3D-Dithered Ortho-Pictures: 3D Models from Independent 2D Images

Gershon Elber
Dept. of Computer Science, Technion – IIT, Haifa 32000, Israel
gershon@cs.technion.ac.il

Abstract

This work portrays a scheme to simultaneously 3D-dither two (or more) 2D gray-level images in $\mathbb{R}^3$, in orthogonal orthographic views, into one 3D model embedded in a cube, so that the different input images are seen from the different faces of the cube. From one axis-parallel orthographic view of the cube model, the first image is seen, and from a second, orthogonal, orthographic view, the second image is seen. We show that the dithering problem of more than one image does not have an exact solution as one image cannot be completely decoupled from the other; however, for images with rich enough gray-levels, the result will be a highly precise 3D-dithering of both images. Moreover, error correction methods common in classical dithering, such as the well known Floyd-Steinberg [8] algorithm, can be exploited in this 3D-dithering scheme. We then present some tangible examples, etched in glass (See Figure 1).

1 Introduction

The process of dithering a 2D image, as is described in any fundamental computer graphics book [9], follows two main steps, in general. In the first, a priori, stage, i/n^2, $i = 0...n^2$ gray-levels are encoded in 2D-dithering matrices of size $(n \times n)$. Each such dithering matrix encodes a gray-level between zero and one, of one pixel, and is created by painting the n^2 entries in the dithering matrix black or white. In this work, we henceforth denote the entries of the dither matrix as *micro-pixels*. As an example, for $n = 3$, $n^2 = 9$ micro-pixels discretely create $n^2 + 1 = 10$ different gray-levels. In the ensuing discussion, we will interchangeably consider gray-levels to be between zero and n^2 or between zero and one, as i/n^2, $i = 0...n^2$, depending on the context, and whenever it is clear.

In the second stage of the traditional 2D-dithering process and given an image of size $(K \times K)$, colored[1] or gray-level, each pixel is quantized to one of the $n^2 + 1$ discretely available intensity gray-levels. The closest quantized intensity is selected for every gray-level pixel and, possibly, the error between the original gray-level at the pixel and the quantized level can be computed. That said, this error can then be diffused or propagated to the neighboring pixels using, for example, the well known Floyd-Steinberg [8, 9] algorithm.

Given two images of size $(K \times K)$, this paper proposes a simultaneous 3D-dithering scheme that operates by meticulously positioning, typically K^2, voxels into a discrete cube volume of size $(K \times K \times K)$. Each voxel is further assigned a 3D-dithering matrix and then divided into *micro-voxels*, again in a 3D micro-grid, of size $(n \times n \times n)$, where up to n^2 micro-voxels are set. Several video examples of different such etched glass cubes can be seen at http://www.cs.technion.ac.il/∼gershon/V3dDithered.

[1] Colors can be converted to gray-levels via the accepted CIE formula of $Gray = Red * 0.3 + Green * 0.59 + Blue * 0.11$ [9]

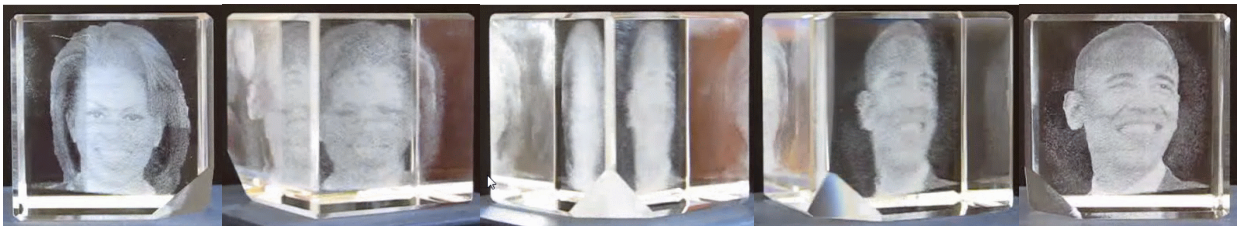


Figure 1 : *3D-dithering of two 2D images of the Obamas (left and right), etched in one 3D glass cube.*

The end result of this arrangement is a 3D model, formed out of $O(n^2K^2)$ micro-voxels, which looks like the first gray-level image from one view and like the second gray-level image from an orthogonal view. See, for example, Figure 1 that is an example of such a 3D dithered model that is etched in glass.

The rest of this work is organized as follows. Section 2 surveys the state of the art in this area. Section 3 presents the different stages of the basic 3D-dithering algorithm and in Section 4, some extensions are considered. In Section 5, additional results are presented and we conclude.

2 Previous Work

This work explores the synthesis of 3D-dithered geometry that projects to different gray-level images in different directions. The outcome has merit in both the sciences and the arts, with relevance to the areas of geometric modeling and the plastic arts, specifically. Hence, we now present related previous work that constitutes a nexus between the geometric modeling community and artistically-oriented design. This, while recalling that the vast majority of contemporary computer-based geometric modeling packages are geared toward mechanical design, focusing on the creation of 3D models that satisfy precise design and/or manufacturing needs.

Some artists specialize in creating models that look different from different views. Yaacov Agam [2] created 3D kinetic statues and used technologies such as lenticular printing. His Agamographs look completely different from different views, exploiting parallel vertical strips of different images that face different directions. Interestingly enough, the recent work of [4, 7] can be seen as an extension of Agam [2], creating shading that follows two input images from two different views by controlling the shapes of zero dimensional small elements like ellipsoids and boxes in [7] and pyramids in [4], instead of the one dimensional strips.

Shigeo Fukuda [10] created beautiful work such as "Duet", "Love Story" and "Cat/Mouse", all depicting two different scenes from two orthogonal directions. Other relevant artists are Markus Raetz [14] and Francis Tabary [16], who also sculpt such work; for example pieces showing one word from one direction, and another word, usually the antonym, from another. Also relevant is the "Escher for Real" project [6] that presents 3D geometry that looks completely different from different view direction; see Figure 2 for an example.

In [13], outline shadows are used as a source for synthesizing such 3D models, which projects to the different outlines in different views. Since the solution does not always exist, the problem is posed and solved as an optimization problem.

In [15], the problem of creating a 3D model that resembles two different shapes from two different views is also posed as an optimization problem of deforming a given model to follow the silhouettes of a different model. Figure 3 shows one result of [15]. Another noteworthy and somewhat related work is [12] where 3D Halftoning is defined toward tiling and filling of volumetric data sets such as porous materials.

Figure 2: *An artifact from the "Escher for Real" project [6] that looks like the Jewish/Israeli 'Menora' symbol/emblem and the Technion logo from two different views.*

The vast majority of this state-of-the-art work dealt with the *silhouettes* or outlines of the geometry and was incapable of handling gray-levels and shading. The noted exception is the work of Y. Agam [2] and his Agamographs and its extensions in [4, 7]. Indeed, our work herein advances this concept and explores the ability of presenting completely intermixed two or more gray-level images, as a one 3D *dithered* model.

3 Algorithm

We seek to derive an ability to simultaneously dither, as precisely as possible, two (or even three or more) 2D images of size K^2 pixels in a single 3D model – a cube of size $(K \times K \times K)$. The 3D cube model will be tiled by at least K^2 voxels, in 3-space, while typically, K^2 voxels are used. These K^2 voxels should cover all K^2 pixels of one image when projected along the $\vec{X}$ direction (YZ plane) presenting the first image[2]. In addition, when projected along the $\vec{Y}$ direction (XZ plane) another set of K^2 pixels should be covered and the second image should be presented. Further, a third set of K^2 pixels may be covered, and if so a third image should be presented when the same set of K^2 voxels is projected along the $\vec{Z}$ direction (XY plane)).

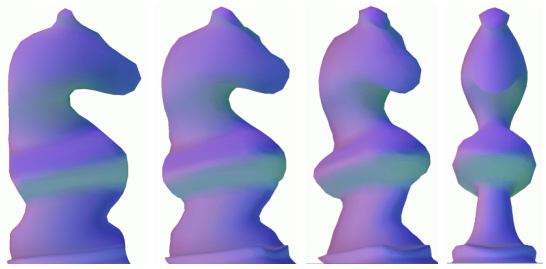


Figure 3 : *The 'knishop' - a combination of a 'knight' (left) and a 'bishop' (right) as one merged smooth 3D model, following [15].*

The problem of placing K^2 voxels in a K^3 cubic volume so that they cover all K^2 pixels when projected along the $\vec{X}$ direction, along the $\vec{Y}$ direction, and possibly along the $\vec{Z}$ direction, has several simple solutions. For a coverage of the K^2 pixels in the $\vec{X}$ and $\vec{Y}$ projection directions only, one can simply place K^2 voxels along the diagonal plane of $X = Y$, at (i, i, k), $0 \leq i, k \leq K-1$. A more general random placement is also feasible via the following observation: Let $\mathcal{P} = Permute(m)$ be some random permutation of the m integers from 0 to $m-1$. Then, Algorithm 3.1 will randomly place K^2 voxels in the K^3 cube so as to cover all K^2 pixels (i.e. constitute a coverage) when projected in the $\vec{X}$ or $\vec{Y}$ directions.

The fact that Algorithm 3.1 constitutes a coverage in the $\vec{X}$ or $\vec{Y}$ directions stems from the observation that in each row (z level) all indices of 0 to $K-1$ are visited in both X and Y. In Algorithm 3.1, we assume that every invocation of $Permute$ results in a new random permutation. Moreover, if the permutation is set to be the identity throughout, i.e. $\mathcal{P}[i] \equiv i$, we are, again, reduced to the diagonal plane, placing voxels in line 5 of the algorithm at (x, x, z);

Algorithm 3.1 (2-projections' 3D coverings)
Input:
K: Size of cube to tile with K^2 voxels;
Output:
$\mathcal{V}$: Set of K^2 3D placements of voxels, covering all K^2 pixels when viewed from $\vec{X}$ or $\vec{Y}$;
Algorithm:

```
1: V ← φ;
2: for z = 0 to K − 1 do
3:     P ← Permute(K);
4:     for x = 0 to K − 1 do
5:         V ← V ∪ {(x, P[x], z)};
6:     end for
7: end for
8: Emit V;
```

Covering all the K^2 pixels by K^2 voxels in 3-space, when projected in the three directions of $\vec{X}$, $\vec{Y}$ and $\vec{Z}$, is also feasible but is a bit more involved. One such feasible placement is along diagonal planes normal to vector $(1, 1, 1)$, as the set (of centers' locations),

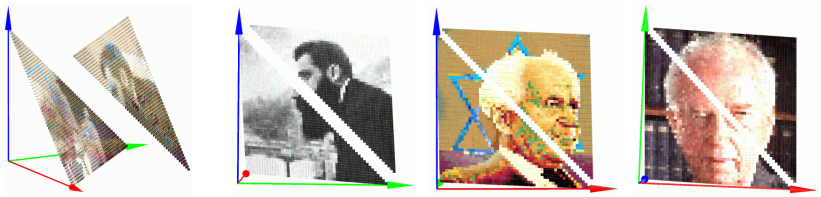
$$\begin{aligned} \mathcal{V} \;=\; \{\, (x,y,z) \mid (x+y+z) \bmod K = 0, \\ x+y+z \neq 0,\; 0 \leq x,y,z < K \,\}, \end{aligned}$$

of $O(K^2)$ voxels which is also shown in Figure 4. Random placement of K^2 pixels that achieves the necessary coverage from the $\vec{X}$, $\vec{Y}$ and $\vec{Z}$ directions is also possible; see [7].

Having the K^2 voxels in place, each voxel is, in turn, formed out of $(n \times n \times n)$ micro-voxels ordered in a 3D-dithering matrix. When projected in the $\vec{X}$, $\vec{Y}$ (and possibly $\vec{Z}$) directions, this voxel will yield the desired gray-level of the relevant pixel in the relevant image, if possible. As a result, the 3D matrix will look like the two (three) given gray-level pixels of the input images, from two (three) different orthogonal views.

[2]From now on and unless otherwise stated, we will assume the placement of only K^2 voxels, which is the obvious minimum requirement while this number can optionally grow up to K^3 voxels, filling the entire 3D cube.

For now, we will assume a 3D-dithering process for two images only. Even this (relatively) simple problem of simultaneously 3D-dithering two images introduces new challenging degrees of freedom and is divided into several stages. In Section 3.1, we will show that for the matching process of pixels in the two images, for 3D dithering, not every gray-level of one pixel, from one view direction, can be a perfect match to every gray-level in a second pixel, in a second view direction. Equipped with the knowledge of what is feasible and what can be expected at the pixel–pixel matching level, in Section 3.2 we examine the matching a whole row of one image with a whole row of the second image. Then, the whole algorithm to 3D-dither two gray-level images in a single 3D cube is finally presented in Section 3.3.



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

Figure 4: *A placement of $O(K^2)$ voxels at $\{(x,y,z) \mid (x+y+z)\ mod\ K = 0,\ x+y+z \neq 0,\ 0 \leq x,y,z < K\}$ (a) covers all K^2 pixels in the projection along the (almost) $\vec{X}$ (b), $\vec{Y}$ (c) and $\vec{Z}$ (d) directions.*

3.1 Pixel–Pixel Matching

Consider two independent pixels of two different orthogonal images, already quantized to two (out of the n^2+1 available) gray-levels, $0 \leq n_x, n_y \leq n^2$. Having a 3D-dithering matrix of size $(n \times n \times n)$ of micro-voxels' cells, we seek to place micro-voxels in the n^3 cells so that when the matrix is projected in $\vec{X}$, exactly n_x micro-voxels will be seen (out of n^2) while a projection of the matrix in $\vec{Y}$ will reveal n_y micro-voxels (out of n^2).

Figure 5 shows one example for the case of $n = 3$. A voxel with $3^3 = 27$ micro-voxels' cells is to be filled up with micro-voxels so that n_x micro-pixels are seen from $\vec{X}$ and n_y micro-pixels are seen from $\vec{Y}$. In effect, this creates a voxel with a gray-level of n_x/n^2 from $\vec{X}$ and a gray-level of n_y/n^2 from $\vec{Y}$. In a 3×3 micro-grid, we can create ten $(n_x, n_y \in \{0,...,9\}/9)$ gray-levels. In Figure 5 (a), three micro-voxels can create a coverage of $n_x = 3$ and $n_y = 1$. Figure 5 (b) reveals a case of three micro-voxels' coverage of $n_x = 3$ and $n_y = 2$, while Figure 5 (c) presents a case of three micro-voxels' coverage of $n_x = 3$ and $n_y = 3$. Possible placements of micro-voxels in these three arrangements are also shown as spheres in Figure 5.

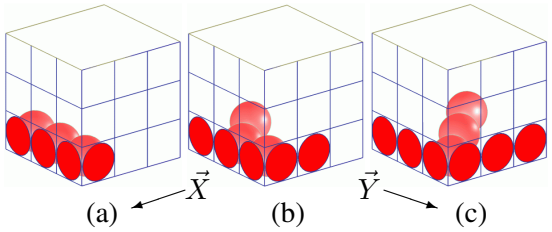


Figure 5: *Having 3 micro-pixels covered from $\vec{X}$ means we can have, for instance, 1 (a), 2 (b), or 3 (c) micro-pixels covered from $\vec{Y}$. Interior micro-voxels are shown as spheres.*

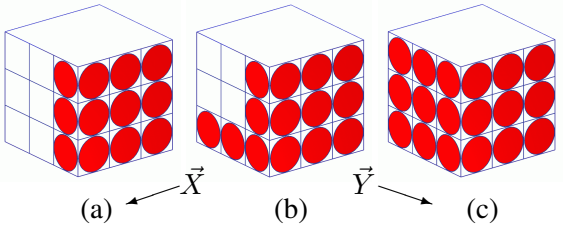


Figure 6: *Having 9 micro-pixels covered from $\vec{Y}$ ($n_y = 9$) means we can have, for example, $n_x = 3$ (a), $n_x = 5$ (b), or $n_x = 9$ (c) micro-pixels covered from $\vec{X}$. In fact, only ($n_x =$) 3 to 9 micro-pixels could be covered from $\vec{X}$, when $n_y = 9$.*

Unfortunately, the difficulty is even greater and beyond this non-uniqueness. Not every gray-level of a pixel from $\vec{X}$ could be matched, using micro-voxels, with every gray-level of a pixel from $\vec{Y}$. Clearly, a coverage of $n_x = 0$ could only be matched against a coverage of $n_y = 0$. Moreover, a coverage of all nine micro-pixels (for $n = 3$), from $\vec{Y}$ ($n_y = 9$), requires a minimum of three micro-voxels covered from $\vec{X}$ ($n_x = 3$). See Figure 6.

A simple inspection of all possibilities of micro-pixel coverage in a matrix of size $n = 3$ (see Table 1) reveals what is possible and what is not. Impossible (n_x, n_y) pairs are the result of close to complete micro-pixel covering in $\vec{X}$ which cannot be concentrated in a few micro-pixel covering in $\vec{Y}$, or vice versa. In other words, very bright images better not be matched with completely dark images. If an infeasible pair $(\tilde{n_x}, \tilde{n_y})$ is forcefully matched, a close feasible pair (n_x, n_y) must be selected, introducing some new gray-level Manhattan distance error,

$$\epsilon = |n_x - \tilde{n_x}| + |n_y - \tilde{n_y}|. \quad (1)$$

This ϵ error can be minimized and/or diffused, a topic we will return to in Section 4.

The 3D-dithering matrices should be built only once. While the optimal (minimal) 3D-dither matrices are not unique, one can select a valid instance and expect a reasonable result. Motivation to alternate among the non unique optimal 3D-dithering matrices could stem from the desire to alleviate Moire patterns [1] in the result. In this case, several alternating dithering matrices could be stored for the same (n_x, n_y) pair and used interchangeably.

Nonetheless and while inherently exponential, a simple algorithm could be written to iterate over all $2^{(n^3)}$ possibilities of 3D-dithering matrices of size n^3, only to select one (or more) instance of a 3D-dithering matrix for every feasible (n_x, n_y) pair. In this work, we have precomputed and employed 3D-dithering matrices for $n = 2, 3, 4$.

Having an understanding of what can (and what cannot) be done at the pixel–pixel matching level, the next section looks at the matching of one complete row from one image with a complete row in a second image.

Cover in $\vec{X}$, n_x	Coverage in $\vec{Y}$, n_y									
	0	1	2	3	4	5	6	7	8	9
0	+	-	-	-	-	-	-	-	-	-
1	-	+	+	+	-	-	-	-	-	-
2	-	+	+	+	+	+	+	-	-	-
3	-	+	+	+	+	+	+	+	+	+
4	-	-	+	+	+	+	+	+	+	+
5	-	-	+	+	+	+	+	+	+	+
6	-	-	+	+	+	+	+	+	+	+
7	-	-	-	+	+	+	+	+	+	+
8	-	-	-	+	+	+	+	+	+	+
9	-	-	-	+	+	+	+	+	+	+

Table 1 : *Coverage possibilities of 3D-dithering of size $(3 \times 3 \times 3)$ micro-pixels from both the $\vec{X}$ and the $\vec{Y}$ viewing directions. A '+' denotes an existing dithering matrix while a '-' hints that none exists. Note that the matrix is always symmetric.*

3.2 Row–Row Matching

At this point, it is clear that the simultaneous 3D-dithering of two images not only introduces errors due to the classic intensity quantization problem, but also due to the inter-dependency in the 3D-dithering matrices between the $\vec{X}$ and the $\vec{Y}$ directions, and the infeasibility of some (n_x, n_y) quantized intensities of pairs of pixels.

Assume two images only, for now, that are inspected from the $\vec{X}$ and $\vec{Y}$ viewing directions. The pixels in a row of one image at some fixed Z level can only be matched with pixels in a row of the second image at the same Z level. Hence, we seek to form a one-to-one match between all the pixels in one row of the first image to all the pixels in one row of the second image; see Figure 7. Denote all the pixels in the row of the first image by $\mathcal{I}^1(p),\ p \in [0, ..., K-1]$ and those of the second im-

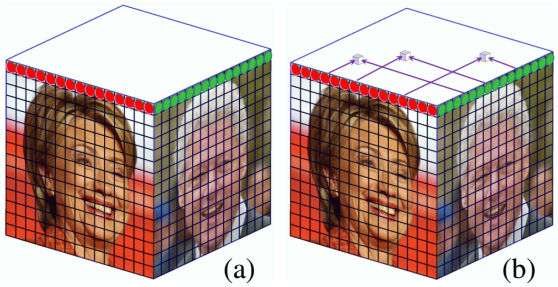


Figure 7 : *Two images are 3D-dithered in a 3D cube model, one row at a time (a). Here, the pixels of one (top level) row of one image, in red, are to be matched with the pixels of a second (top level) row of the second image, in green. Three matched pairs of pixels are presented in (b), as examples.*

age by $\mathcal{I}^2(q),\ q \in [0, ..., K-1]$. Several possibilities for such one-to-one matchings exist:

1. Enforce all voxels to be in one diagonal plane, which means the placements of voxels at $(p, p, z)\ \forall p \in [0, ..., K-1]$. That is, $\mathcal{I}^1(p)$ is matched with $\mathcal{I}^2(p)$. This is likely to generate the largest ϵ error.
2. Allow a completely free row matching, letting any one pixel in $\mathcal{I}^1(p)$ uniquely match any one pixel in $\mathcal{I}^2(p)$, and vice versa. This freedom reduces the problem to a *weighted bipartite graph matching* problem [5] with the following constraints: pixel $\mathcal{I}^1(p_1)$ ($\mathcal{I}^2(q_1)$) must be matched to precisely one pixel $\mathcal{I}^2(q_2)$ ($\mathcal{I}^1(p_2)$) (only "monogamous marriages" are allowed). This is likely to generate the smallest ϵ error.
3. Allow a compromise between the above, 1 and 2, options. That is, $\mathcal{I}^1(p)$ will be matched only with $\mathcal{I}^2(q)$, such that $|p-q| < w$, where w is some bound on the deviation, also denoted as the *width of the matching*. For small w, the 3D model is still bound to an almost diagonal plane. For $w = 1$, this case is reduced to option 1 and for $w = K$, we are reduced to option 2.

Much like [15, 13, 4], we assume a precise set of (orthographic) input views over the model. The larger the width w is, the more distorted the presented image will be, when the 3D model is inspected from a finite distance or a somewhat different view. A partial remedy can be offered here to the influence of the distortion by limiting this w width.

The solution to a weighted bipartite graph matching is well known in computer science [5]. The Hungarian algorithm is used in this work (and hence no code is presented in this Section) to compute the optimal match, with a time complexity of $O(K^3)$ and actual real time performance of a few seconds for $K < 1000$. The weights here are set to zero for intensity matching pairs that have a valid 3D-dithering matrix (i.e. a '+' in Table 1) and the weights are set to be the Manhattan gray-level ϵ error, Equation (1), if invalid (a '$-$' in Table 1). For example, and using Table 1 for 3D-dithering matrices of size $(3 \times 3 \times 3)$, the weight of pair $(4, 5)$, a '+' in Table 1, will be zero but the weight of pair $(4, 0)$, a '$-$', will be two as the distance error in Table 1 to the closest '+' is two.

3.3 Image–Image Matching

We are now ready to combine it all together and create a 3D cube model of size $(K \times K \times K)$ formed out of K^2 voxels, with each voxel being formed out of up to n^2 micro-voxels. Algorithm 3.2 presents these top level steps. $RowRowMatching$ is the algorithm that matches two rows at level z, from Section 3.2, and returns the rows' matched positions. z is provided to $RowRowMatching$ so it can locate the matched points at the correct Z level. Each voxel, with intensity pair (n_x^m, n_y^m), is then tiled with micro-voxels, following the description in Section 3.1.

Algorithm 3.2 (3D image–image dithering)
Input:
K: Output size to tile with K^2 3D-dithering voxels;
n: 3D Dithering size of micro-voxels;
$\mathcal{I}_{ij}^1$, $\mathcal{I}_{ij}^2$: Two images of size $(K \times K)$;
Output:
$\mathcal{C}$: K^2 voxel positions, $P^m = (x_p^m, y_p^m, z_p^m)$, and intensity pairs, (n_x^m, n_y^m), $m = 1, ..., K^2$, $0 \le n_x^m, n_y^m < n$, to 3D-dither images $\mathcal{I}_{ij}^1$ from $\vec{X}$ and $\mathcal{I}_{ij}^2$ from $\vec{Y}$;
Algorithm:
1: $\mathcal{P} \leftarrow \phi$;
2: ***for*** $z = 0$ *to* $K-1$ ***do***
3: $\quad \mathcal{I}_z^1 \leftarrow \mathcal{I}_{iz}^1, \forall i$;
4: $\quad \mathcal{I}_z^2 \leftarrow \mathcal{I}_{iz}^2, \forall i$;
5: $\quad \mathcal{P} \leftarrow \mathcal{P} \cup RowRowMatching(\mathcal{I}_z^1, \mathcal{I}_z^2, z)$;
6: ***end for***
7: $\mathcal{C} \leftarrow \phi$;
8: ***for*** *each* P^m *in* $\mathcal{P}$ ***do***
9: $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \left\{ P^m, (n_x^m, n_y^m) \right\}$;
10: ***end for***
11: *Emit* $\mathcal{C}$;

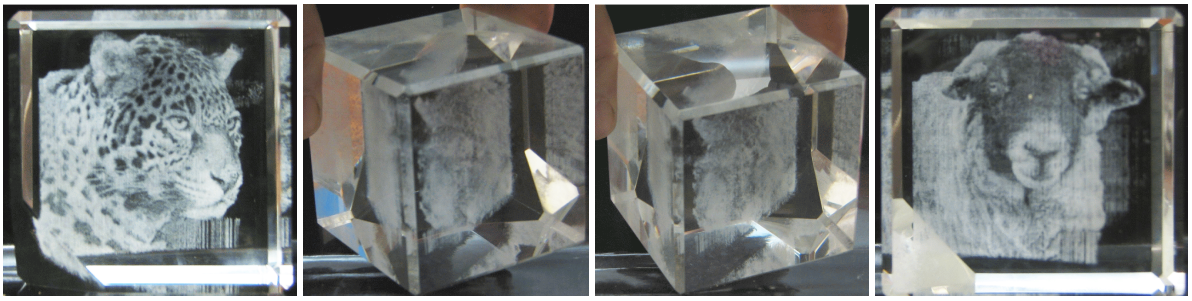


Figure 9 : *An example of two pictures etched in one 3D cube of glass, of a tiger and a lamb.*

4 Possible Extensions

The error between the desired (original) pixel intensity and the actual intensity can be accumulated for 3D-dithering of two images much like in the regular single 2D image dithering. Hence, classical error diffusion techniques can be used and are capable of alleviating the final global error, for example by adapting a variation of Floyd-Steinberg [8, 9] algorithm.

Because the width of the matching, w, is typically greater than one pixel and since the expected pixel intensities in a row are set before the matching algorithm is applied, propagation of the dithering error to neighboring pixels is limited in this work to pixels in the next row only. The classic Floyd-Steinberg algorithm suggests the diffusion of the error that is introduced in pixel (x, y) as in Figure 8 (a) whereas herein, we purged the $(x + 1, y)$ direction, being in the same row, and instead use the table shown in Figure 8 (b).

A central question throughout the discussion so far has been the extension of the presented solutions to three (or more) images viewed from the $\vec{X}$, $\vec{Y}$, and $\vec{Z}$ directions. Portions of the answer were already introduced when we showed how to tile the K^3 volume using K^2 voxels so that all the K^2 pixels of the three images, which are viewed from $\vec{X}$, $\vec{Y}$, and $\vec{Z}$, are covered. With this ability, 3D-dithering matrices can also be designed with prescribed intensities from three orthogonal directions, a tedious task that again must be computed only once, delineating the feasible supported tri-gray-levels from those that are not.

The rest of the solution to the simultaneous dithering of three images is conceptually simple but computationally challenging. While the proper placement of K^2 voxels in a cube so they cover three images from three different directions is feasible, a *tripartite graph matching*, seeking the optimal matching (placement) for *three* sets, is to be applied. Unfortunately, the optimal solution to weighted tripartite matching is known to have exponential complexity. Only heuristic approaches could be expected to be feasible. See, for example, [3].

Pixel	Weight
$(x+1, y)$	7/16
$(x-1, y+1)$	3/16
$(x, y+1)$	5/16
$(x+1, y+1)$	1/16

(a)

Pixel	Weight
$(x-1, y+1)$	3/9
$(x, y+1)$	5/9
$(x+1, y+1)$	1/9

(b)

Figure 8 : *Error diffusion used in the classic Floyd-Steinberg algorithm (a) and herein (b).*

5 Additional Examples and Conclusions

We complete our presentation with two additional etched-in-glass examples, of pictures of a tiger and a lamb, in Figure 9, and of an Einstein portrait and his famous matter-energy conversion equation, duplicated many times, in the other image, in Figure 10. These glass cubes (I.e. Figures 1, 9 and 10) are 60mm on the side and contain an order of a million micro-voxels each, etched in glass using an existing focused laser beam technology.

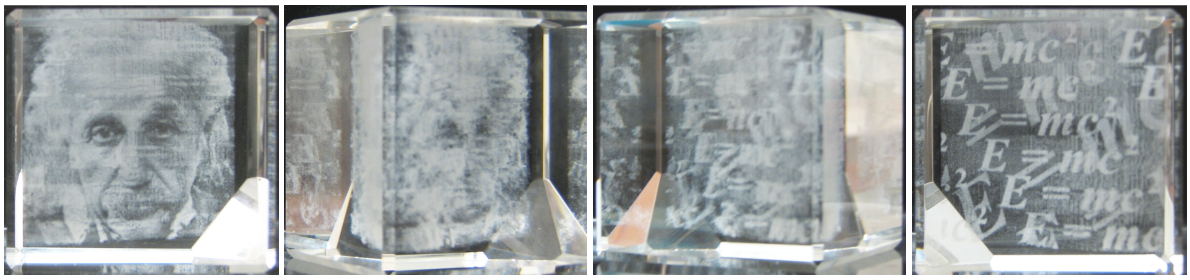


Figure 10 : *A portrait is matched with randomly spread text etched in one 3D cube of glass.*

We presented algorithms to derive the necessary set of points to 3D dither two orthogonal 2D images. Clearly this work can be extended in a variety of direction, having more than two images dithered together. Other directions of extending this work will be to relax the strict one-to-one matching, and the support of colors, although contemporary glass etching technology barely supports two colors.

6 Acknowledgments

This work was supported in part by the ISRAEL SCIENCE FOUNDATION (grant No.278/13). I would like to thank Tomer Vromen for implementing the Hungarian algorithm. I would also like to thank Lev Dvorkin and Gregory Gisinsky for their help in making the real tangible glass cubes.

References

[1] Moire patterns. http://en.wikipedia.org/wiki/Moire_pattern.

[2] AGAM, Y. http://en.wikipedia.org/wiki/Yaacov_Agam.

[3] AIEX, R. M., RESENDE, M. G. C., PARDALOS, P. M., AND TORALDO, G. Grasp with path relinking for three-index assignment. *INFORMS Journal on Computing 17* (2005), 224–247.

[4] ALEXA, M., AND MATUSIK, W. Reliefs as images. In *ACM SIGGRAPH 2010 papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 60:1–60:7.

[5] CORMEN, T. H., LEISERSON, C. E., AND RIVEST, R. L. *Introduction to Algorithms*. The MIT Press, 2009.

[6] ELBER, G. "Beyond Escher for Real" project, 2002. http://www.cs.technion.ac.il/∼gershon/BeyondEscherForReal.

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

[8] FLOYD, R. W., AND STEINBERG, L. An Adaptive Algorithm for Spatial Greyscale. *Proceedings of the Society for Information Display 17*, 2 (1976), 75–77.

[9] FOLEY, J. D., VAN DAM, A., FEINER, S. K., AND HUGHES, J. F. *Fundamentals of Interactive Computer Graphics*. Addison-Wesley Publishing Company, second edition, 1990.

[10] FUKUDA, S. http://en.wikipedia.org/wiki/Shigeo_Fukuda.

[11] IRIT. The irit geometric modeling environment 2010. http://www.cs.technion.ac.il/∼irit.

[12] LOU, Q., AND STUCKI, P. Fundamentals of 3d halftoning. In *Electronic Publishing, Artistic Imaging, and Digital Typography. Lecture Notes in Computer Science* (1998), Springer, pp. 224–239.

[13] MITRA, N. J., AND PAULY, M. Shadow art. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers* (New York, NY, USA, 2009), ACM, pp. 1–7.

[14] RAETZ, M. http://www.crownpoint.com/artists/raetz.

[15] SELA, G., AND ELBER, G. Generation of view dependent models using free form deformation. *The Visual Computer 23* (2007), 219–229.

[16] TABARY, F. http://www.francistabary.com.