Signal, Image and Video Processing manuscript No.
(will be inserted by the editor)

Stereoscopic Image Stippling

Chuan-Kai Yang · Chien-Yu Hou

Received: date / Accepted: date

Abstract In recent years, stereoscopy has become quite popular and has attracted the attention from many researchers, leading to numerous research studies on stereoscopic image processing, editing, and stylization. In particular, it is to our attention that, in terms of stereoscopic image stippling, one type of image stylization, there is still much room for further improvement over some existing approach, which could suffer from the effect of *binocular rivalry*. In this study, assuming the input is an *anaglyph* or *red-cyan* image pair, we propose approaches to convert the input into two styles of stereoscopic image stippling, i.e., *simple* and *hybrid*, where the first one is with stipples of the same size, while the second one with stipples of different sizes. Several experiments, including one stereoscopic test and 5 user studies, have been conducted, to justify the effectiveness of our proposed approaches and improvement over some existing method.

Keywords Stereoscopic Images · Stippling Images · Disparity Maps · Image Segmentation

1 Introduction

Stereoscopy has become tremendously successful in recent years, and its wide range of applications ranging from movie industry to game industry have attracted many related researches. The basic idea of stereoscopy is to make use of the *disparities* between two corresponding images, as our brain could combine such an image pair to gain a stereoscopic feeling of the *reconstructed scene*, as long as both images satisfy *stereo consistency*, which means the disparities are mostly *horizontal*, and within a certain range. In terms of displaying technology, there are many types of stereoscopic display, such as *polarization systems*, *color anaglyph systems*, *autostereoscopy systems*, etc., and in this study, we focus only on *color anaglyph systems*, although we believe many of our algorithms could be easily generalized or ported into other systems. In terms of research, one popular trend is to extend existing researches on images to stereoscopic images, for example, *image retargetting*, *image warping*, *image inpainting*, and *image stylization*, etc., and the most important principle is to make sure the generated results still carry the stereo-

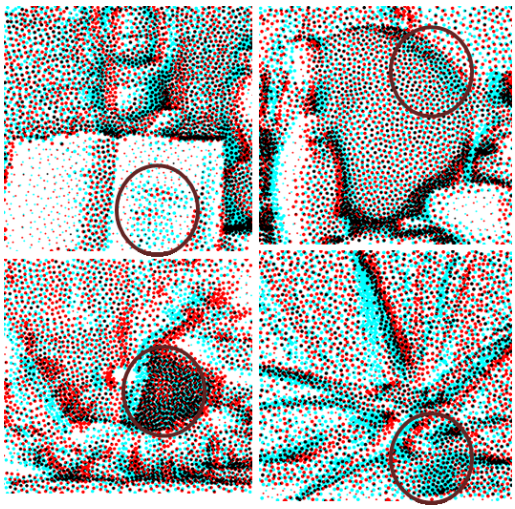


Fig. 1: Examples of binocular rivalry, and the places of occurrence are marked by circles.

Chuan-Kai Yang · Chien-Yu Hou
Department of Information Management,
National Taiwan University of Science and Technology,
43, Section 4, Keelung Road, Taipei, 10607, Taiwan
Tel.: +886-2-27376756
Fax: +886-2-27375777
E-mail: ckyang@cs.ntust.edu.tw,M10109115@mail.ntust.edu.tw

scopic feelings. Notice that the naive way of applying the aforementioned techniques originally designed on an single image to both the left and right images independently of the input stereoscopic image pair could often cause the effect of *binocular rivalry*, shown in Figure 1, as pointed out by Richardt et al. [20], who also devised a way to measure the *stereo consistency*. This is mainly due to the fact that the processing independently done on both images could undesirably destroy the correlation and/or disparities meant to preserve the stereoscopic feelings.

To solve this, Northam et al. [17] proposed a framework, through which many of the image stylizations can be carried out to be still stereoscopic, and their system is a *color anaglyph system*, where the input and output can be seen through *red-cyan* glasses. However, it is to our attention that the *image stippling* results are not particularly good. To address this, in this study, we propose two *image stippling* styles, *simple* and *hybrid*, to not only improve the existing approach, but also to show some different ways of performing image stippling. As mentioned, independently converting the left and right images into their corresponding stippling counterparts would not work due to *binocular rivalry*, we therefore opt to combine the involved stipple sampling scheme, the *disparity maps*, *occlusion maps*, and a *mutual checking mechanism* to ensure the converted results not only to faithfully carry the desired stereoscopic feelings, but also successfully present the two proposed stippling styles. We have conducted numerous experiments, including a stereoscopic test and 5 user studies, while within which we have also made comparisons with the result from Northam et al. [17] to prove our claim. In addition, comparisons between the *simple* and *hybrid* stippling styles have also been made to see the various preferences among the persons involved in the experiments.

The rest of the paper is organized as follows. Section 2 reviews the literature related to this work. Section 3 describes the procedures involved to convert a stereoscopic image pair into a stereoscopic stippling pair. Section 4 shows the results generated by the proposed system and compares them with others when applicable. Section 5 concludes this work and hints for some potential future directions.

2 Related Work

In terms of related work, there are at least three types of research fields involved, namely *stereoscopy correspondence*, *stereoscopic image processing* and *stippling*. Each of these research fields will be discussed in turn.

The first one is *stereoscopy correspondence*. Given a pair of images or even more than two images, the task is to identify, for each pixel on one of these images, its corresponding pixel(s), if any, on the other image(s). By corresponding pixels properly it could make a human brain have the impression of depth, thus the feeling of stereoscopy. There are already several approaches proposed; however, none of them is perfect in terms of quality and performance, and as a result, this issue has still attracted new researches till today. Scharstein et al. [21] gave a taxonomy of existing stereoscopy correspondence algorithms, and they also provided a software platform and a collection of data sets [2] for the purpose of evaluation. Klaus et al. [8] proposed to use *mean-shift* for *image segmentation* to improve the accuracy of stereo correspondence. The basic idea is that the same segment of pixels should have their depths falling within a continuous range, while depth discontinuity should only happen near the segment boundary. After segmentation, *sum of absolute difference*, or *SAD* for short, is adopted to measure the similarity between two segments. The third step is to construct *depth planes*, followed by the final step of performing an *energy optimization* on the results from the previous step to find the best depth value for each pixel. During the optimization process, the algorithm of *Loopy Belief Propagation*, proposed by Felzenszwalb et al. [5], was adopted to find the best depth labeling. Smith et al. [23] proposed to treat each pixel as a *feature vector*, and then the original correspondence problem becomes a matching problem of a *point cloud* in the *feature space*. Similar vectors (points) are combined to form a *connected network*, and a *global stereo matching* is performed on these networks. The correspondence problem is solved through an *energy minimization* framework to derive the proper depth value for each pixel, and finally the disparity value is obtained through the *graph-cut* [3] approach. Hosni et al. [7] proposed a general framework to deal with the core *labeling* issue used for a correspondence problem. The framework starts by assigning a *cost* to each pixel, and then these cost values are *filtered* to smoothen the values, followed by a *winner-take-all* selection rule.

The second one is *stereoscopic image processing*. In general, this refers to the techniques of image processing, where the involved images are mostly stereoscopic image pairs, and the processing in discussion includes traditional image processing, image editing, or image stylization, etc. Note that the most important principle for all these processings is that the resulting images should still be stereoscopic. Lo et al. [10] proposed to perform the *copy & paste* operations on stereoscopic images, that is, a user could cut a portion from one stereo-

scopic image pair and paste it onto another stereoscopic image pair. To be able to achieve this, the work proposed by [23] is used first to construct the *disparity map*, followed by the *mean-shift clustering* proposed by Comanniciu et al. [4] and the cluster merging mechanism proposed by Ning et al. [15] after a user manually selects an object from the left image of the source stereoscopic image pair. Once the objects are roughly identified, *graph-cut* is used to further optimize the object segmentation result. Then the disparity map is used to locate the corresponding object on the right image in the source stereoscopic image pair. A rotation is performed if there exist angular differences between the source the target image pairs. A concept of *stereo billboard* is proposed to retain the size of the object in operation, and cope with the occlusion problem by determining the correct order through the help of disparity map, thus the depth information, derived from a previous step. Finally, shadows are added to make the results look more convincing. Niu et al. [16] proposed a system that allows a user to perform a *warping* on a stereoscopic image pair. The user could either warp or *deform* the left image of the input stereoscopic image pair, while the system automatically adjusts the corresponding left disparity map accordingly to fit the desired change. Then the change is mapped to the right image through a $m \times n$ *image mesh* together with an energy minimization process to find the best location for each pixel. The mapping is also governed by the *SIFT* [11] feature to avoid the potential interference caused by the scaling or rotation operations. Luo et al. [12] also proposed a system that supports stereoscopic *image cloning*, and in particular, is able to handle *perspective distortion*. Furthermore, the technique of *Poisson blending*, proposed by Perez et al. [19], is adopted to smoothly fuse the colors between the source and target regions. Lee et al. [9] proposed to resize a stereoscopic image pair by first calculating the corresponding disparity map, and then decomposing the image pair into multiple *layers* according to color and depth information. The next step is to place a *mesh* on each layer, followed by an *energy minimization* process to achieve the desired resizing effect, while maintaining the image content or quality. Northam et al. [17] performed image stylizations on stereoscopic images. Based on disparity (depth), *masks* and *merged views* can be constructed for each depth layer. If stylization is done layer by layer, there could exist some overlapping inconsistency, instead, they chose to stylize the merged view first, and also borrowed some techniques from the *painterly rendering algorithm* proposed by Hertzmann [6].

The final one is related to *stippling*, which can be thought as a way for performing *non-photorealistic rendering*, or *NPR* for short. In general, it is a way to represent an image where several parameters, including point size, number of points, etc., could vary, thus leading to versatile and artistic results. Secord [22] proposed a real-time image stippling system based on the concept of *weighted Voronoi diagram*. In the initialization step, sample points are chosen and a *Voronoi diagram* can be constructed. A *centroid Voronoi diagram*, or *CVD* for short, can be achieved by iteratively adjusting the sample points toward the centers for all regions. Secord made use of the method proposed by McCool et al. [13] to convert the initial sample points into *CVD*, and introduced a *density function* which allows points to cluster on where the grayscale values of the image are low or dark. Xu et al. [24] proposed to perform stippling by a *Delauney triangulation*, where there are two possible approaches: *uniform* and *non-uniform*. In the first case, it is to find the best locations for stipples by minimizing the difference between each triangle and the average triangle, in terms of area. In the second case, stipples are deployed according to the variable that is related to the image grayscale values or other information, and the goal is to minimize the difference between each triangle and its neighboring triangles, in terms of the variable value. And in general, this approach is faster than the previous approach.

3 Stereoscopic Image Stippling

3.1 System Flow

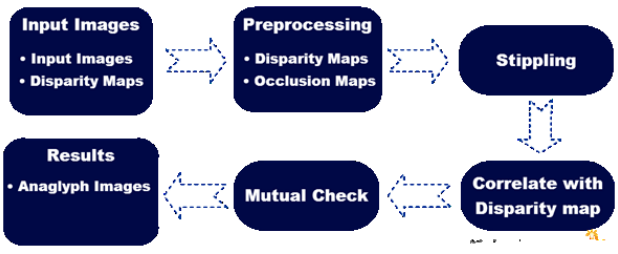


Fig. 2: The system flow of this work.

Figure 2 illustrates the system flow of this work. First, a stereoscopic image pair, together with its corresponding disparity maps, if exist, form the inputs to our system. The next step involves a preprocessing, where the disparity maps and occlusion maps are derived or processed for later use. The third step is to convert the left image of the input image pair into a stippling result. Based on the disparity maps, the right stippling can also be generated in the fourth step. As the inconsistency of parameters used in both stippling may lead

to *binocular rivalry*, a *mutual check* is performed in the fifth step. Finally, the left and right stippling results are combined to generate the desired *anaglyph image pair*.

3.2 Preprocessing

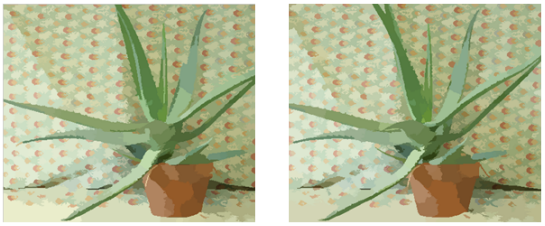
There are some preprocessing tasks before the creation of the stippling results, i.e., the generation of *disparity maps* and *occlusion maps*. The first one is the correlation between the input image pair, which can help derive the depth information of the input, while the second one records the related information where some views can only be seen from one of the left and right images, but not both.

3.2.1 Disparity Map

Disparity maps define the correspondence between the input image pair, and can be used to determine the depth for each pixel. The larger the disparity value is, the larger the depth is for the corresponding pixel. There have been numerous existing researches, while some of them have been discussed in the Related Work section.

There are two types of disparity maps that we use in this work. The first type is to make use of the *Middlebury stereo vision ground truth datasets* [2], where there are still some pixels with uncertain disparity values. Using the *mean-shift* approach we cluster the images in terms of the *RGB* color space, and we assign disparity values to these pixels with more confidence. The second type is to generate disparity maps directly through the metric of *sum of absolute difference*, or *SAD* for short.

Regarding the first type of disparity maps, we first perform an image segmentation on both images of the input image pair based on *EDISON* [1], which in turn is based on *mean-shift* [4] clustering and *edge detection* [14], and the results are shown in Figure 3.



(a) (b)

Fig. 3: Color segmentation results. (a): The left image. (b): The right image.

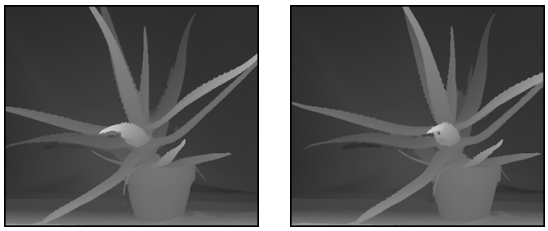
The second step is to apply *8-connected morphological erosion and dilation* to the portions in the images that need to be patched, denoted as Ω, as marked in black in Figure 4(a), in the order of an *erosion* followed by a *dilation* to derive the noise-like image, as shown in Figure 4(b), to become Ω'. The resulting disparity map, after the patching, is shown in Figure 4(c). In other words, Figure 4(c) is the result after we inpaint Figure 4(a) by focusing only on the regions shown in Figure 4(b).

(a) (b) (c)

Fig. 4: Fixing the disparity map. (a): The input disparity map. (b): The regions to inpaint. (c): The resulting disparity map.

More specifically, the second step consists of three operations. First, based on the previous segmentation result, we can build the relationships among segmentation, and from this information, we know the neighboring segments for each given pixel. Second, depending on the *Euclidean distance* between two pixels in the *RGB color space*, we can measure their *similarity*. Assume the disparity value for pixel A needs to be fixed, we then search from the segment it belongs to and its neighboring segments to find the pixel, say pixel B, that is the most similar one to pixel A, and use pixel B's disparity value for pixel A's disparity value, provided pixel B's disparity value is sound. The third step is basically to optimize the result. For a pixel within the aforementioned region Ω whose disparity value requires a fix, we calculate the minimal, maximal and average disparity values from its *8-connected* neighbors, and if the difference between the minimal and maximal values is larger than a threshold, the average value is chosen; otherwise we count the frequency of the minimal and maximal disparity values and select the one with a higher frequency. Such a process can be applied iteratively, and in our current implementation, 5 iterations are used and the results are shown in Figure 5.

As for the second type of disparity maps, we make use of *SAD* to generate the disparity maps. Regarding the *SAD* comparison, it involves a $w \times w$ block size, where w is set to be 15 in this study by referring to [18]. In addition, we use the *Middlebury* dataset [2], with the image size to be 1/4 of the full size, and therefore the depth ranges are also reduced to be 1/4 of their original ranges. To calculate *SAD*, it follows Equation 1, where $I_L(i,j)$ and $I_R(i,j)$ are the pixel values at the



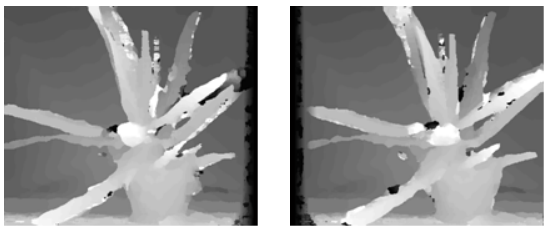
(a) (b)

Fig. 5: Fixed disparity maps. (a): The left disparity map. (b): The right disparity map.

location (i, j) for the left and right images, respectively, d a particular disparity value, and $W(x, y)$ a $w \times w$ block with (x, y) being the center for the block. For each pixel, the best disparity value $\bar{d}$ is determined if it corresponds to the least *SAD*, as described in Equation 2. Figure 6 shows the results using this *SAD* approach.

$$C_{SAD} = \sum_{(i,j) \in W(x,y)} |I_L(i, j) - I_R(i + d, j)| \quad (1)$$

$$disparity \;\; value = \bar{d}, \quad if \;\; C_{SAD}(x, y, \bar{d}) = min(C_{SAD}(x, y)) \quad (2)$$



(a) (b)

Fig. 6: The disparity maps generated by using SAD. (a): The left disparity map. (b): The right disparity map.

3.2.2 Occlusion Map

One important information in stereoscopic image processing is *occlusion maps*, which tell what can be seen from the left image, but not from the right image, and vice versa. The generation of this information is necessary for our ensuing processing.

Given $disp_L$ and $disp_R$, which represent the left disparity map and right disparity map derived previously, essentially we need to compare their values to check if their differences are beyond a threshold range. In this study, we only consider the case when the disparity happens *horizontally*; that is, there exists no *vertical disparity*. How the left occlusion map is generated is

```
tempx = x − disp_L(x, y)
if tempx < 0
   occlusion(x, y) = 0
else {
  if diff_disp < ε
     occlusion(x, y) = 1
  else
     occlusion(x, y) = 0
} /* compare disp_L(x, y), disp_R(tempx, y) */
```

Fig. 7: The pseudo code for computing the left occlusion map.

detailed in Figure 7 (and the right occlusion map can be derived similarly). In this Figure, assume x is a horizontal coordinate for a pixel, and after the disparity mapping, the corresponding horizontal coordinate becomes *tempx*. If *tempx* is outside of the image range, this pixel is *occluded*; otherwise, we compare the disparity difference between these two pixels as described in Equation 3, where $k1$ and $k2$ can be L or R. If the difference is larger than a threshold ϵ, set to be 5 in this study, this pixel is thought to be occluded. As a result, the corresponding occlusion maps are shown in Figure 8, with the occlusion parts represented in black.

$$diff_{disp} = |disp_{k1}(x, y) - disp_{k2}(tempx, y)| \quad (3)$$

(a) (b)

Fig. 8: The resulting occlusion maps generated by using SAD. (a): The left image. (b): The right image.

3.3 Stippling

As mentioned, we use the Middlebury dataset [2], with the image sizes to be 1/4 of their full sizes, and apply the approach proposed by Xu et al. [24] to derive stippling results. The basic idea in [24] is to perform *Delauney triangulation* from a given set of points, and then to calculate the total image information covered under each triangle, such as the grayscale values. The derived information is used to adjust the triangulation, and thus the positions of the points.

It is evident that one crucial thing in stippling is the initial selection of sample points. For this, we adopt a

simple scheme by testing if the *information of a pixel at* (x, y), denoted as $\hat{I}(x, y)$, is larger than or equal to a threshold m or not, and if so, we then place a black stipple at that location, as described in Equation 4. As for how $\hat{I}(x, y)$ is defined is to be given next.

$$I_{stipple} = black, \quad if \quad \hat{I}(x, y) \geq m \tag{4}$$

To determine what should be put into $\hat{I}(x, y)$ with respect to pixel (x, y), we observe that its grayscale value and the *gradient magnitude* of its disparity value play important roles. Also note that this information will be referenced not only for placing the sample points, but also for the *mutual check mechanism* to be discussed a bit later. More specifically, in terms of stippling, it is intuitive to have more points associated with the places where the grayscale values are smaller, or equivalently, the corresponding image pixels are darker. In addition, from a normal image, it is easy to tell the objects in the image by their relative depths, and as a result, it is also intuitive to make use of the depth information, or equivalently, the disparity map information, and in particular, the gradient magnitude of the disparity map values, to help us determine if points should deployed or not. In general, the larger the gradient magnitude for a pixel, the larger the probability it is corresponding to an object's boundary, and therefore the higher the probability to have a sample point at that location. Equation 5 tells how exactly this is achieved, where I_{gray} is the grayscale value for a pixel, $\bigtriangledown D$ the gradient magnitude of its disparity value, α the weight for I_{gray}, β the weight for $\bigtriangledown D$, respectively. Note that both I_{gray} and $\bigtriangledown D$ are normalized to the range from 0 to 1 (inclusive), while $\alpha + \beta = 1$, $\alpha \geq 0$, and $\beta \geq 0$. As a result, the range of $\hat{I}(x, y)$ is also constrained to be in the range from 0 to 1 (inclusive).

$$\hat{I} = \alpha I_{gray} + \beta \bigtriangledown D \tag{5}$$

When generating stippling results, there are two parameters that can affect the results, i.e., the size of a stipple and the number of stipples. A proper setting could help create various stylized results; however, improper choices of these two parameters may lead to problematical results, such as the *binocular rivalry* mentioned previously. More specifically, in the *hybrid* style, where the stipple sizes can be different, the most important thing is to have the same stipple size for the left and right corresponding stipple. For the *simple* style, where all the stipples are of the same size, the number of stipples could affect the rendering results in the following sense: too less stipples may not be able to faithfully represent the original image content, while too many stipples could resemble the original image too much thus making the stippling representation pointless. Fig 36, to be shown later, illustrates more or less this phenomenon while the best number of stipples may be image(content)-dependent. In general, the size of a stipple can be determined from $\hat{I}$, and the larger the value of $\hat{I}$, the larger the corresponding stipple. Figure 9 shows how the stipple size is estimated. First, we calculate the enclosed $\hat{I}$ in each triangle in the Delauney triangulation, and then we take the average of the sum of the information covered from the k neighboring triangles, while each of them is denoted as $\hat{I}^{T_i}$, to become $d_{(x,y)}$, as described in Equation 6, to be used in Equation 7 to determine the size or *radius* of the stipple in use. Note that in Equation 7, $r_{(x,y)}$ is the radius for the stipple placed at pixel (x, y), $min(d)$ the minimal d value calculated from Equation 6, $max(d)$ the maximal d value calculated from Equation 6, $min(r)$ the minimal stipple radius in use, and $max(r)$ the maximal stipple radius in use, respectively.

$$d_{(x,y)} = (\sum_{i=1}^{k} \hat{I}^{T_i})/k \tag{6}$$

$$r_{(x,y)} = \frac{d_{(x,y)} - min(d)}{max(d) - min(d)} \cdot (max(r) - min(r)) + min(r) \tag{7}$$

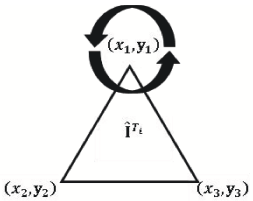
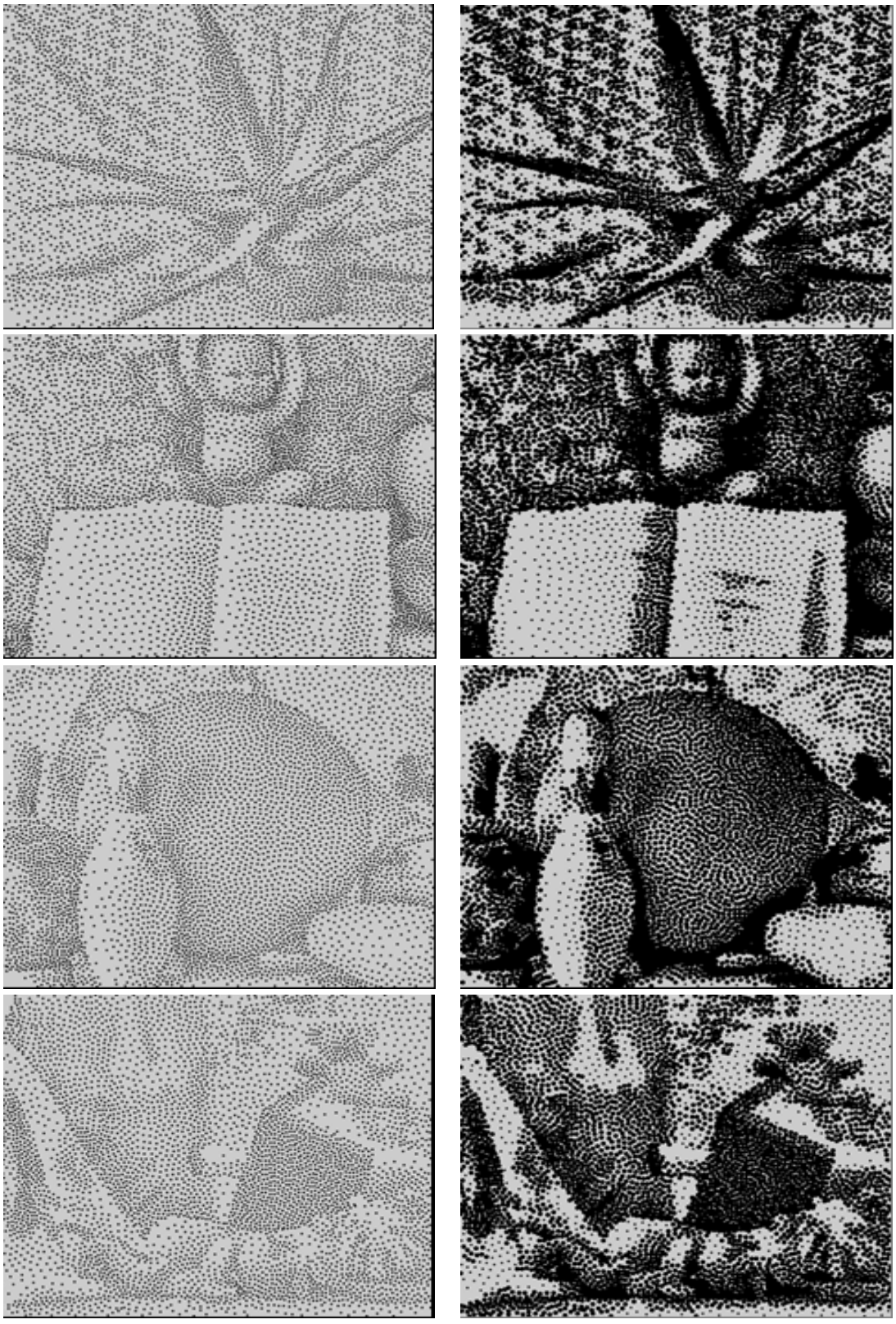


Fig. 9: Estimating the stipple size using the corresponding triangle mesh.

Some stippling results generated this way are shown in Figure 10. Note that the left column contains the results where the stipples are of the same size, while the right column the results of stipples with different sizes. As can be seen from these figures, the *simple* style, shown in Figure 10(a), has the property of being *simple* and *clean*, while the *hybrid* style, shown in Figure 10(b), on the other hand, has the property of being *expressive* and *complicated*. It can be shown that more people are relatively fond of the *simple* style, as demonstrated in Figure 40 to be shown later in the Results and Evaluations Section.



(a) (b)

Fig. 10: (a): the stippling results where stipples are of the same size. (b): the stippling results with stipples of different sizes.

3.4 Stereoscopic Stippling

3.4.1 Disparity-Map Processing

Once we can generate a stippling result, the next step is to extend the whole process to a *stereoscopic stippling*. Figure 11 shows the flow with which we can turn a stereoscopic image pair into a stereoscopic stippling result. The basic idea is to first transform the left image into its corresponding stippling image, and then map it to become the right stippling result through the left disparity map, while at the same time the right image can also be transformed to its right stippling image and mapped to the left stippling result through the right disparity map. There are definitely some inconsistency due to the two independent processes, and therefore some correlated *merging* is required. Furthermore, such a stippling process is in fact an iterative process, and it means several iterations are needed before a satisfactory stippling result can be derived.

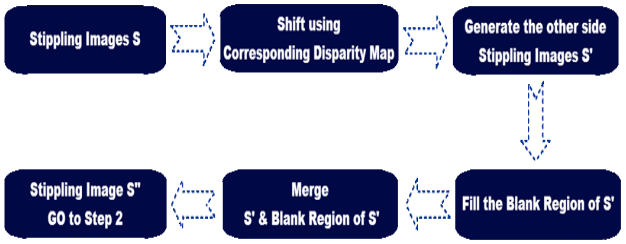
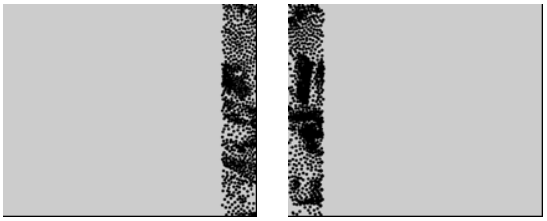


Fig. 11: The work flow for correlating the disparity maps.

More specifically, there are three core steps involved: *shift*, *fill* and *merge with constraint*, to be described in details in turn. Regarding the first step, namely *shift*, is to use the left (right) disparity map to map the left (right) stippling result to become the right (left) stippling result, as described in Equation 8. However, it is possible that after such a shift, some stippling image portion will be out of the boundary, and therefore will be cropped. Figure 12 shows the results after the corresponding shifting, where Figure 12(a), and Figure 12(c) are the original left and right stippling images, while Figure 12(b) and Figure 12(d) are the corresponding results after the shifting.

$$\begin{cases} x' = x \mp disp_R^L(x,y) \\ \\ y' = y \end{cases}, \quad 1 \le x' \le width \tag{8}$$

As can be easily noticed in Figure 12, the corresponding shifts will unavoidably leave blank areas, and thus the necessity of the second step: to *fill* the resulting blank areas. We first adopt the random sampling scheme as mentioned in Section 3.3 to deploy sample points on the corresponding blank areas shown in Figure 12(b) and Figure 12(d), by taking into account the image information along the boundary of the original right and left images, and obtain the results shown in Figure 13. As a result, these filled area may not align well with the existing content. And this is where the next step of *merging with constraint* comes in.



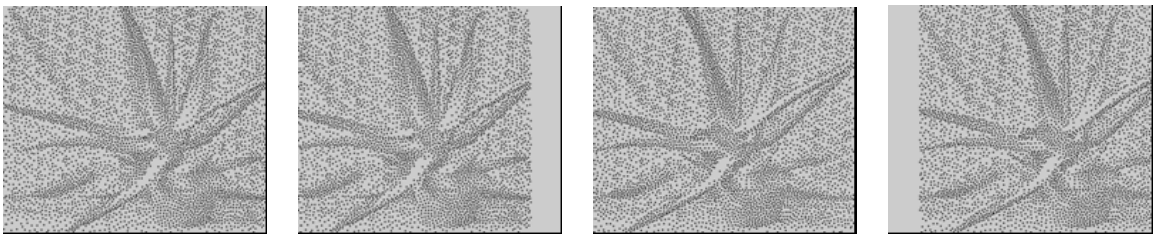
(a) (b)

Fig. 13: Filling of the stippling images. (a): Left image. (b): Right Image.

To merge the results from Figure 12(b), (d) with the results from Figure 13, there are two constraints to consider in order to maintain a proper correlation between the different angles associated with the left and right images. The first one is to require minimal amount of *vertical shift*, and the second one is to require the minimal amount of *content change*. To achieve this, we need to calculate the image information, as described in Equation 5 and Equation 6, under each triangle in the Delauney triangulation so that each point can be deployed to a proper position. Now assume that a stipple at (x, y), after the above calculation, should be relocated to (x', y'), define ∇v to be $|y - y'|$, and $\nabla \hat{I}$ to be $|\hat{I} - \hat{I}'|$, then the final position after the adjustment, is:

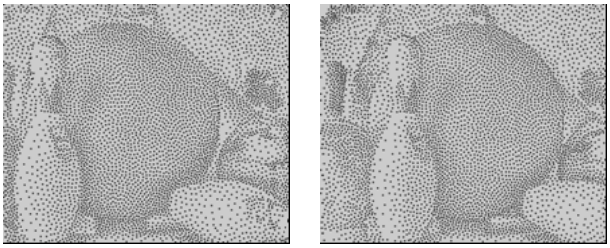
$$\begin{aligned} &(1-\varphi)(x,y) + \varphi(x',y') \\ &if \quad |y-y'| > \nabla v \quad and \quad |\hat{I}(x,y) - \hat{I}(x',y')| > \nabla \hat{I} \end{aligned} \tag{9}$$

where φ is a weighting factor used to adjust the proportions associated with the positions of (x, y) and (x', y'). In our current implementation, φ is set to be 0.2, meaning it respects more to the original position, while ∇v is 5 pixels, and $\nabla \hat{I}$ is set to be 0.2, respectively. Note that the value of ϕ is chosen this way is to strike a balance between *stereoscopic effect* and *image content*. That is, if the value is closer to 0, the point tends not to move, resulting a better stereoscopic effect; however, the image content may become affected, e.g., the boundary of an object could appear more rugged than it should be. The corresponding results after this merging step are shown in Figure 14.



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

Fig. 12: The stippling images after shifting.



(a) (b)

Fig. 14: The final stippling images. (a): Left stippling image. (b): Right stippling image.

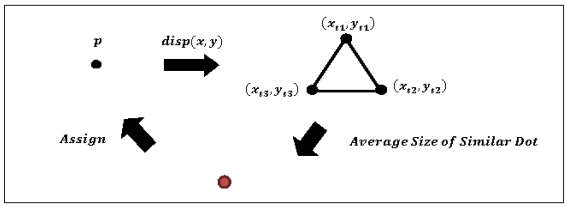


Fig. 15: The work flow of mutual check mechanism.

3.4.2 Mutual Checking Mechanism

Note that in our image stippling, there are two styles, as shown in Figure 10, i.e., the left column corresponding to the style where all the stipple are of the same size, denoted as the *simple* style hereafter, and the right column the style where the stipples could be of various sizes, denoted as the *hybrid* style hereafter. Although we have tried to correlate the left and right stippling results in the last step, some problems may still remain, such as different stipple sizes from the left and right views, as sometimes can be seen in the *hybrid* style. As a result, it could cause some undesired overlappings or flickering. To solve this issue, we propose a *mutual checking mechanism* that can help reduce this undesired effect.

$$dotSize(x,y) = \frac{\sum_{1\le i\le 3} dotSize(x_{t_i}, y_{t_i})}{n}, \quad if \;\; |\hat{I}(x_{t_i}, y_{t_i}) - \hat{I}(x,y)| < \epsilon \tag{10}$$

The basic idea is simple: we check the stipple size difference between a stipple and that of its mapped counterpart through the disparity map, and adjust the size accordingly if the difference is too much, as shown in Figure 15. However, as it is not necessary that the mapped position contains a stipple, some approximation should be done. More specifically, assume that there is a stipple (dot) p at position (x, y), and according to Equation 8, after the disparity mapping it gets mapped to (x', y'), we could then find the triangle, as shown in Figure 15 in the Delauney triangulation that contains the (x', y'). Further assume there are n stipples within the triangle that are similar to stipple p, in terms of image information as mentioned before, and then we calculate the average $dotSize(x, y)$ as in Equation 10. The size of p and $dotSize(x, y)$ are compared to determine if the size of p should be adjusted. However as there could exist occlusions during the correspondence calculation, a special care should be taken, as to be described next.

Assume there is a pixel p, whose corresponding pixel, through the disparity map, is pixel q, and if q is occluded, the stipple size at p cannot reference what happens at q; otherwise, they can reference each other. The way they reference each other, based on the image information, goes like the following. If p's image information value is larger than that of q, then we take the size of the stipple at p; otherwise, if q is occluded, then the size is set to be the average size of the stipple sizes at p and q. Finally, if q is not occluded, then the size is set to be the stipple size at q. Please refer to Figure 16 for the pseudo-code for more details.

4 Results and Evaluations

4.1 System Setup

We have developed our system on an Intel Core (TM) i7-3770 3.40 GHz machine with 4 GByte memory running on the Windows 7 64 bit operating system, and

```
if ∇ s < δ
   s_p = s_p_init
else
{
 if  d_p ≥ d_q
     s_p = s_q = s_p_init
 else
 {
  if q is occluded
     s_p = s_q = 0.5 s_p_init + 0.5 s_q
  else
     s_p = s_q
 }
}

where
p: a point
q: a corresponding point
s_p: the dot size of p
s_p_init: the initial dot size of p
s_q: the dot size of q
∇s: the difference of size between dots
δ: a threshold value of p
d_p: the image information of p
d_q: the image information of q
```

Fig. 16: The pseudo code for computing the dot size under different situations.

using Matlab, while using Java for the related tests to be shown in this section.

4.2 Stereoscopic Stippling Results

Figure 17 and Figure 18 show the stereoscopic stippling results of both *simple* style and *hybrid* styles using the algorithms we have described so far.

4.3 Evaluations

$$confidence_{value} = 1 - (\sum_{i=1}^{6}(N_1^i - N_2^i))/6 \qquad (11)$$

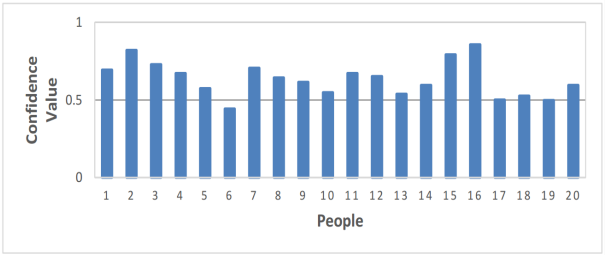


Fig. 19: Confidence value.

To evaluate our proposed system, we have conducted 1 test and 5 user studies, totally 6 experiments, to be described one by one, and there are totally 20 persons involved, and these persons are all students from the same department (information management), with their ages ranging from 20 to 30. To improve the accuracy, each test is repeated twice, with each person taking a break for 5 to 10 minutes before being tested again. Let N_j^i be the jth testing result for the ith experiment, then according to Equation 11, the *confidence value*, denoted as $confidence_{value}$ in Equation 11, can be defined, and it is evident that the higher the value is, the more confidence we have towards the consistency of all these experiments. Figure 19 shows the graph of the confidence values for all 20 persons for all the experiments.

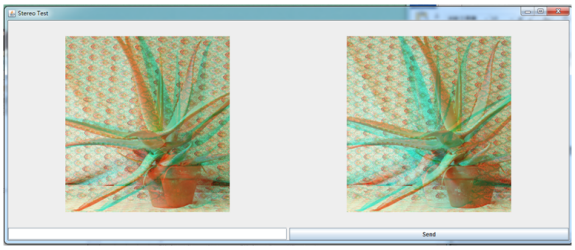


Fig. 20: The interface of the stereoscopic test.

The first experiment we have conducted is the *stereoscopic test*. The reason we want to perform this test is to evaluate the persons involved in our ensuing user studies so that we could better explain the results collected afterwards if necessary, in other words, each person can then be associated with a confidence value. There are 7 set of stereoscopic images being used and we try to adjust the associated disparity value to alter the perceived depth. For our testing, we present one image pair with the original disparity setting, while the other one with the disparity being shrinked to 1/4 of its original value. Since less disparity value corresponds to less depth, we ask each person, through an interface shown in Figure 20 with a random image order, to determine which one is farther from the person by clicking on the bottom below the corresponding stereoscopic image pair. Figure 21 and Figure 22 show the stereoscopic images used in the test, where Figure 21(b) and Figure 22(b) are the ones that are closer from the viewer while Figure 21(a) and Figure 22(a) are the ones that are farther from the viewer.

Figure 23 shows the stereoscopic feeling of each person for each stereoscopic image pair in terms of numbers, also represented as the vertical indices in the figure, where *Can't Feel* means the the participants got the wrong answer, *Same* means the participants felt that

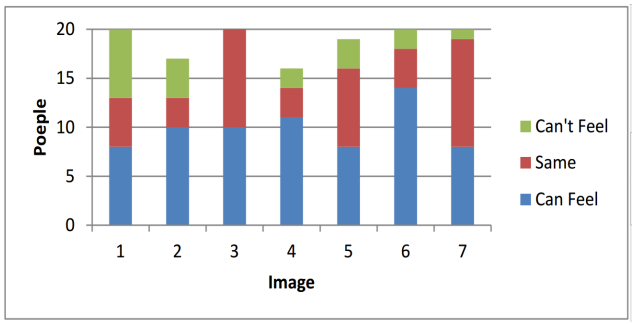


Fig. 23: Stereoscopic feeling for images/people.

the two images are non-distinguishable (both stereoscopic image pairs are of the same depth), and *Can Feel* means the participants got the correct answer. As can be seen from the figure, most people get the right answers for most of the testing images.

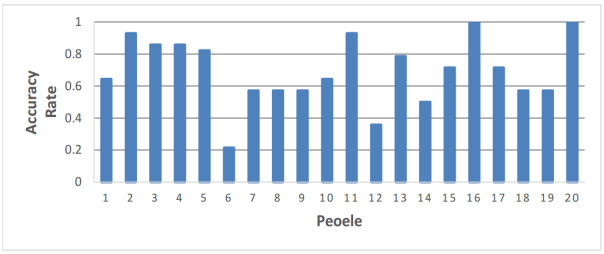


Fig. 24: Stereoscopic accuracy for people.

In terms of accuracy, we show the results in terms of numbers in Figure 24, where the the horizontal axis lists the person index. Overall, 80% of the persons can have accuracy rate about 0.6, and it is obvious that the accuracy could be even higher if person 6,12 and 14 who perform badly in the stereoscopic test in general, are excluded from the test.

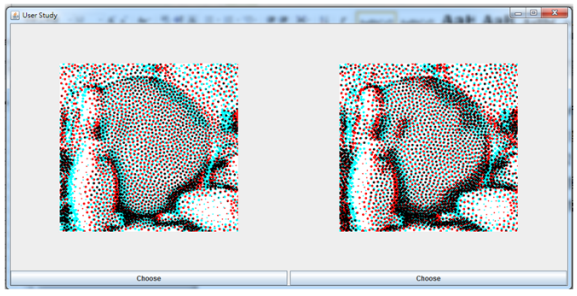


Fig. 25: The interface of the first user study.

Once we have evaluated the stereoscopic capability of those persons in the test, we conduct 5 user studies to further evaluate our system. The first one is to compare the stereoscopic stippling results, generated by performing image stippling *independently* on both the left and right images, referred to as the *Direct* approach, with the ones generated using our proposed system, referred to as the *Ours* approach, where some coordination between the left and right images has been done. Figure 25 shows the interface used for this test.

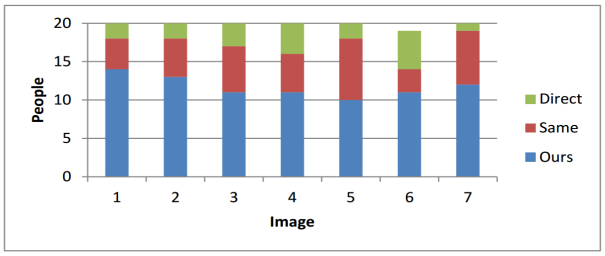


Fig. 26: The comparisons in the first user study (Direct v.s. Ours) for all images/persons.

Figure 26 shows the numerical results of this user study, where the vertical axis represents the choice of a person: *Direct* means he/she prefers the *Direct* approach, and *Ours* means he/she prefers the *Ours* approach, while *Same* means he/she cannot distinguish which one is better. The horizontal axis lists the image index in the test while the vertical axis shows the *stacked bar graph*. As can be seen from this figure, most persons choose *Ours* rather than the *Direct* approach.

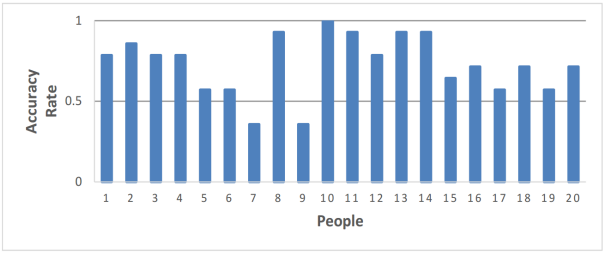


Fig. 27: The accuracy of the first user study (Direct v.s. Ours) for all persons.

For convenience, if we just deem choosing *Ours* as being correct, Figure 27 shows the accuracy results for all persons by averaging all of them over all 7 testing stereoscopic images. It can be calculated that 90% of the persons show at least 0.6 accuracy ratio.

Figure 28 and Figure 29 show all the testing images used in this user study. Figure 28(a) and Figure 29(a) are the results from the *Direct* approach, while Figure 28(b) and Figure 29(b) are the results from the *Ours* approach.

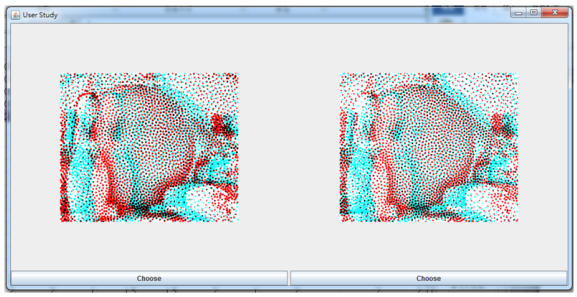


Fig. 30: The interface of the second user study.

Note that for the first user study, all the stippling image results are drawn by using Matlab; however, through numerous experiments it seems Matlab does not have a *smooth size increment* on the drawing of stipples (dots) with various sizes. As a result, we conduct the second user study, the same experiment except by replacing the results drawn from Matlab with the ones drawn by using *OpenGL*. Figure 30 shows the interface we used for this second user study, where the images are randomly shuffled.

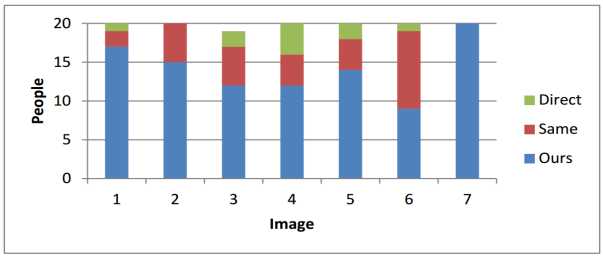


Fig. 31: The comparisons in the second user study (Direct v.s. Ours in OpenGL) for all images/persons.

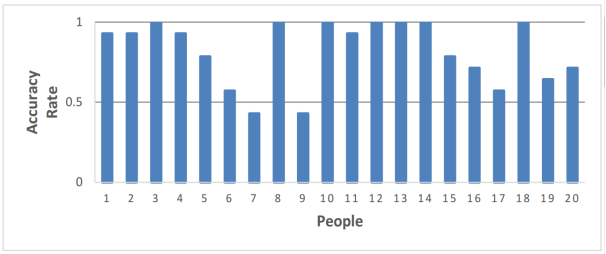


Fig. 32: The accuracy ratio in the second user study (Direct v.s. Ours in OpenGL) for all persons.

As can be seen from Figure 31, and Figure 32, the results are slightly better. In fact, as can be calculated from Figure 32, 90% of the persons are with at least 0.6 accuracy. Through observation, the results generated from OpenGL indeed show more smooth size variation, thus leading to slightly better quality.

Similarly, Figure 33 and Figure 34 show all the images used for the second user study, where Figure 33(a) and Figure 34(a) are the ones generated from the *Direct* approach, while Figure 33(b) and Figure 34(b) are the ones generated from the *Ours* approach.

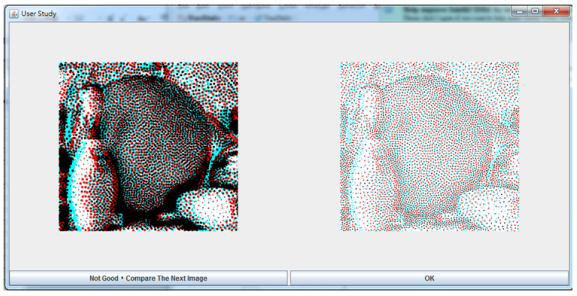


Fig. 35: The interface of the third user study.

For the third user study, we want to compare how persons perceive between the two stippling styles as mentioned before, i.e., *simple* versus *hybrid*, where the former one refers to the stippling style with all stipples of the same size, while the latter one the stippling style where the stipples may be of different sizes. Figure 35 shows the interface that we use for this user study. To make it easier, we always display a stippling with the *hybrid* style on the left, and a stippling with the *simple* style on the right, with the initial setting of 2500 sample points. A user can compare both stereoscopic stippling images and select the one which he/she thinks is better by clicking the button below the image. In the case when the user thinks the *hybrid* style is better, our system will increase the number of sample points on the right by adding 500 more sample points, until the *simple* style is at least not worse than the *hybrid* one on the left.

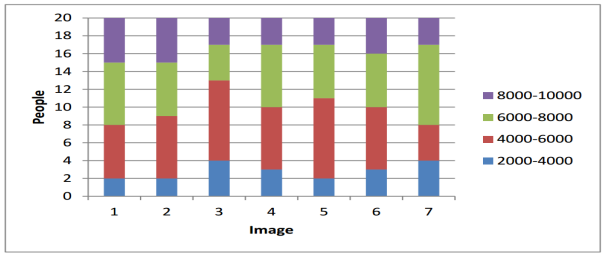


Fig. 36: The comparisons in the third user study for all images/persons.

Figure 36 shows the numerical results of the third user study, where the vertical axis shows the number of

sample points required for a user to feel that the *simple* style's stippling is not worse than that of the *hybrid* style, the horizontal axis is the stereoscopic image index. It can be shown that about 60% of the persons feel that it takes the *simple* style at least 1000 more of sample points for being able to be compared with the fixed *hybrid* style. Notice that there still some persons feel that even with less number of sample points, the *simple* style can still be better than the *hybrid* style, and after further discussions with those persons we learn the fact that some people just consistently think a *simple* stippling style shows the results in a more elegant and unmessy way.

Figure 37 and Figure 38 show some of the stereoscopic images for both styles that we use for this user study, where column (a) are the results of the *hybrid* style with 5000 sample points, column (b) the results of the *simple* style with 5000 sample points, and column (c) the results of the *simple* style with 7000 sample points.

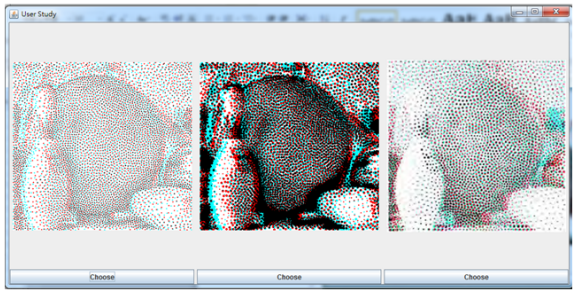


Fig. 39: The interface of the fourth user study.

The fourth user study is to compare the results from our system, both *simple* and *hybrid* styles, with the one generated from work by Northam et al. [17]. Figure 39 shows the interface for this user study, where a user can choose the best of the three displayed stereoscopic stippling results by clicking the associated button below. Note that to have a fair comparison, the corresponding images are randomly shuffled in this user study.

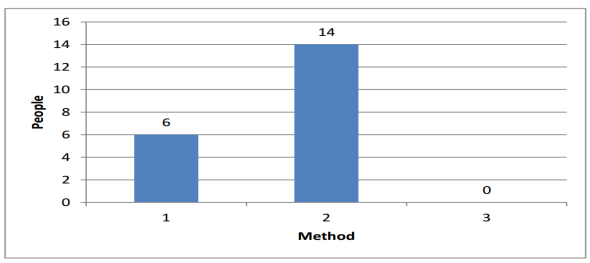


Fig. 40: The comparisons in the fourth user study for all the persons.

Figure 40 shows the numerical result of this user study, where the vertical axis represents the three methods selected, i.e., 3 means Northam et al. [17]'s method being selected, 2 means the *simple* style of ours being selected, and 1 means the *hybrid* style of ours being selected. The horizontal axis represents the person index. As can be seen from this figure, all the persons think at least one of ours, either *simple* or *hybrid*, is better than the one proposed by Northam et al. [17]. Also notice that most people like the *simple* style more than the *hybrid* style for its simplicity, as was also mentioned in another user study.

Figure 41 shows one of the testing image sets for this user study, where (a) is the result of *simple* style with 5000 sample points, (b) the result of *hybrid* style with 5000 sample points, and (c) the result from Northam et al. [17] with a comparable sample point setting. Notice how our proposed approaches help better preserve the objects' boundary in this comparison.

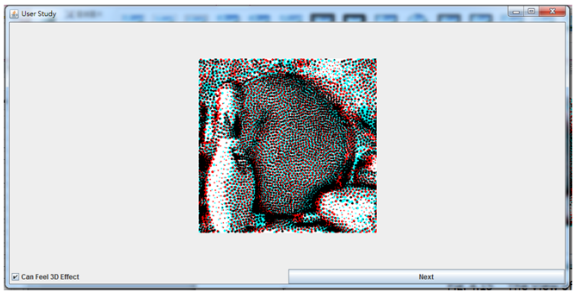


Fig. 42: The interface of the fifth user study.

The fifth and the last user study is to test the quality of stippling result if the *SAD* approach is used to construct a relatively rough disparity map, as described in Section 3.2.1. Figure 42 shows the interface used in this user study.

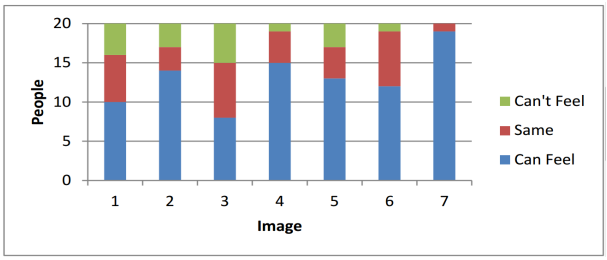


Fig. 43: The comparisons in the fifth user study for all images/persons.

Figure 43 shows the numerical result of this user study, where the vertical axis represent the number of people bearing a *stereoscopic feeling* for a particular

stereoscopic stippling image. Image indices are represented as the horizontal axis. It can be seen that most people still have the stereoscopic feeling even when the disparity maps are not constructed very well.

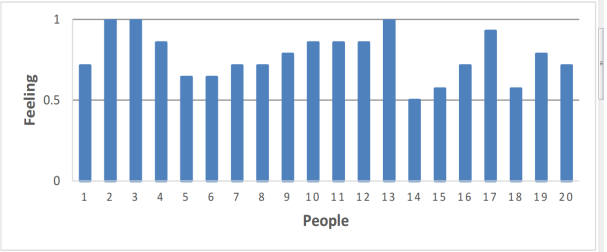


Fig. 44: The average feeling in the fifth user study for all persons.

Figure 44 shows the corresponding results by averaging the feeling from all images for each person, and the vertical axis is normalized to be from 0 to 1 for simplicity. It can be shown that about 90% of the persons can still feel the desired stereoscopic effects for the disparity maps constructed this way.

Figure 45 shows the stereoscopic stippling images in the *hybrid* style used in this user study. In terms of running time, it normally takes less than two minutes to generate a result image with the aforementioned testing environment.

5 Conclusions

Given a stereoscopic image pair with its corresponding disparity maps, our proposed system can turn it into a stereoscopic stippling image pair. Rather than converting the left and right images independently into stippling images, some constraints and *mutual check mechanism* are added to improve the *binocular rivalry* problem often experienced in stereoscopic image processing, thus preserving the stereoscopic feeling during the stippling process. Compared with some existing work, our proposed system shows clear improvement, which is proved by the user study that we conduct in this work. We have also shown that, even without the provision of disparity maps, the generated results can still carry the desired stereoscopic feeling by using the disparity maps roughly constructed through a *SAD* comparison scheme.

However, there is still limitation in this work. For the *hybrid* stippling style that we propose, the difference in stipple size may cause the results to look messy. As a result, a small portion of the persons in the user study think that our results are not better than the ones generated by directly and independently converting the left and right input images into the left and right stippling images.

In the future, not only we would like to make improvement on the aforementioned issue, but also like to seek the possibility of reducing the input to be just a single non-stereoscopic image, while still being able to produce a desired stereoscopic stippling image pair. Another interesting direction worth further studying is the generalization of this work to video.

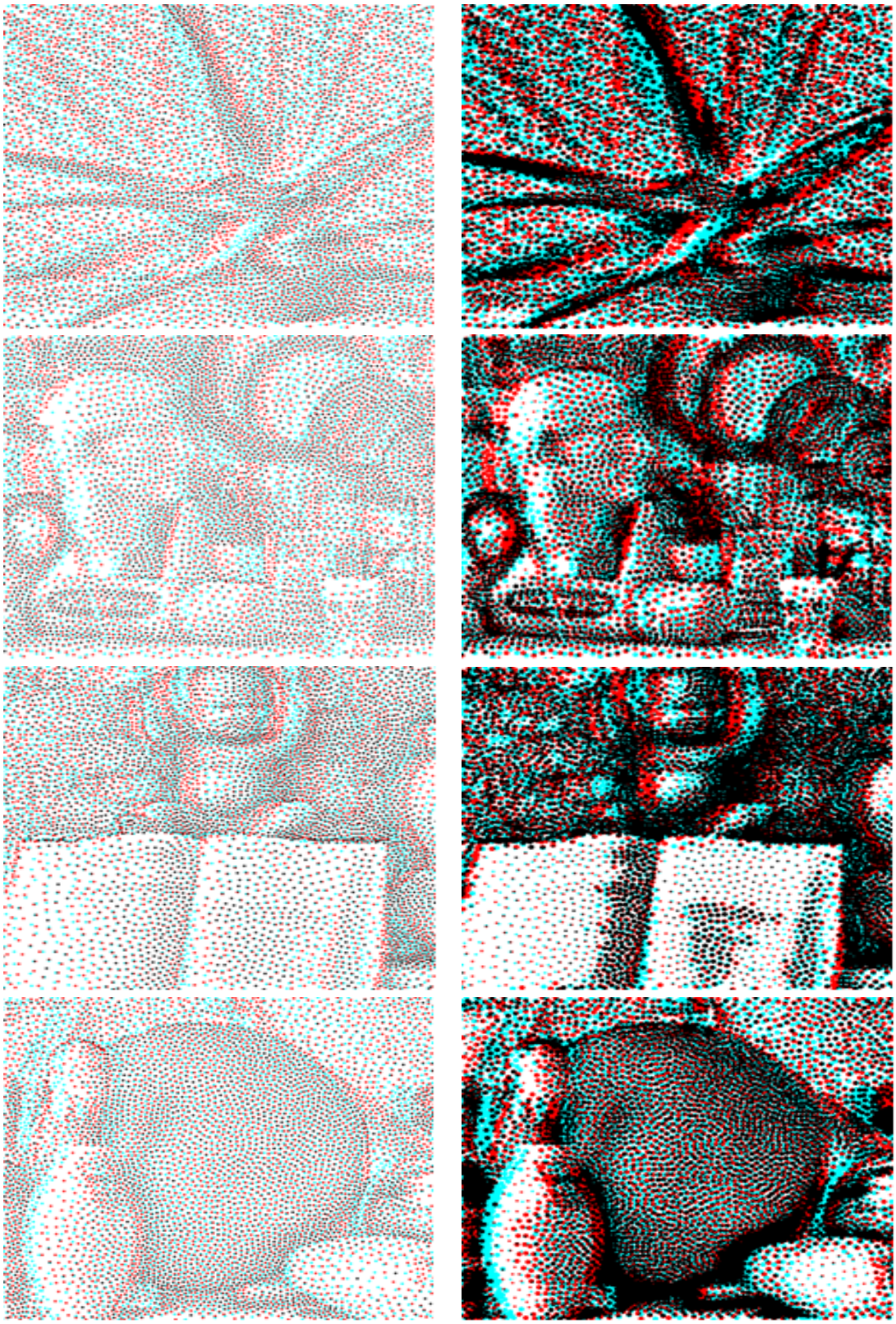
Acknowledgment

This work was supported in part by the National Science Council of Taiwan under the grant NSC 101-2221-E-011-150-MY3 and MOST 104-2218-E-001-002.

References

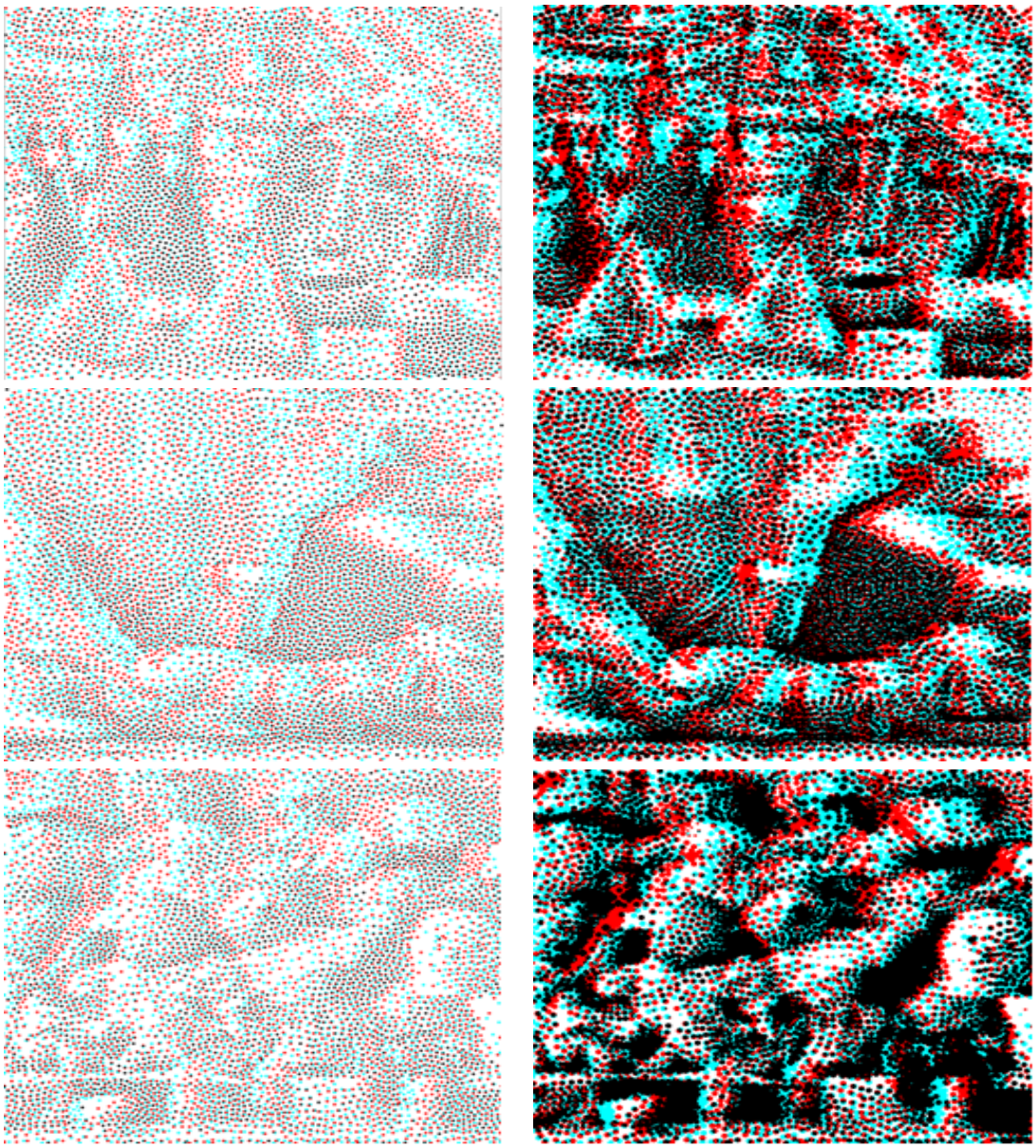
1. EDISON. http://coewww.rutgers.edu/riul/research/code/EDISON/doc/help.html
2. Middlebury Stereo Datasets. http://vision.middlebury.edu/stereo/data/
3. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In: ICCV '2001, pp. 105–112 (2001)
4. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**, 603–619 (2002)
5. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. Int. J. Comput. Vision **70**(1), 41–54 (2006)
6. Hertzmann, A.: Painterly rendering with curved brush strokes of multiple sizes. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '98, pp. 453–460. ACM, New York, NY, USA (1998)
7. Hosni, A., Rhemann, C., Bleyer, M., Rother, C., Gelautz, M.: Fast cost-volume filtering for visual correspondence and beyond. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(2), 504–511 (2013)
8. Klaus, A., Sormann, M., Karner, K.: Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In: Proceedings of the 18th International Conference on Pattern Recognition - Volume 03, ICPR '06, pp. 15–18. IEEE Computer Society, Washington, DC, USA (2006)
9. Lee, K.Y., Chung, C.D., Chuang, Y.Y.: Scene warping: Layer-based stereoscopic image resizing. In: CVPR'12, pp. 49–56 (2012)
10. Lo, W.Y., van Baar, J., Knaus, C., Zwicker, M., Gross, M.H.: Stereoscopic 3d copy & paste. ACM Trans. Graph. **29**(6), 147 (2010)
11. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004)
12. Luo, S.J., Shen, I.C., Chen, B.Y., Cheng, W.H., Chuang, Y.Y.: Perspective-aware warping for seamless stereoscopic image cloning. Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2012) **31**(6), 182:1–182:8 (2012)

13. McCool, M., Fiume, E.: Hierarchical poisson disk sampling distributions. In: Proceedings of the Conference on Graphics Interface '92, pp. 94–105. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1992)
14. Meer, P., Georgescu, B.: Edge detection with embedded confidence. IEEE Trans. Pattern Anal. Mach. Intell. **23**(12), 1351–1365 (2001)
15. Ning, J., Zhang, L., Zhang, D., Wu, C.: Interactive image segmentation by maximal similarity based region merging. Pattern Recogn. **43**(2), 445–456 (2010)
16. Niu, Y., Feng, W.C., Liu, F.: Enabling warping on stereoscopic images. ACM Trans. Graph. **31**(6), 183:1–183:7 (2012)
17. Northam, L., Asente, P., Kaplan, C.S.: Consistent stylization and painterly rendering of stereoscopic 3d images. In: Proceedings of the Symposium on Non-Photorealistic Animation and Rendering, NPAR '12, pp. 47–56. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2012)
18. Pears, N., Liu, Y., Bunting, P.: 3D Imaging, Analysis and Applications. Springer (2012)
19. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. In: ACM SIGGRAPH 2003 Papers, SIGGRAPH '03, pp. 313–318. ACM, New York, NY, USA (2003)
20. Richardt, C., Świrski, L., Davies, I., Dodgson, N.A.: Predicting stereoscopic viewing comfort using a coherence-based computational model. In: D. Cunningham, T. Isenberg (eds.) Proceedings of Computational Aesthetics (CAe) (2011)
21. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision **47**, 7–42 (2002)
22. Secord, A.: Weighted Voronoi stippling. In: Proceedings of the second international symposium on Non-photorealistic animation and rendering, pp. 37–43. ACM Press (2002)
23. Smith, B.M., 0003, L.Z., Jin, H.: Stereo matching with nonparametric smoothness priors in feature space. In: CVPR, pp. 485–492. IEEE (2009)
24. Xu, Y., Liu, L., Gotsman, C., Gortler, S.J.: Smi 2011: Full paper: Capacity-constrained delaunay triangulation for point distributions. Comput. Graph. **35**(3), 510–516 (2011)



(a) (b)

Fig. 17: Stippling results. (a): anaglyph stippling images with the same point size. (b): anaglyph stippling images with different point sizes.

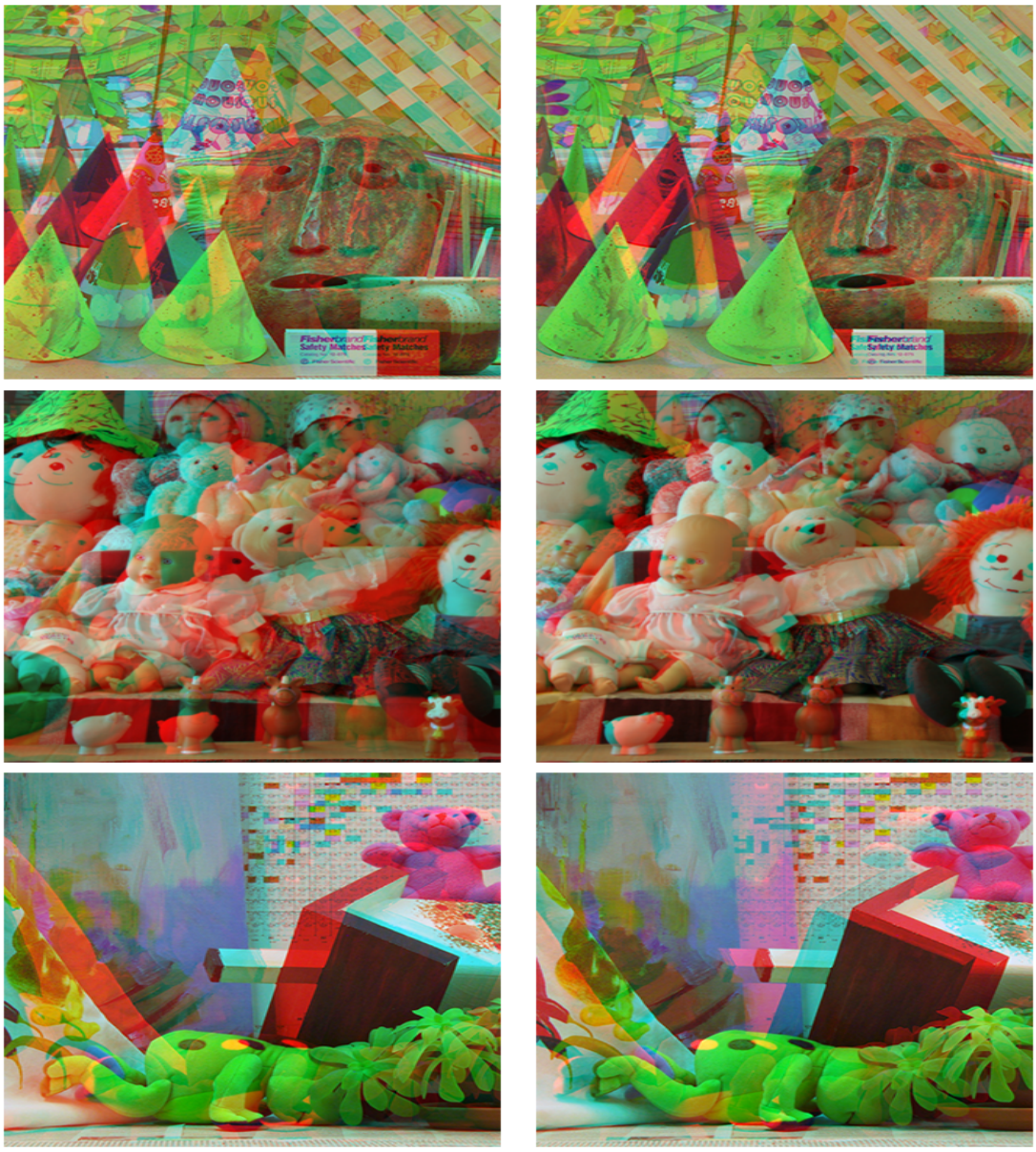


(a) (b)

Fig. 18: More stippling results. (a): anaglyph stippling images with the same point size. (b): anaglyph stippling images with different point sizes.

(a) (b)

Fig. 21: Stereo test examples. (a): Farther images. (b): Closer images.

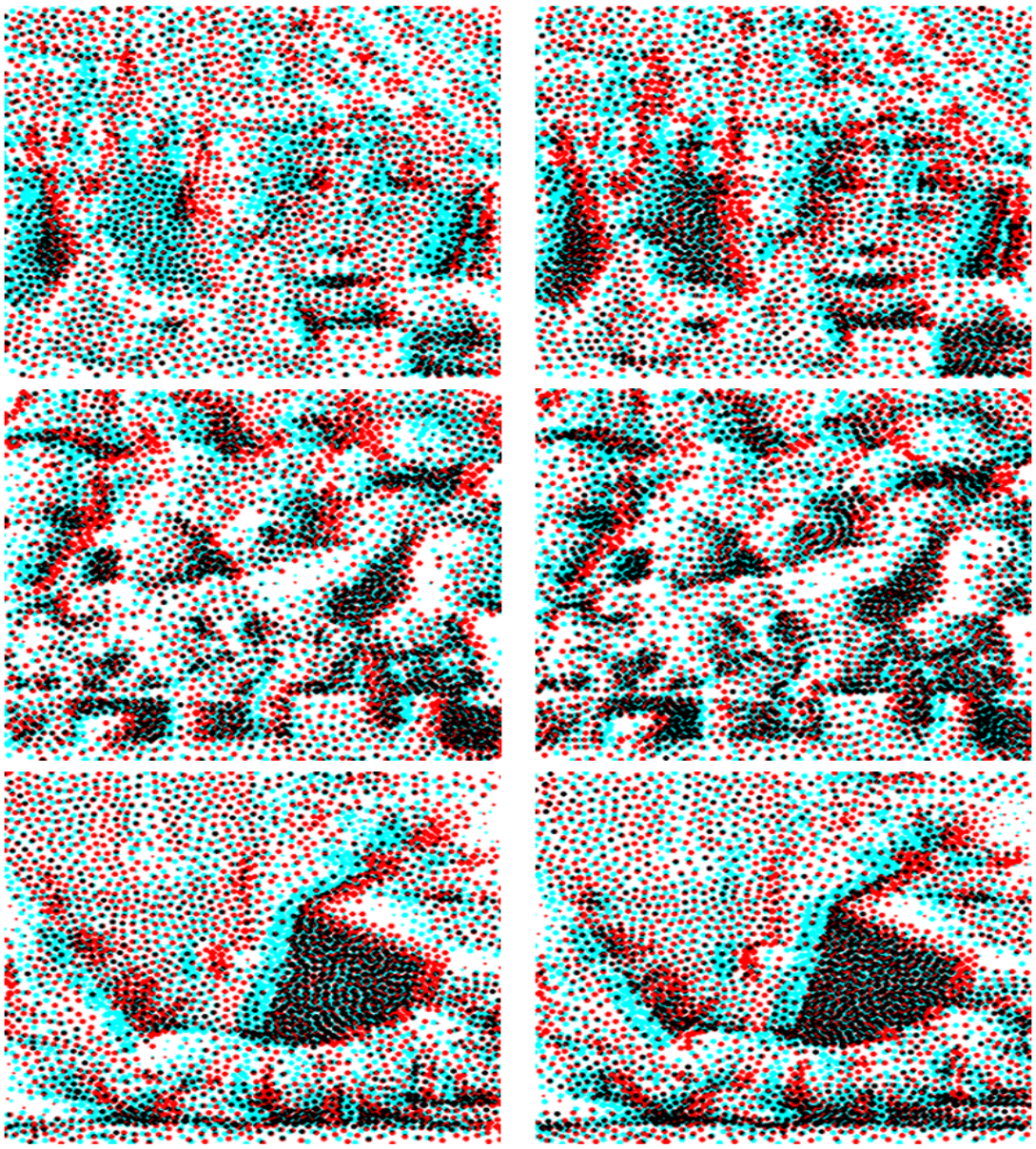


(a) (b)

Fig. 22: More stereo test examples. (a): Farther images. (b): Closer images.

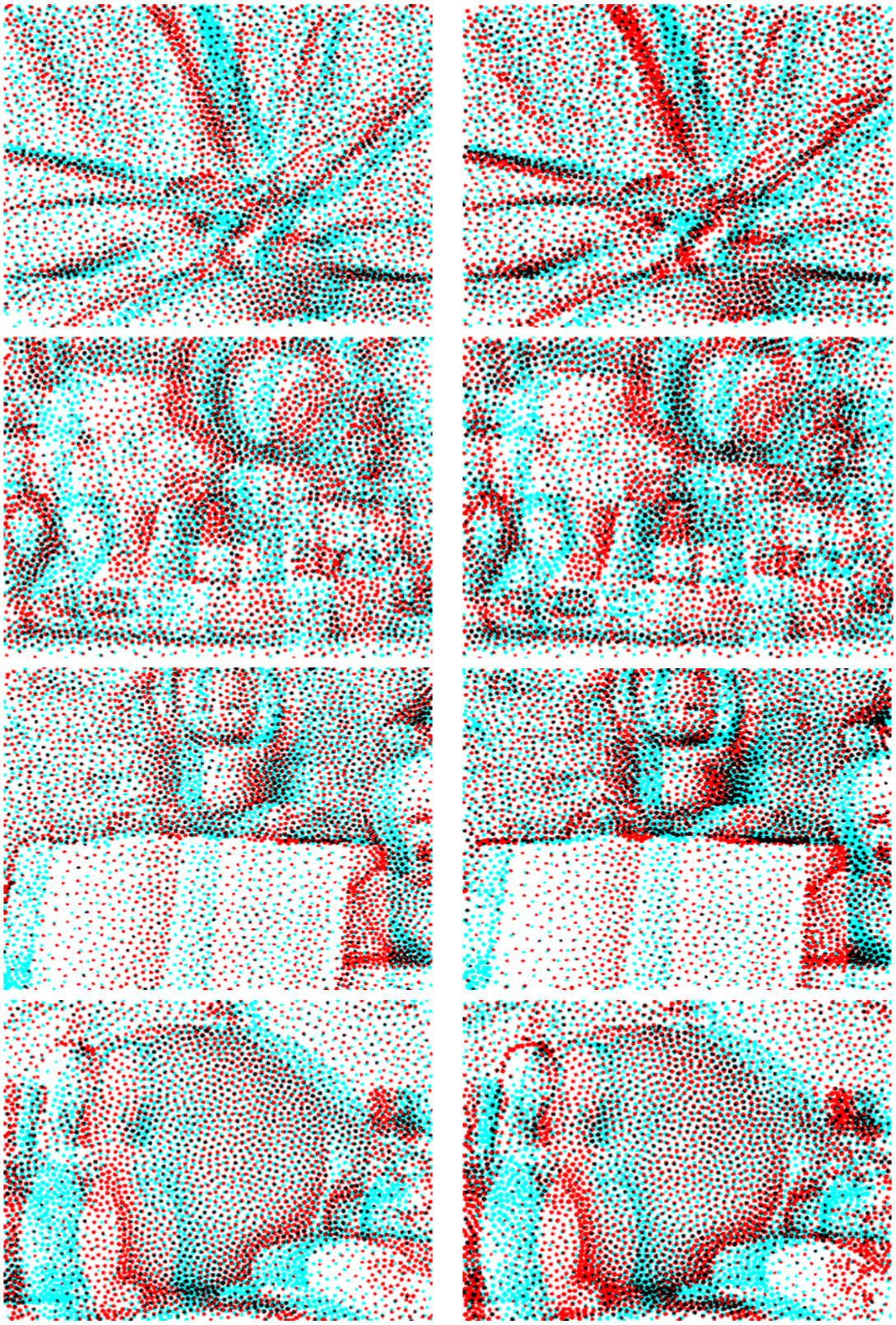
(a) (b)

Fig. 28: Images used for the first user study. (a): Direct. (b): Ours.



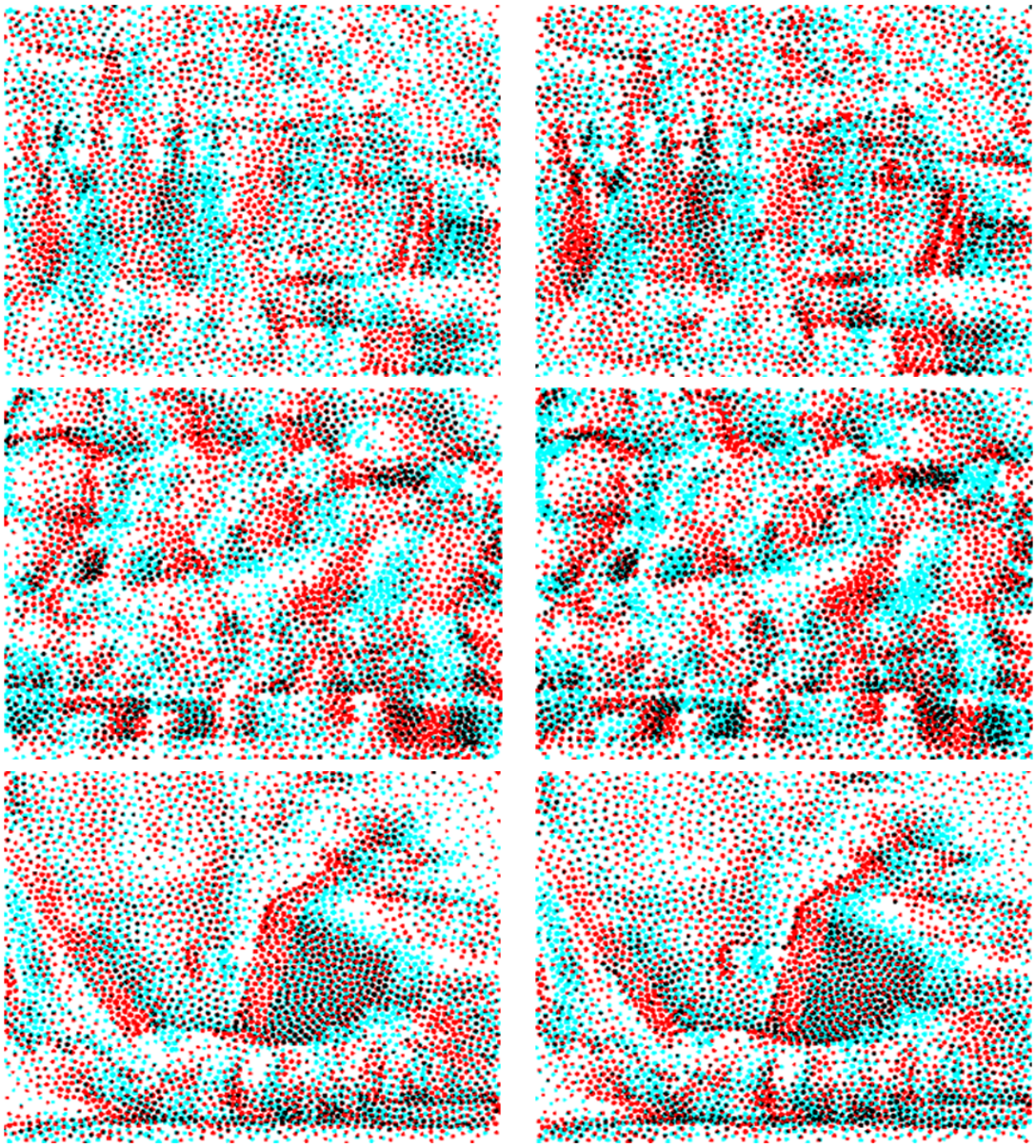
(a) (b)

Fig. 29: More images used for the first user study. (a): Direct. (b): Ours.



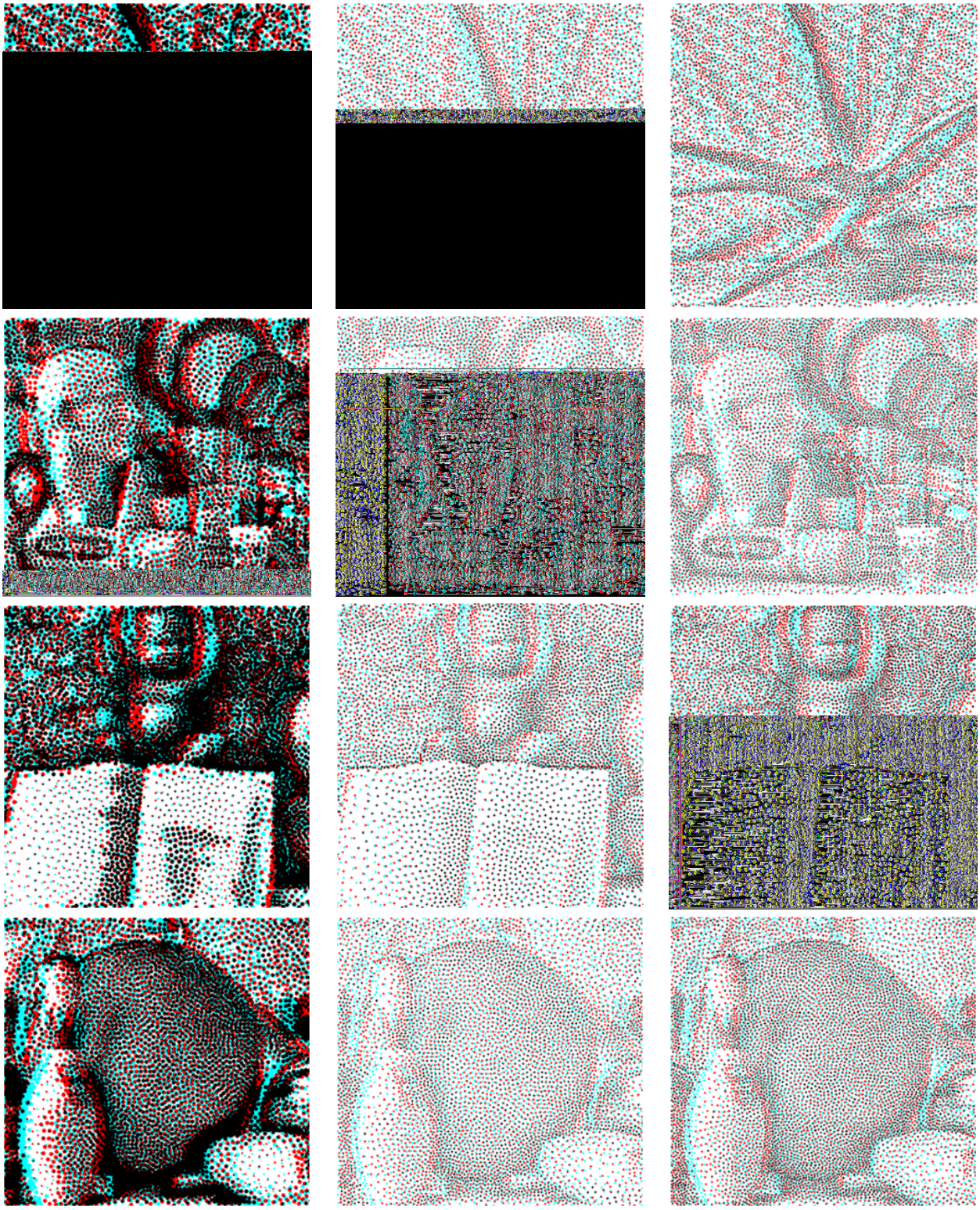
(a) (b)

Fig. 33: Images used for the second user study. (a): Direct. (b): Ours.



(a) (b)

Fig. 34: More images used for the second user study. (a): Direct. (b): Ours.



(a) (b) (c)

Fig. 37: Images used in the third user study. (a): Hybrid 5000. (b): Simple 5000. (c): Simple 7000.

(a) (b) (c)

Fig. 38: More images used in the third user study. (a): Hybrid 5000. (b): Simple 5000. (c): Simple 7000.

(a) (b) (c)

Fig. 41: The fourth user study. (a): Simple 5000. (b): Hybrid 5000. (c): The result from Northam et al. [17].

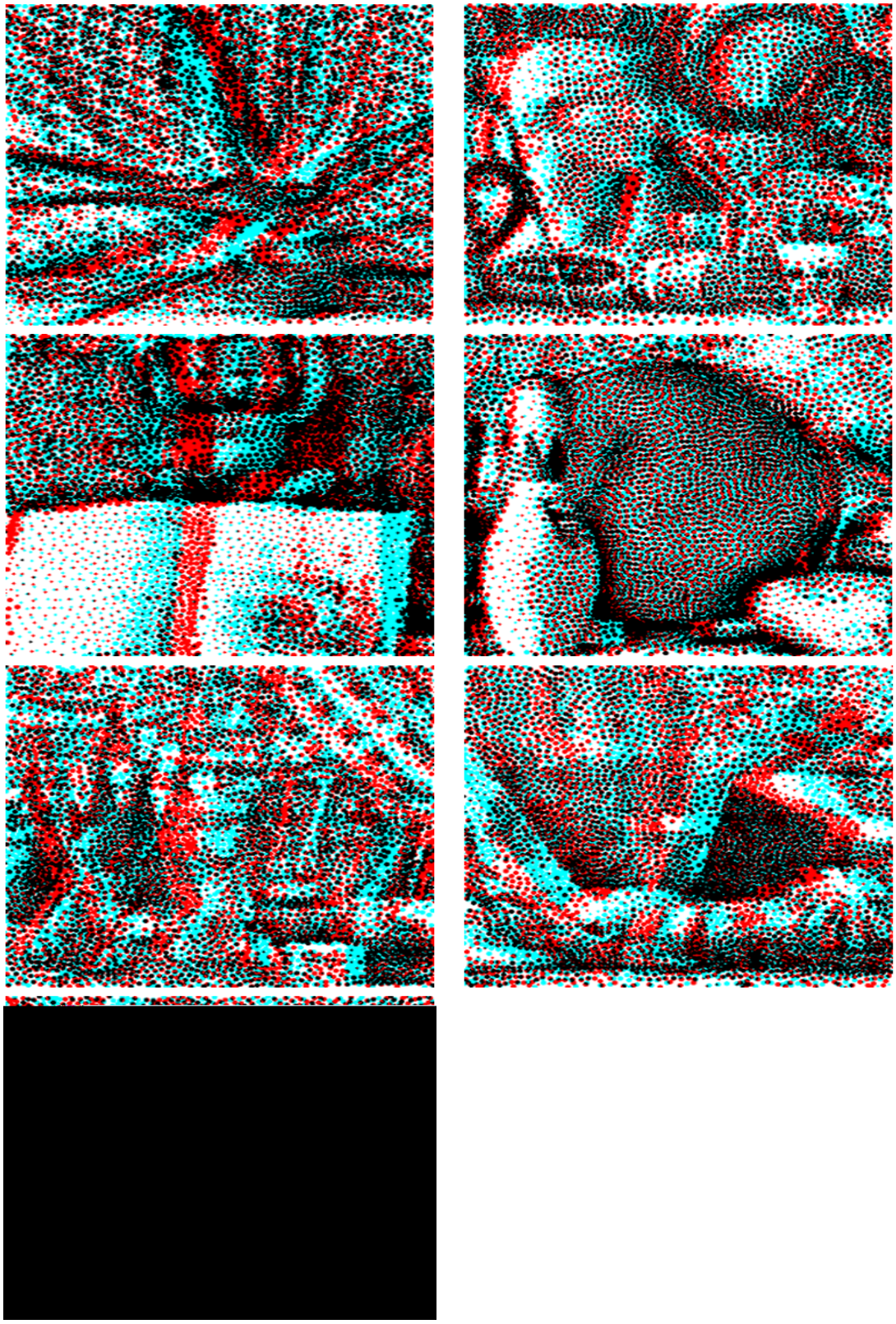


Fig. 45: The fifth user study.