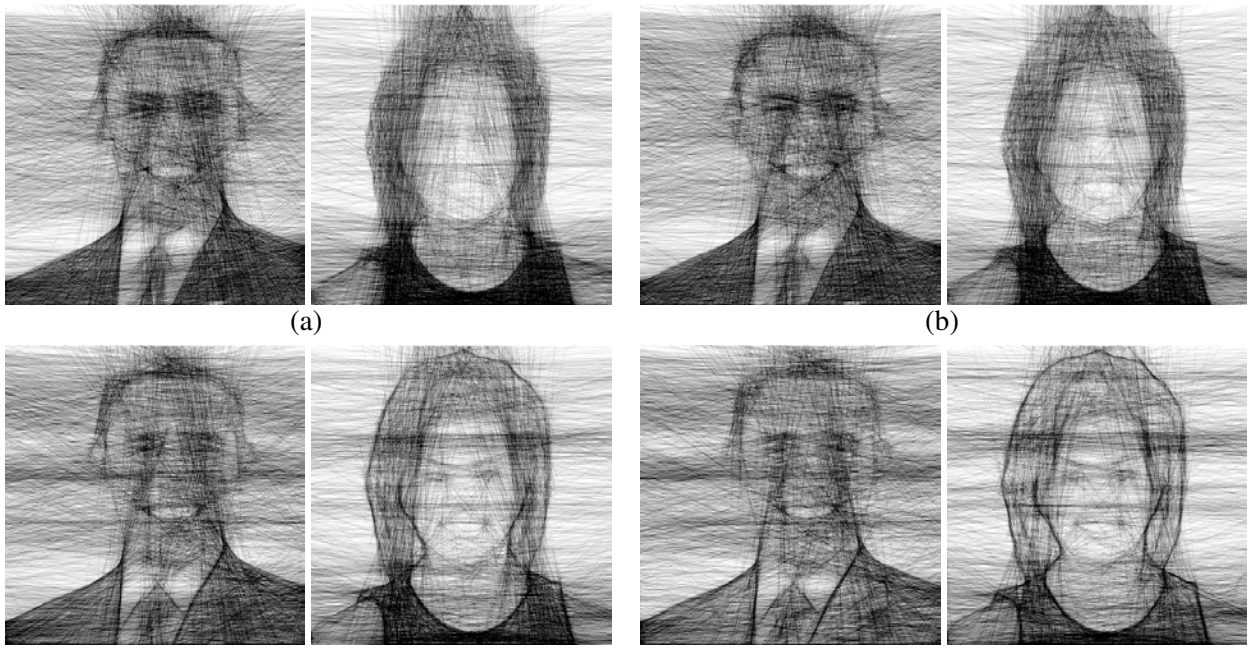
5 Conclusions and Future Work

In this work, we have shown that 3D dithering by wires is feasible, for 3D wires that dither a pair of images, simultaneously. Clearly, this work opens directions for further possible extensions. To begin with and while



(a) (b) (c) (d)

Figure 13: *Figure 12 showed the Obamas with a minor biased fairness toward Barack. Here, in (a), the same example is shown with a strong biased toward Barack, (b) is weakly biased toward Barack, (c) shows similar fairness and (d) is weakly biased toward Michelle. A weak bias toward Barack is probably yielding the best balanced result for both. 4000 wires.*

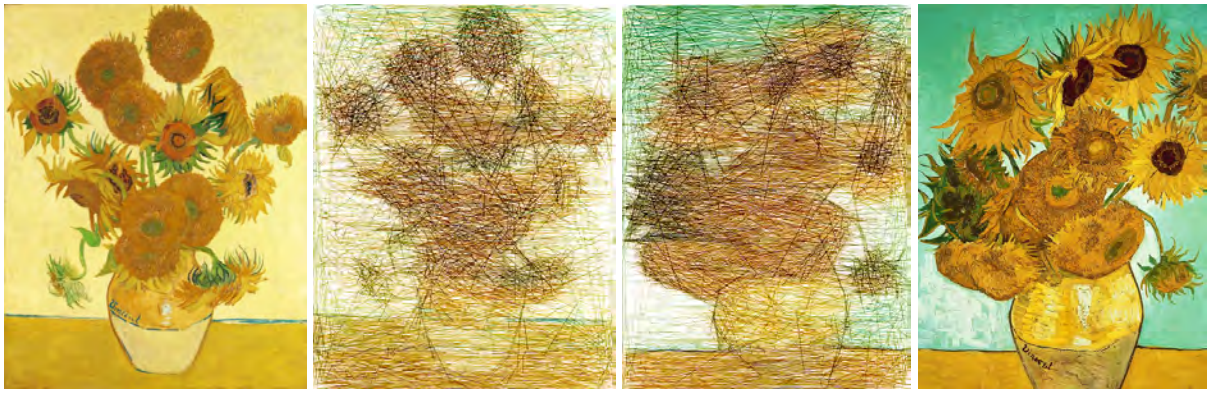


(a) (b) (c) (d)

Figure 14: *Examining the influence of feature enhancements (here edge detection - recall Figure 2). Reduced to no feature importance in (a), minimal feature importance is shown in (b), larger than usual feature importance is presented in (c) and very large importance to features is displayed in (d). All other parameters are the same as in Figure 12.*

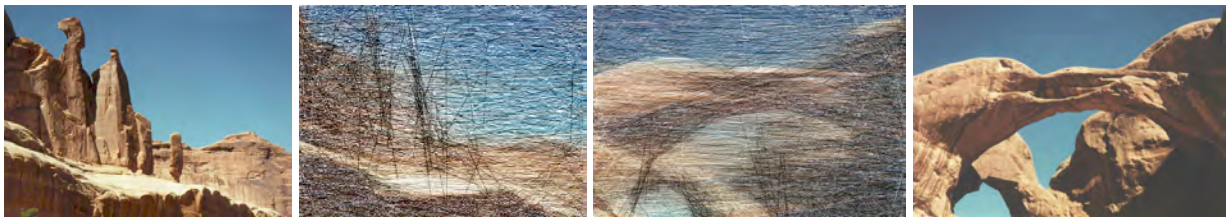
(a) (b) (c) (d)

Figure 15: *Two independent colored wire ditherings, in (b) and (d), of outdoor sceneries of lakes in Utah, in (a) and (c). The dithering was performed on each individual image, with wires of arbitrary length. 5000 wires, 1 : 1 Fairness.*



(a) (b) (c) (d)

Figure 16: *A simultaneous colored wire dithering, with colored wires of arbitrary length, in (b) and (c), of Sunflowers by Van Goch in (a) and (d). 5000 wires, 1 : 1 Fairness.*



(a) (b) (c) (d)

Figure 17: *A simultaneous colored dithering, with colored wires that terminate on the faces of a cube, in (b) and (c), of outdoor sceneries from parks in Utah, in (a) and (d). 5000 wires, 1 : 1 Fairness.*

Figure	Computation time	Input images size	# of wires
5	10.1 sec.	256×256	2000
6	7.25 sec.	256×256	2000
10	23.8 sec.	196×240	5500
12	66.3 sec.	400×400	4000
16	120.1 sec.	400×500	5000

Table 1: *Some statistics on some of the presented examples.*

we focused on two dithered images in this work, the support of three orthogonal images (on the XY, XZ, and YZ planes) is feasible as well. Interestingly, more than three views could also be considered, not necessarily orthogonal!

The assembly of these results as real tangible objects is clearly a worthy goal, and the fact that we can handle wires that start and end on the faces of a cube makes it a bit simpler. In [3], a robotic arm has been used for 2D assembly of string arts. One can foresee a multi-axis robot with a long needle as its end-effector, stitching these wires one at a time, between two faces of the cube, possibly made from some fabric or net. For example, the needle will be inserted into the cubical space along the wire's direction, from the entry face, stitching the far end at the other face, only to retract and stitch the near end at the entry face.

To make sure no two wires collide in this stitching process, the presented algorithms could be augmented with a collision detection test between wires. Newly selected wires that are too close to previously selected wires will be simply purged in favor of other random wires. Then, the final result will ensure the needle-assemblability of the entire wire arrangement.

Fixing the location of the pins, as is done in the combinatorial exhaustive algorithm (recall Figure 3) can help in the assembly as well. Further, one can aim at DBW of a single long wire, where the end point of the i'th wire will serve as the starting point of wire $i+1$th. Such an arrangement will further ease the assembly process.

In this work we employed linear wires. However, bent (metal) wires could be considered as well, with a significant added complexity of searching the next best wire and its shape.

As shown in the end of Section 4, initial experimentations with colored wires have shown promise, and should be further explored. In fact, wires in images with high resolution sharp details, like trees, performed surprisingly well (i.e., Figure 15).

We have introduced one optimization criteria to select the best wire, which examines the best darkening contribution for grey level. Clearly other possible optimization criteria should be examined.

Another future direction to consider is to possibly employ the (inverse) Radon transform [4] toward DBW from multiple directions, a transform that is typically employed in medical volumetric reconstruction. Similarly, one can consider the exploitation of the Hough transform [1], possibly seeking line (or arc) features, simultaneously, in the input images. Finally, a more advanced saliency computation approach and taking aliasing in the wires into account, in the computation of the best wire, could improve the quality of the result.

6 Acknowledgement

The authors would like to thank anonymous reviewers for their comments which were very useful in improving the quality and expository style of this paper.

This research was supported in part by the ISRAEL SCIENCE FOUNDATION (grant No. 597/18) and in part with funding from the Defense Advanced Research Projects Agency (DARPA), under contract HR0011-17-2-0028. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

7 Videos

is

Several videos exemplifying the 3D wires in space, can be seen in https://youtu.be/OJs6pLu5DCg, https://youtu.be/BR5tOlm5YAU, and https://youtu.be/NFCzW1qfdX4

References

[1] The hough transform, https://en.wikipedia.org/wiki/Hough_transform.

[2] Linify me, http://linify.me/.

[3] Birsak, M., Rist, F., Wonka, P., and Musialski, P. String art: Towards computational fabrication of string images. *Computer Graphics Forum 37*, 2 (May 2018), 263–274.

[4] Deans, S. R. *The Radon Transform and some of Its Applications*. Dover Publications, Inc., Mineola, New York, 1993.

[5] Drougard, M. M. Knit report, https://github.com/MaloDrougard/knit/blob/master/Doc/knit-final-report.pdf. Tech. rep., Swiss Federal Institute of Technology Lausanne, 2017.

[6] Elber, G. Ortho-pictures: 3d objects from independent 2d data sets. In *Advanced in Architectual Geometry* (Vienna, 2010), Springer-Verlag, pp. 175–192.

[7] Elber, G. The irit modeling environment, version 12. "http://www.cs.technion.ac.il/~irit, June 2021.

[8] Foley, D., van Dam, A., Feiner, S. K., , and Hughes, J. F. *Fundamentals of Interactive Computer Graphics*. Addison Wesley, second edition, 1990.

[9] Gooch, B., and Gooch, A. *Non-Photorealistic Rendering*. A.K. Peters Ltd. Publishers ISBN: 1-56881-133-0, 2001.

[10] Mitra, N. J., and Pauly, M. Shadow art. In *ACM SIGGRAPH Asia 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH Asia '09, Association for Computing Machinery.

[11] Reif, R. Drawing with straight lines, https://www.robertoreif.com/blog/2018/1/7/drawing-with-straight-lines.

[12] Siegel, C. Knitter, https://github.com/christiansiegel/knitter.

[13] Sobel, I. The sobel edge detector, https://en.wikipedia.org/wiki/Sobel_operator.

[14] Vrellis, P. A new way to knit (2016), http://artof01.com/vrellis/works/knit.html, 2016.