Example-Based Stippling using a Scale-Dependent Grayscale Process

Domingo Martín* Germán Arroyo* M. Victoria Luzón* Tobias Isenberg†
* University of Granada, Spain † University of Groningen, The Netherlands

Abstract

We present an example-based approach to synthesizing stipple illustrations for static 2D images that produces scale-dependent results appropriate for an intended spatial output size and resolution. We show how treating stippling as a grayscale process allows us to both produce on-screen output and to achieve stipple merging at medium tonal ranges. At the same time we can also produce images with high spatial and low color resolution for print reproduction. In addition, we discuss how to incorporate high-level illustration considerations into the stippling process based on discussions with and observations of a stipple artist. The implementation of the technique is based on a fast method for distributing dots using halftoning and can be used to create stipple images interactively.

CR Categories: I.3.m [Computer Graphics]: Miscellaneous—Non-photorealistic rendering (NPR); I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

Keywords: Stippling, high-quality rendering, scale-dependent NPR, example-based techniques, illustrative visualization.

1 Introduction

Stippling is a traditional pen-and-ink technique that is popular for creating illustrations in many domains. It relies on placing stipples (small dots that are created with a pen) on a medium such as paper such that the dots represent shading, material, and structure of the depicted objects. Stippling is frequently used by professional illustrators for creating illustrations, for example, in archeology (e. g., see Fig. 2), entomology, ornithology, and botany.

One of the essential advantages of stippling that it shares with other pen-and-ink techniques is that it can be used in the common bilevel printing process. By treating the stipple dots as completely black marks on a white background they can easily be reproduced without loosing spatial precision due to halftoning artifacts. This property and the simplicity of the dot as the main rendering elements has lead to numerous approaches to synthesize stippling within NPR, describing techniques for distributing stipple dots such that they represent a given tone. Unfortunately, computer-generated stippling sometimes creates distributions with artifacts such as unwanted lines, needs a lot of computation power due to the involved computational complexity of the approaches, cannot re-create the merging of stipple dots in middle tonal ranges that characterizes many hand-drawn examples, or produces output with dense stipple points unlike those of hand-drawn illustrations.

To address these issues, our goal is to realize a stippling process for 2D images (see example result in Fig. 1) that is easy to implement, that considers the whole process of hand-made stippling, and that takes scale and output devices into account. For this purpose we do not consider stippling to always be a black-and-white technique, in contrast to previous NPR approaches (e. g., [Kim et al. 2009]). In fact, we use the grayscale properties of hand-made stipple illustrations to inform the design of a grayscale stipple process. This different approach lets us solve not only the stipple merging problem but also lets us create output adapted to the intended output device. To summarize, our paper makes the following contributions:

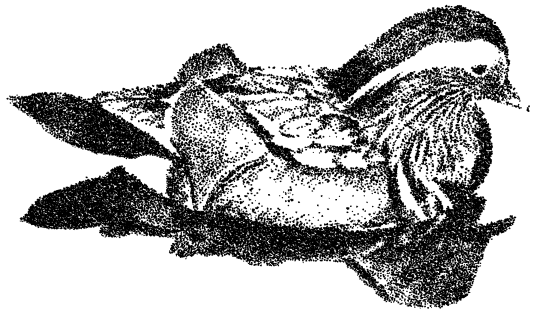


Figure 1: *An example stipple image created with our technique.*

- an analysis of high-level processes involved in stippling and a discussion on how to support these using image-processing,
- a method for example-based stippling in which the stipple dot placement is based on halftoning,
- the scale-dependent treatment of scanned stipple dot examples, desired output, and stipple placement,
- a grayscale stippling process that can faithfully reproduce the merging of stipples at middle tonal ranges,
- the enabling of both print output in black-and-white and on-screen display in gray or tonal scales at the appropriate spatial and color resolutions, and
- that the resulting technique is easy to implement and permits the interactive creation of stipple images.

The remainder of the paper is structured as follows. First we review related work in the context of computer-generated stippling in Section 2. Next, we analyze hand-made stippling in Section 3, both with respect to high-level processes performed by the illustrator and low-level properties of the stipple dots. Based on this analysis we describe our scale-dependent grayscale stippling process in Section 4 and discuss the results in Section 5. We conclude the paper and suggest some ideas for future work in Section 6.

2 Related Work

Pen-and-ink rendering and, specifically, computer-generated stippling are well-examined areas within non-photorealistic rendering (NPR). Many approaches exist to both replicating the appearance of hand-drawn stipple illustrations and using stippling within new contexts such as animation. In the following discussion we distinguish between stipple rendering based on 3D models such as boundary representations or volumetric data on the one side and stippling that uses 2D pixel images as input on the other.

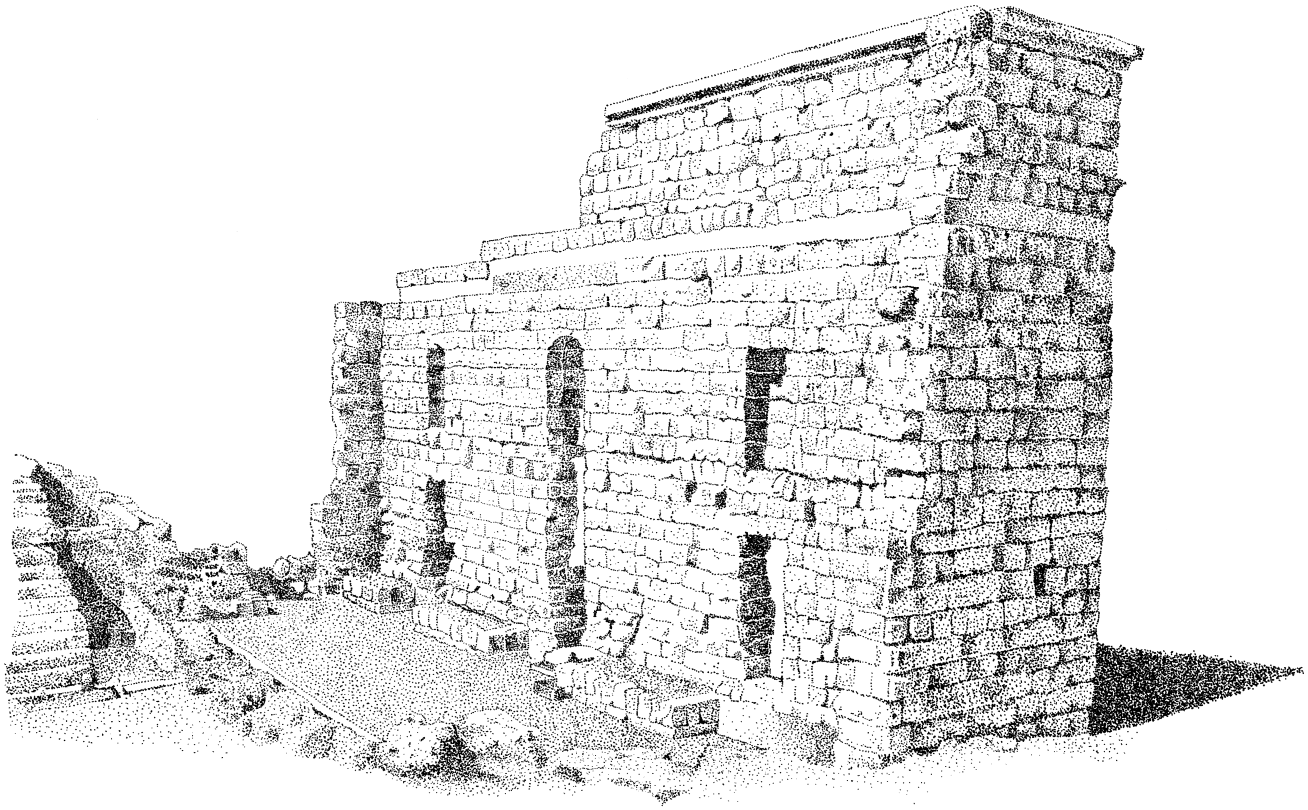


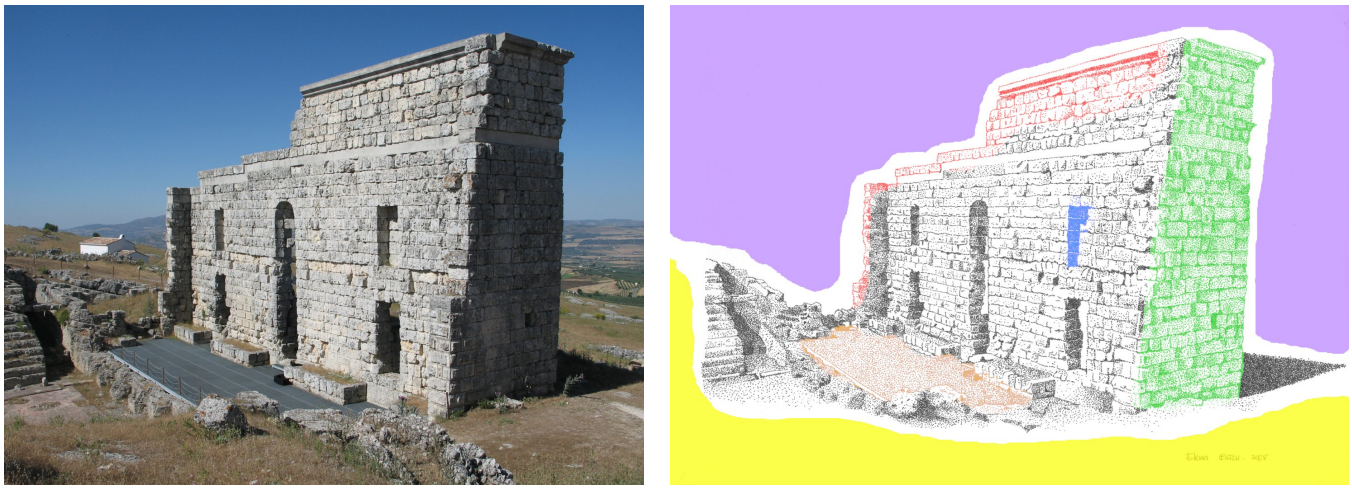
Figure 2: *Hand-drawn stipple image by illustrator Elena Piñar of the Roman theater of Acinipo in Ronda, Málaga (originally on A4 paper). This image is © 2009 Elena Piñar, used with permission.*

The variety of types of 3D models used in computer graphics is also reflected in the diversity of stipple rendering approaches designed for them. There exist techniques for stippling of volume data [Lu et al. 2002a; Lu et al. 2003] typically aimed at visualization applications, stipple methods for implicit surfaces [Foster et al. 2005; Schmidt et al. 2007], point-sampled surfaces [Xu and Chen 2004; Zakaria and Seidel 2004], and stippling of polygonal surfaces [Lu et al. 2002b; Sousa et al. 2003]. The placement of stipple dots in 3D space creates a unique challenge for these techniques because the viewer ultimately perceives the point distribution on the 2D plane. Related to this issue, animation of stippled surfaces [Meruvia Pastor et al. 2003; Vanderhaeghe et al. 2007] presents an additional challenge as the stipples have to properly move with the changing object surface to avoid the shower-door effect. A special case of stippling of 3D models is the computation in a geometry-image domain [Yuan et al. 2005] where the stippling is computed on a 2D geometry image onto which the 3D surface is mapped.

While Yuan et al. [2005] map the computed stipples back onto the 3D surface, many approaches compute stippling only on 2D pixel images. The challenge here is to achieve an evenly spaced distribution that also reflects the gray value of the image to be represented, a optimization problem within stroke-based rendering [Hertzmann 2003]. One way to achieve a desired distribution is Lloyd's method [Lloyd 1982; McCool and Fiume 1992] that is based on iteratively computing the centroidal Voronoi diagram (CVD) of a point distribution. Deussen et al. [2000] apply this technique to locally adjust the point spacing through interactive brushes, starting from initial point distribution—generated, e. g., by random sampling or halftoning—in which the point density reflects the intended gray values. The interactive but local application addresses a number of problems: the computational complexity of the technique as well as the issue that automatically processing the entire image would simply lead to a completely evenly distributed set of points. Thus, to allow automatic processing while maintaining the desired density, Secord [2002] uses weighted Voronoi diagrams to reflect the intended local point density. A related way of achieving stipple placement was explored by Schlechtweg et al. [2005] using a multi-agent system whose RenderBots evaluate their local neighborhood and try to move such that they optimize spacing to nearby agents with respect to the desired point density.

Besides evenly-spaced distributions it is sometimes desirable to achieve different dot patterns. For instance, Mould [2007] employs a distance-based path search in a weighted regular graph that ensures that stipple chains are placed along meaningful edges in the image. In another example, Kim et al. [2008] use a constrained version of Lloyd's method to the arrange stipples along offset lines to illustrate images of faces. However, in most cases the distribution should not contain patterns such as chains of stipple points—professional illustrators specifically aim to avoid these artifacts. Thus, Kopf et al. [2006] use Wang-tiling to arrange stipple tiles in large, non-repetitive ways and show how to provide a continuous level of detail while maintaining blue noise properties. This means that one can zoom into a stipple image with new points continuously being added to maintain the desired point density.

In addition to stipple placement, another issue that has previously been addressed is the shape of stipple points. While most early methods use circles or rounded shapes to represent stipples [Deussen et al. 2000; Secord 2002], several techniques have since been developed for other shapes, adapting Lloyd's method accordingly [Hiller et al. 2003], using a probability density function [Secord et al. 2002], or employing spectral packing [Dalal et al. 2006].



(a) Original photograph, Roman theater of Acinipo in Ronda, Málaga. (b) Interpreted regions.

Figure 3: *Original photograph and interpreted regions for Fig. 2.*

(a) Grayscale version of Fig. 3(a). (b) Adjusting global contrast and local detail of (a). (c) Manually added edges, inversion, local contrast.

Figure 4: *Deriving the helper image to capture the high-level stippling processes.*

It is also interesting to examine the differences between computer-generated stipple images and hand-drawn examples. For example, Isenberg et al. [2006] used an ethnographic pile-sorting approach to learn what people thought about both and what differences they perceive. They found that both the perfectly round shapes of stipple dots and the artifacts in placing them can give computer-generated images away as such, but also that people still valued them due to their precision and detail. Looking specifically at the statistics of stipple distributions, Maciejewski et al. [2008] quantified these differences with statistical texture measures and found that, for example, computer-generated stippling exhibits an undesired spatial correlation away from the stipple points and a lack of correlation close to them. This lead to the exploration of example-based stippling, for instance, by Kim et al. [2009]. They employ the same statistical evaluation as Maciejewski et al. [2008] and use it to generate new stipple distributions that have the same statistical properties as hand-drawn examples. By then placing scanned stipple dots onto the synthesized positions Kim et al. [2009] are able to generate convincing stipple illustrations. However, because the technique relies on being able to identify the centers of stipple points in the hand-drawn examples it has problems with middle tonal ranges because there stipple dots merge into larger conglomerates.

This issue of stipple merging in the middle tonal ranges is one problem that remains to be solved. In addition, while computer-generated stippling thus far has addressed stipple dot placement, stipple dot shapes, and animation, other aspects such as how to change an input image to create more powerful results (i. e., how to interpret the input image) have not yet been addressed.

3 Analysis of Hand-Drawn Stippling

To inform our technical approach for generating high-quality computer-generated stipple illustrations, we start by analyzing the process professional stipple illustrators perform when creating a drawing. For this purpose we involved a professional stipple artist and asked her to explain her approach/process using the example illustration shown in Fig. 2. From this analysis we extract a number of specific high-level processes that are often employed by professional stipple artists that go beyond simple dot placement and use of specific dot shapes. We discuss these in Section 3.1 before analyzing the low-level properties of stipple dots in Section 3.2 that guide our synthesis process.

3.1 High-Level Processes

The manual stipple process has previously been analyzed to inform computer-generated stippling. As part of this analysis and guided by literature on scientific illustration (e. g., [Hodges 2003]), researchers identified an even distribution of stipple points as one of the major goals (e. g., [Deussen et al. 2000; Secord 2002]) as well as the removal of artifacts (e. g., [Kopf et al. 2006; Maciejewski et al. 2008]). Also, Kim et al. [2009] noted the use of tone maps by illustrators to guide the correct reproduction of tone. While these aspects of stippling concentrate on rather low-level proper-

ties, there are also higher-level processes that stipple artists often employ in their work. Artists apply prior knowledge about good practices, knowledge about shapes and properties of the depicted objects, knowledge about the interaction of light with surfaces, and knowledge about the goal of the illustration. This leads to an interpretation of the original image or scene, meaning that stippling goes beyond an automatic and algorithmic tonal reproduction.

To explore these processes further we asked Elena Piñar, a professional illustrator, to create a stipple illustration (Fig. 2) from a digital photo (Fig. 3(a)). We observed and video-recorded her work on this illustration and also met with her afterwards to discuss her work and process. In this interview we asked her to explain the approach she took and the techniques she employed. From this conversation with her we could identify the following higher-level processes (see Fig. 3 for a visual explanation with respect to the hand-made illustration in Fig. 2 and photograph in Fig. 3(a)). While this list is not comprehensive, according to Elena Piñar it comprises the most commonly used and most important techniques (some of these are mentioned, e. g., by Guptill [1997]). Also, each artist has his/her own set of techniques as part of their own personal style.

Abstraction: One of the most commonly used techniques is removing certain parts or details in the image to focus the observer's attention on more important areas. In our example, the sky and some parts of the landscape have been fully removed (shown in violet in Fig. 3(b)). In addition, removing areas contributes to a better image contrast.

Increase in contrast: Some parts of the original color image exhibit a low level of contrast, reducing their readability when stippled. To avoid this problem illustrators increase the contrast in such regions (green area) through *global* and *local* evaluation of lightness, enhancing the detail where necessary.

Irregular but smoothly shaped outlines: If objects in an image are depicted with a regular or rectilinear shape they are often perceived as being man-made. To avoid this impression for natural shapes, stipple artists eliminate parts of these objects to produce an irregular form and add a tonal gradient (yellow).

Reduction of complexity: It is not always possible to remove all unimportant areas. In these cases the complexity or amount of detail is reduced. This effect is shown in orange in Fig. 3(b): the artist has removed some small parts that do not contribute to the illustration's intended message.

Additional detail: As visible in the red areas in Fig. 3(b), some parts that are not (or not clearly) visible in the original are still shown in the illustration. Here, the illustrator has enhanced details of the rocks based on her prior knowledge.

Inversion: Sometimes artists convert very dark zones or edges into very clear ones to improve the contrast. This technique is applied subjectively to specific parts of the drawing rather than to the image as a whole. In the hand-made stippled drawing the cracks between rocks are shown in white while they are black in the original photograph (blue in Fig. 3(b)).

Despite the fact that these high-level processes are an integral part of hand-drawn stipple illustration, computer-generated stippling techniques have largely concentrated on dot placement and dot shapes. This is understandable as these low-level processes can be automated while the higher-level processes to a large degree rely on human intelligence and sense of aesthetics. To be able to incorporate higher-level interpretations of images, therefore, we manually apply global and local image processing operations to the input image (Fig. 4). Instead of directly using a gray-level input image (Fig. 4(a)) we first apply pre-processing to accommodate the identified high-level processes. Following the list of processes given above (see Fig. 4(b)), we remove non-relevant parts from the image such as the sky and parts of the background. Also, we increase the contrast globally but also increase the brightness of some regions locally. Next, we locally delete parts of natural objects and smooth the border of these regions. To reduce the complexity of certain parts such as the metal grid we select these regions and apply a large degree of blur. Adding elements based on previous knowledge typically requires the artist painting into the image. Some additional information, however, can be added with algorithmic support, in particular edges that border regions that are similar in brightness such as the top borders of the ruin. We support this by either extracting an edge image from the original color image, taking only regions into account that have not been deleted in Fig. 4(b) and adding them to the helper image. Alternatively, artists can manually draw the necessary edges as shown in Fig. 4(c). Finally, inversion can be achieved by also manually drawing the intended inverted edges as white lines into the touched-up grayscale image (see Fig. 4(c)).

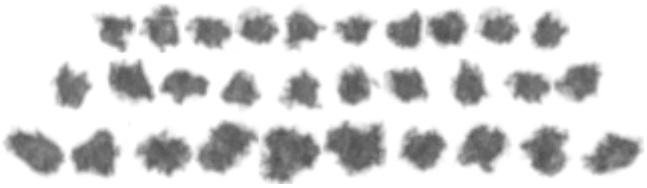


Figure 5: *Enlarged hand-drawn stipple dots (scanned at 1200 ppi).*

While this interactive pre-processing could be included into a comprehensive stipple illustration tool, we apply the manipulations using a regular image processing suite (e. g., Adobe Photoshop or GIMP). This allows us to make use of a great variety of image manipulation tools and effects to give us freedom to achieve the desired effects. The remainder of the process, on the other hand, is implemented in a dedicated tool. Before discussing our algorithm in detail, however, we first discuss some low-level aspects of hand-drawn stipple dots that are relevant for the approach.

3.2 Low-Level Properties of Stipple Dots

Stipple dots and their shapes have been analyzed and used in many previous computer stippling techniques, inspired by the traditional hand-drawn stippling. Hodges [2003] notes that each dot should have a purpose and that dots should not become dashes. In computer-generated stippling, therefore, dots have typically been represented as circular or rounded shapes[1] or pixels that are placed as black elements on a white ground. However, each use of a pen to place a dot creates a unique shape (e. g., Fig. 5) which is partially responsible for the individual characteristics of a hand-made stipple illustration. Thus, recent computer-generated stippling employed scans of dots from hand-made illustrations to better capture the characteristics of hand-drawn stippling [Kim et al. 2009].

We follow a similar approach and collected a database of stipple points from a high-resolution scan of a hand-drawn original illustration, a sample of which is shown in Fig. 5. These stipple dots are not all equal but have varying shapes and sizes. One can also notice that the stipples are not completely black but do exhibit a grayscale texture. This texture is likely due to the specific interaction of the pen's ink with the paper and typically disappears when the stipple illustration is reproduced in a printing process. This lead to the assumption that stipple dots are always completely black marks on a white background as used in much of the previous literature. However, the grayscale properties of real stipple dots are a characteristic

[1] Aside from more complex artificial shapes that have also been used but that are not necessarily inspired by hand-drawn stippling, see Section 2.

of the real stippling process which one may want to reproduce provided that one employs an appropriate reproduction technology. In addition, we also make use of these grayscale properties for realizing a technique that can address one of the remaining challenges in stippling: the merging of dots in the middle tonal ranges.

4 A Grayscale Stippling Process

Based on the previous analysis we now present our process for high-quality example-based stippling. In contrast to previous approaches, our process captures and maintains the stipple image throughout the entire process as high-resolution grayscale image, which allows us to achieve both stipple merging for the middle tonal ranges and high-quality print output. Also, to allow for interactive control of the technique, we use Ostromoukhov's [2001] fast error-diffusion halftoning technique to place the stipples. Below we step through the whole process by explaining the stipple placement (Section 4.1), the stipple dot selection and accumulation (Section 4.2), and the generation of both print and on-screen output (Section 4.3). In addition, we discuss adaptations for interactive processing (Section 4.4).

4.1 Stipple Dot Placement using Halftoning

Our stippling process starts by obtaining a grayscale version of the target image using the techniques described in Section 3.1. In principle, we use this image to run Ostromoukhov's [2001] error-diffusion technique to derive locations for placing stipple dots, similar to the use of halftoning to determine the starting distribution in the work by Deussen et al. [2000]. However, in contrast to their approach that adds point relaxation based on Lloyd's method, we use the locations derived from the halftoning directly which allows us to let stipple dots merge, unlike the results from relaxation. The reason for choosing a halftoning technique over, for example, distributions based on hand-drawn stipple statistics [Kim et al. 2009] is twofold. The main reason is that halftoning has the evaluation of tone built-in so that it does not require a tone map being extracted from the hand-drawn example. The second reason lies in that halftoning provides a continuum of pixel density representing tonal changes as opposed to the approach of using both black pixels on a white background for brighter regions and white pixels on a black background for darker areas as used by Kim et al. [2009].

Specifically, we are employing Ostromoukhov's [2001] error-diffusion, partially because it is easy to implement and produces results at interactive frame-rates. More importantly, however, the resulting point distributions have blue noise properties, a quality also desired by related approaches [Kopf et al. 2006]. This means that the result is nearly free of dot pattern artifacts that are present in results produced by many other error-diffusion techniques, a quality that is important for stippling [Hodges 2003].

Using a halftoning approach, however, means that we produce a point distribution based on pixels that are arranged on a regular grid, in contrast to stipples that can be placed at arbitrary positions. In addition, we cannot use the grayscale input image in the same resolution as the intended output resolution for the stipple image. Let us use an example to better explain this problem and describe its solution. Suppose we have a hand-stippled A4 image (in landscape) that we scan for analysis and extraction of stipple dot examples at 1200 ppi.[2] This means that the resulting image has roughly $14{,}000 \times 10{,}000$ pixels, with stipple dot sizes ranging from approximately 10×10 to 20×20 pixels.[3] If we now were to produce an equivalent A4 output image at 1200 ppi, would hence use an $14{,}000 \times 10{,}000$ pixel grayscale image, and compute its halftoned version at this size, each pixel in this image would represent one dot. This means we would need to place scanned stipple dots (whose average size is 16×16 pixels, this is equivalent to a spatial size of 0.338 mm, slightly larger than the nominal size of 0.2 mm of the tip of the used pen) at the pixel locations of the halftoned image. Consequently, this would produce a result that is 16× larger than the intended output image and reproducing it at A4 size would result in the characteristics of the stipple dot shapes being lost because each stipple would again be shrunken down to the size of ca. one pixel.

Therefore, for a given output resolution res_o we compute the dot distribution using error-diffusion halftoning at a smaller halftoning resolution res_{ht}. Intuitively for our 1200 ppi example, one could suggest a halftoning factor f_{ht} whose size is $1/16$ of res_o to compute res_{ht}, using the average stipple's diameter of 16 pixels:

$$res_{ht} = f_{ht} \cdot res_o \,; \quad f_{ht} = 1/16 \,; \quad res_o = 1200\,ppi\,. \tag{1}$$

In a completely black region and for ideally circular stipples, however, this would result in a pattern of white spots because the black dots on the grid only touch. To avoid this issue one has to use a factor f_{ht} of $\sqrt{2}/16$ to allow for a denser packing of stipple points such that they overlap. For realistic stipple points with non-circular shapes one may even have to use a f_{ht} of $2/16$ or more. On the other hand, even in the darkest regions in our example the stipple density is not such that they completely cover the canvas. Thus, we leave this choice up to the user to decide how dense to pack the stipples, and introduce a *packing factor* f_p to be multiplied with f_{ht} to control the density of the stipples:

$$f_{ht} = f_p/16\,. \tag{2}$$

For $f_p = 1$ and, thus, $f_{ht} = 1/16$ we would perform the halftoning in our example on an image with size 875×625 pixels to eventually yield a 1200 ppi landscape A4 output. For other output resolutions, however, the situation is different. Because the stipples have to use proportionally smaller pixel sizes for smaller resolutions to be reproduced at the same spatial sizes, the factor between output image and halftoning image has to change proportionally as well. For example, for a 600 ppi output resolution the average stipple's diameter would only be 8 pixels, and consequently f_{ht} would only be $1/8$. Thus, we can derive the halftoning resolution res_{ht} for a given output resolution res_o in ppi based on the observations we made from scanning a sample at 1200 ppi as follows:

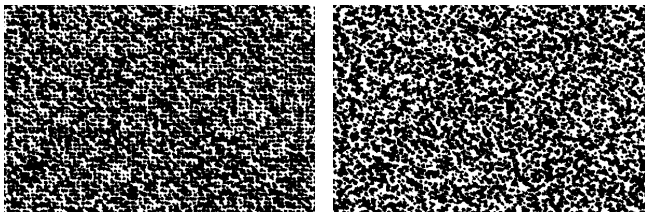
$$\begin{aligned} res_{ht} &= f_{ht} \cdot res_o \,; \quad f_{ht} = \frac{f_p \cdot 1200\,ppi}{16 \cdot res_o} \\ res_{ht} &= 75\,ppi \cdot f_p\,. \end{aligned} \tag{3}$$

This means that the halftoning resolution is, in fact, independent of the output resolution. Consequently, the pixel size of the halftoning image only depends on the spatial size of the intended output and the chosen packing factor (and ultimately the chosen scanned example stippling whose stipple dot size depends on the used pen).

This leaves the other mentioned problem that arises from computing the stipple distribution through halftoning: the stipple dots would be arranged on a regular grid, their centers always being at the centers of the pixels from the halftoning image. To avoid this issue, we perturb the locations of the stipple dots by a random proportion of between 0 and ±100% of their average diameter, in both x- and y-direction. Together with the random selection of stipple sizes from the database of scanned stipples and the blue noise quality of the dot distribution due to the chosen halftoning technique this successfully eliminates most observable patterns in dot placement (see Fig. 6).

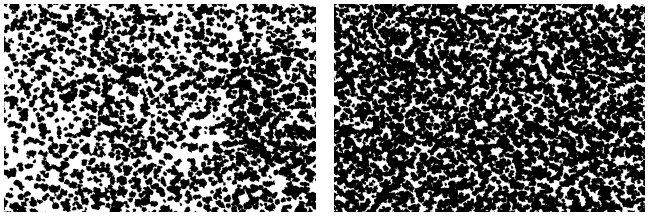
[2] Please notice that ppi stands for pixels per inch and is used intentionally while dots per inch (dpi) is used when we discuss printing.

[3] These values are derived from the example in Fig. 2 (done with a pen with a 0.2 mm tip) and can be assumed to be valid for many stipple images as similar pen sizes are used by professional illustrators [Hodges 2003].



(a) Grid placement of stipples. (b) After random perturbation.

Figure 6: *Magnified comparison of stipple placement before and after random perturbation of the stipple locations.*



(a) Lighter region. (b) Darker region.

Figure 7: *Merging of synthesized stipples at two tonal ranges.*

4.2 Stipple Dot Selection and Accumulation

We begin the collection of computed stipple points by creating a grayscale output buffer of the desired resolution, with all pixels having been assigned the full intensity (i. e., white). Then we derive the stipple placement by re-scaling the modified grayscale image to the halftoning resolution and running the error-diffusion as described in the previous section. Based on the resulting halftoned image and the mentioned random perturbations we can now derive stipple location with respect to the full resolution of the output buffer.

For each computed location we randomly select a scanned stipple dot from the previously collected database. This database is organized by approximate stipple dot sizes, so that for each location we first randomly determine the size class (using a normal distribution centered on the average size) and then randomly select a specific dot from this class. This is then added to the output buffer, combining intensities of the new stipple dot i_s and the pixels previously placed into the buffer i_{bg} using $(i_s \cdot i_{bg})/255$. This not only ensures that stipples placed on a white background are represented faithfully but also that the result gets darker if two dark pixels are combined (accumulation of ink).

Both the range of stipple sizes and the partially random stipple placement ensure that stipples can overlap. This overlapping is essential for our approach, it ensures the gradual merging of stipples into larger conglomerates as darker tones are reproduced (see example in Fig. 7). Therefore, we can for the first time simulate this aspect of the aesthetics of hand-drawn stippling.

4.3 Generation of Print and Screen Output

One challenge that remains is to generate the appropriate output for the intended device. Here we typically have two options: on-screen display and traditional print reproduction. These two options for output differ primarily in their spatial resolution and their color resolution. While normal bilevel printing offers a high spatial resolution (e. g., 1200 dpi), it has an extremely low color resolution (1 bit or 2) while typical displays have a lower spatial resolution (ca. 100 ppi) but a higher color resolution (e. g., 8 bit or 256 per primary). These differences also affect the goals for generating stipple illustrations. For example, it does not make much sense to print a grayscale stipple image because the properties of the individual stipple points (shape, grayscale texture) cannot be reproduced by most printing technology, they would disappear behind the pattern generated by the printer's halftoning [Isenberg et al. 2005]. In contrast, for on-screen display it does not make sense to generate a very high-resolution image because this cannot be seen on the screen.

Therefore, we adjust our stippling process according to the desired output resolutions, both color and spatial. For output designed for print reproduction we run the process at 1200 ppi, using a stipple library from a 1200 ppi scan, and compute the scaling factor for the halftoning process to place the stipples accordingly. The resulting 1200 ppi grayscale output image is then thresholded using a user-controlled cut-off value, and stored as a 1 bit pixel image, ready for high-quality print at up to 1200 dpi (e. g., Fig. 8). These images, of course, do no longer contain stipples with a grayscale texture but instead are more closely related to printed illustrations in books.

For on-screen display, in contrast, we run the process at a lower resolution, e. g., 300 ppi (while 300 ppi is larger than the typical screen resolution, it also allows viewers to zoom into the stipple image to some degree before seeing pixel artifacts). For this purpose the stipples in the database are scaled down accordingly, and the appropriate scaling value for the halftoning process is computed based on average stipple size at this lower resolution. The resulting image (e. g., Fig. 10) is smaller spatially but we preserve the texture information of the stipples. These can then be appreciated on the screen and potentially be colored using special palettes (e. g., sepia). In addition, grayscale stipple images can also be used for special continuous tone printing processes such as dye-sublimation.

4.4 User Interaction and Interactive Processing

Several parameters of the process can be adjusted interactively according to aesthetic considerations of the user, in addition to applying the high-level processes (parallel or as pre-processing). The most important settings are the intended (spatial) *size* and *output resolution* because these affect the resolutions at which the different parts of the process are performed. For example, a user would select A5 as the output size and 300 ppi as the intended resolution. Based on this the (pixel) resolution of both output buffer and halftoning buffer are derived as outlined in Section 4.1. To control the stipple density, we let the users interactively adjust the *packing factor* (the default value is 2). In addition we let users control the amount of *placement randomness* as a percentage of the average size of the stipple dots (the default value is 25%). This means that we specify the packing factor and placement randomness based on the average stipple size at the chosen resolution, which results in visually equivalent results regardless of which specific resolution is chosen.

Another aspect of user interaction is the performance of the process. While we can easily allow interactive work with the program at resolutions of up to 300 ppi for A4 output, the process is less responsive for larger images. For example, stippling the image shown in Fig. 4(c) takes approximately 0.51, 0.25, and 0.13 seconds for A4, A5, and A6 output, respectively, while a completely black input image requires 1.41, 0.70, and 0.36 seconds, respectively (Intel Core2 Duo E6600 at 3GHz with 2GB RAM, running Linux). However, our approach can easily allow users to adjust the parameters interactively at a lower resolution and then produce the final result at the intended high resolution such as for print output. For example, stippling Fig. 4(c) at A4 1200 dpi in black-and-white takes ca. 10.1 seconds while a completely black image requires approximately 15.2 seconds. We ensure that both low- and high-resolution results are equivalent by inherently computing the same halftoning resolution for both resolutions using the resolution-dependent scaling factor and appropriately seeding the random computations.

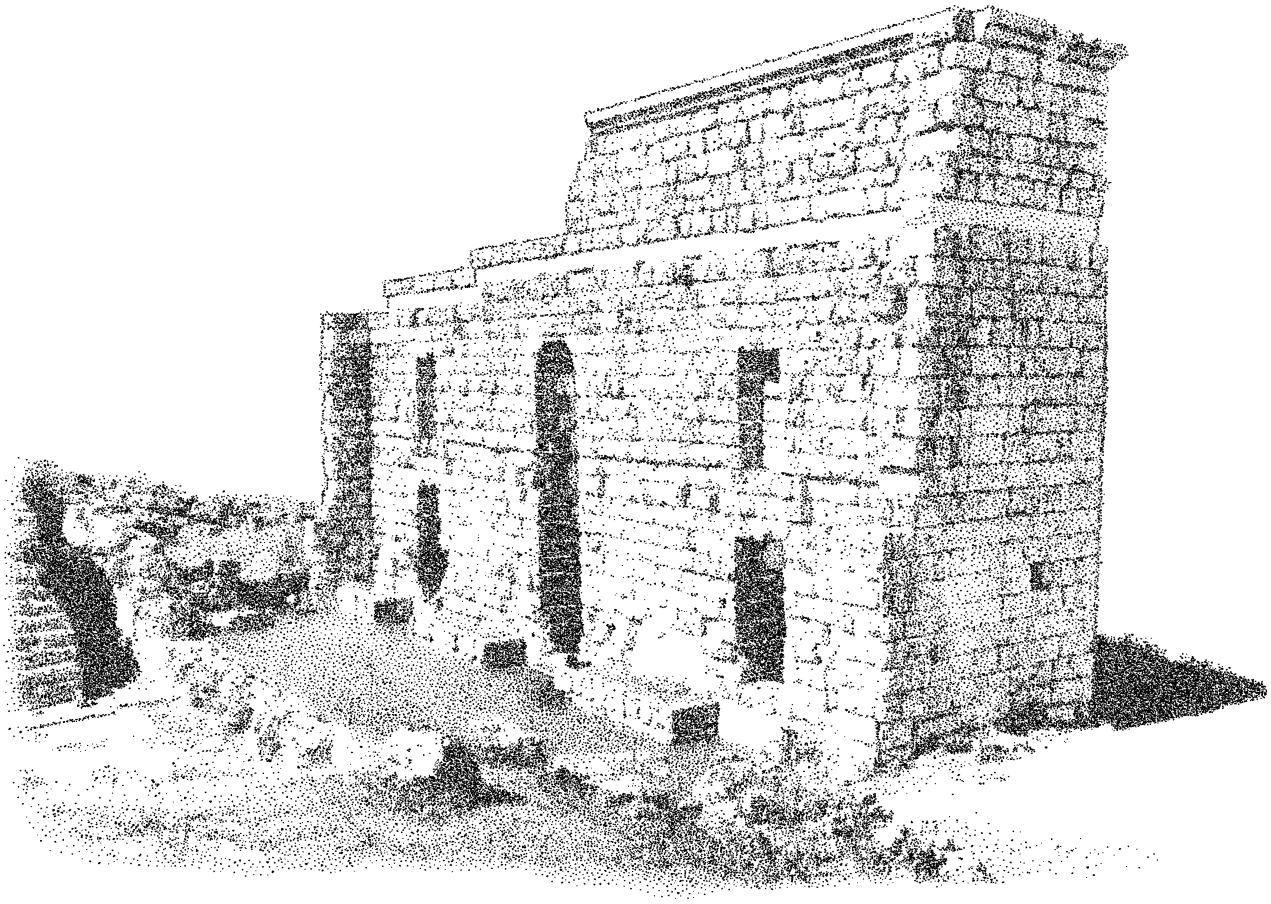
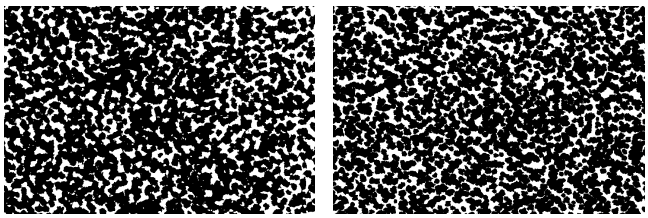


Figure 8: *Example generated for A4 print reproduction at 1200 dpi resolution, using Fig. 4(c) as input.*



(a) Detail from Fig. 2. (b) Detail from Fig. 8.

Figure 9: *Comparison of stipple merging between a hand-drawn and a synthesized sample, taken from the same region of the images.*

5 Results and Discussion

Fig. 8 shows a synthesized stipple image based on the photo (Fig. 3(a)) in its touched-up form (Fig. 4(c)) that was also used to create the hand-drawn example in Fig. 2. Fig. 8 was produced for print-reproduction at A5 and 1200 dpi. As can be seen from Fig. 9, our process can nicely reproduce the merging of stipples in a way that is comparable between the hand-drawn example and the computer-generated result. In addition, this process preserves the characteristic stipple outlines found in hand-drawn illustrations.

Fig. 10 shows an example produced for on-screen viewing. In contrast to the black-and-white image in Fig. 8, this time the grayscale texture of the stipple dots is preserved. In fact, in this example we replaced the grayscale color palette with a sepia palette to give the illustration a warmer tone, a technique that is often associated with aging materials. However, in typical print processes, images like this will be reproduced with halftoning to depict the gray values. These halftoning patterns typically 'fight' with the stipple shapes and placement patterns. To avoid these, one has to produce b/w output as discussed before or use dye-sublimation printing. Fig. 11–13 show two more black-and-white examples and one grayscale one.

Of course, our approach is not without flaws. An important problem arises from the use of halftoning on a lower resolution to derive stipple distributions because this initially leads to the stipples being placed on a grid. While we address this grid arrangement by introducing randomness to the final stipple placement, this also leads to noise being added to otherwise clear straight lines in the input image. This effect can be observed by comparing the upper edge of the ruin in Fig. 8 with the same location in the hand-made example in Fig. 2 where the line of stipple dots is nicely straight. This issue could possibly be addressed by analyzing the local character of the source image with an edge detector: if no edges are found use randomness as before; otherwise reduce the amount of introduced randomness for placing stipples.

One final aspect that we would like to discuss in the context stipple placement the choice of halftoning in the first place. While we have presented the reasons for our decision in Section 4.1, one may also argue that it may be better to use other types of halftoning (e. g., [Chang et al. 2009]) or try to adapt Kim et al.'s [2009] technique to fit our needs. To investigate this issue further, we took samples from both Figures 2 and 8 and analyzed them using Maciejewski et al.'s [2008] technique. The result of this analysis showed that the examined hand-drawn and our computer-generated stippling examples exhibit almost identical statistical behavior when compared with each other with respect to the correlation, energy, and contrast

Figure 10: *Sepia tonal stipple example generated for A4 on-screen viewing at up to 300 ppi resolution, with slight gamma correction.*

measures. Also, our computer-generated examples do not exhibit the correlation artifacts described by Maciejewski et al. [2008] for other computer-generated stippling techniques. Thus, our choice to base the stipple distribution on halftoning seems to be justified.

6 Conclusion and Future Work

In summary, we presented a scale-dependent, example-based stippling technique that supports both low-level stipple placement and high-level interaction with the stipple illustration. In our approach we employ halftoning for stipple placement and focus on the stipples' shape and texture to produce both gray-level output for on-screen viewing and high-resolution binary output for printing. By capturing and maintaining the stipple dots as grayscale textures throughout the process we solve the problem of the merging of stipple dots at intermediate resolutions as previously reported by Kim et al. [2009]. The combined technique allows us to capture the entire process from artistic and presentation decisions of the illustrator to the scale-dependence of the produced output.

One of the interesting observations from this process is that the resolution at which the stipple distribution occurs (using halftoning in our case) depends on the spatial size of the target image but needs to be independent from its resolution, just like other pen-and-ink rendering [Salisbury et al. 1996; Jeong et al. 2005; Ni et al. 2006]. For example, there should be the same number of stipples for a 1200 ppi printer as there should be for a 100 ppi on-screen display. However, there need to be fewer stipples for an A6 image compared to an A4 image. This complements the observation by Isenberg et al. [2006] that stippling with many dense stipple points is often perceived by viewers to be computer-generated.

While our approach allows us to support the interactive creation of stipple illustrations, this process still has a number of limitations. Besides the mentioned limitations in stipple placement at edges in the input image, one of the most important limitations concerns the presentation of the interaction: the use of high-level processes as described in Section 3.1 is currently a separate process that does require knowledge of the underlying artistic principles—a better integration of this procedure into the user interface would be desirable. Also, we would like to investigate additional algorithmic support for these high-level interaction. This includes, for example, an advanced color-to-gray conversion techniques [Gooch et al. 2005; Rasche et al. 2005a; Rasche et al. 2005b] to support illustrators in their work. In addition, an interactive or partially algorithmically supported creation of layering or image sections according to the discussed high-level criteria such as background, low or high detail, level of contrast, or inversion would be interesting to investigate as future work. For this automatic or salience-based abstraction techniques [Santella and DeCarlo 2004] could be employed.

Acknowledgments

We thank, in particular, Elena Piñar for investing her time and creating the stippling examples for us. We also thank Ross Maciejewski who provided the stipple statistics script and Moritz Gerl for his help with Matlab. Finally, we acknowledge the support of the Spanish Ministry of Education and Science to the projects TIN2007-67474-C03-02 and TIN2007-67474-C03-01.

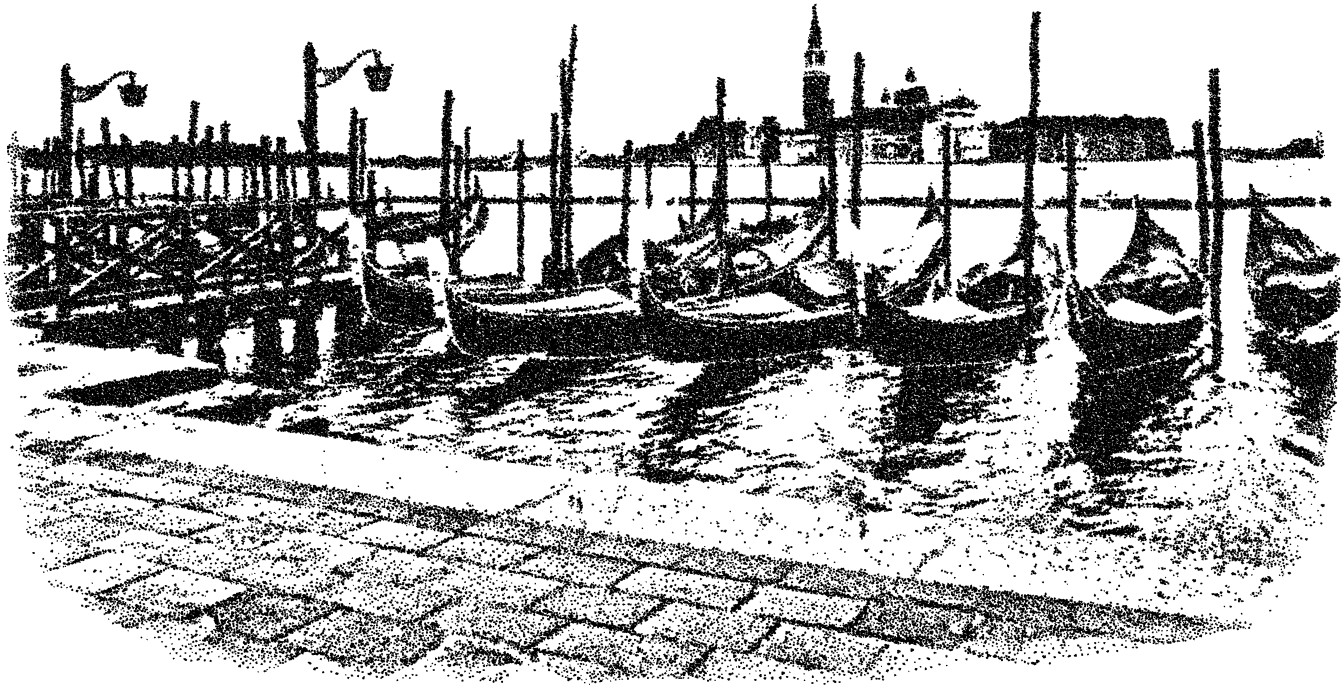


Figure 11: *A5 1200 dpi black-and-white stippling of a view of Venice.*

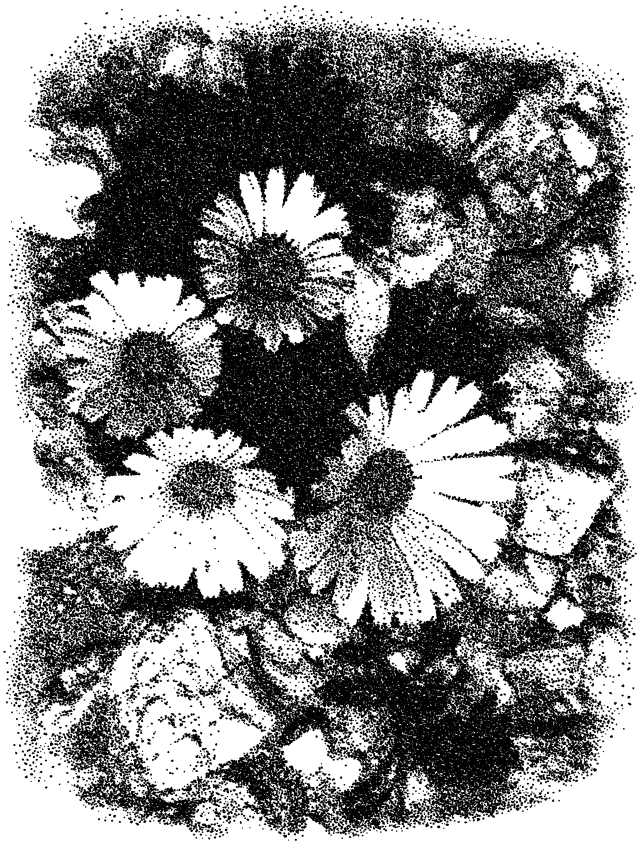


Figure 12: *A6 1200 dpi black-and-white stippling of flowers.*

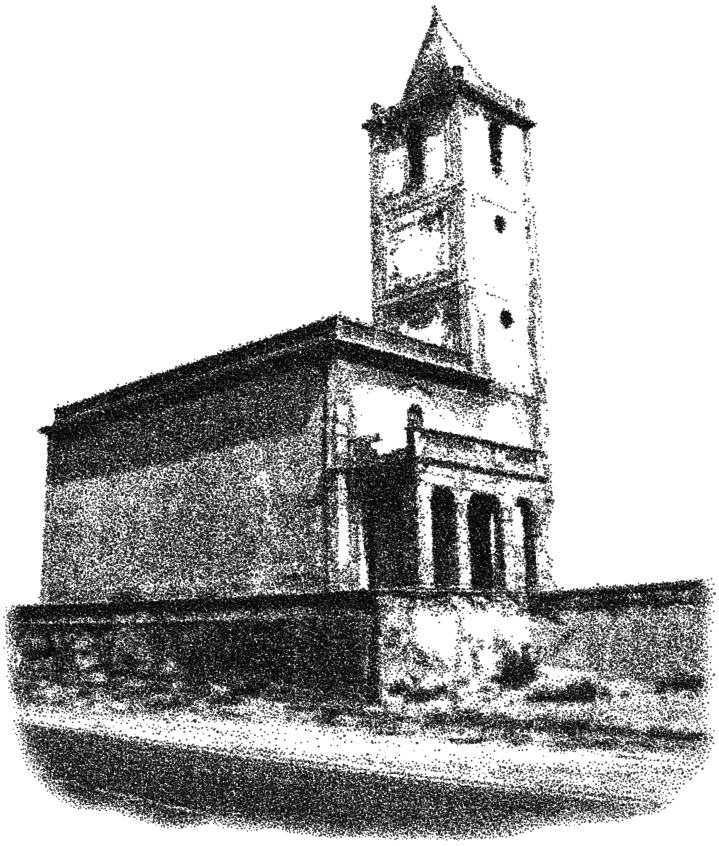


Figure 13: *A6 300 ppi grayscale stippling of a church.*

References

CHANG, J., ALAIN, B., AND OSTROMOUKHOV, V. 2009. Structure-Aware Error-Diffusion. *ACM Transactions on Graphics 28*, 5 (Dec.), 162:1–162:8. DOI: 10.1145/1618452.1618508

DALAL, K., KLEIN, A. W., LIU, Y., AND SMITH, K. 2006. A Spectral Approach to NPR Packing. In *Proc. NPAR*, ACM, New York, 71–78. DOI: 10.1145/1124728.1124741

DEUSSEN, O., HILLER, S., VAN OVERVELD, C., AND STROTHOTTE, T. 2000. Floating Points: A Method for Computing Stipple Drawings. *Computer Graphics Forum 19*, 3 (Sept.), 40–51. DOI: 10.1111/1467-8659.00396

FOSTER, K., JEPP, P., WYVILL, B., SOUSA, M. C., GALBRAITH, C., AND JORGE, J. A. 2005. Pen-and-Ink for BlobTree Implicit Models. *Computer Graphics Forum 24*, 3 (Sept.), 267–276. DOI: 10.1111/j.1467-8659.2005.00851.x

GOOCH, A. A., OLSEN, S. C., TUMBLIN, J., AND GOOCH, B. 2005. Color2Gray: Salience-Preserving Color Removal. In *Proc. SIGGRAPH*, ACM, New York, 634–639. DOI: 10.1145/1186822.1073241

GUPTILL, A. L. 1997. *Rendering in Pen and Ink*. Watson-Guptill Publications, New York.

HERTZMANN, A. 2003. A Survey of Stroke-Based Rendering. *IEEE Computer Graphics and Applications 23*, 4 (July/Aug.), 70–81. DOI: 10.1109/MCG.2003.1210867

HILLER, S., HELLWIG, H., AND DEUSSEN, O. 2003. Beyond Stippling – Methods for Distributing Objects on the Plane. *Computer Graphics Forum 22*, 3 (Sept.), 515–522. DOI: 10.1111/1467-8659.00699

HODGES, E. R. S., Ed. 2003. *The Guild Handbook of Scientific Illustration*, 2nd ed. John Wiley & Sons, Hoboken, NJ, USA.

ISENBERG, T., CARPENDALE, M. S. T., AND SOUSA, M. C. 2005. Breaking the Pixel Barrier. In *Proc. CAe*, Eurographics Association, Aire-la-Ville, Switzerland, 41–48. DOI: 10.2312/COMPAESTH/COMPAESTH05/041-048

ISENBERG, T., NEUMANN, P., CARPENDALE, S., SOUSA, M. C., AND JORGE, J. A. 2006. Non-Photorealistic Rendering in Context: An Observational Study. In *Proc. NPAR*, ACM, New York, 115–126. DOI: 10.1145/1124728.1124747

JEONG, K., NI, A., LEE, S., AND MARKOSIAN, L. 2005. Detail Control in Line Drawings of 3D Meshes. *The Visual Computer 21*, 8–10 (Sept.), 698–706. DOI: 10.1007/s00371-005-0323-1

KIM, D., SON, M., LEE, Y., KANG, H., AND LEE, S. 2008. Feature-Guided Image Stippling. *Computer Graphics Forum 27*, 4 (June), 1209–1216. DOI: 10.1111/j.1467-8659.2008.01259.x

KIM, S., MACIEJEWSKI, R., ISENBERG, T., ANDREWS, W. M., CHEN, W., SOUSA, M. C., AND EBERT, D. S. 2009. Stippling By Example. In *Proc. NPAR*, ACM, New York, 41–50. DOI: 10.1145/1572614.1572622

KOPF, J., COHEN-OR, D., DEUSSEN, O., AND LISCHINSKI, D. 2006. Recursive Wang Tiles for Real-Time Blue Noise. *ACM Transactions on Graphics 25*, 3 (July), 509–518. DOI: 10.1145/1141911.1141916

LLOYD, S. P. 1982. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (Mar.), 129–137.

LU, A., MORRIS, C. J., EBERT, D. S., RHEINGANS, P., AND HANSEN, C. 2002. Non-Photorealistic Volume Rendering using Stippling Techniques. In *Proc. VIS*, IEEE Computer Society, Los Alamitos, 211–218. DOI: 10.1109/VISUAL.2002.1183777

LU, A., TAYLOR, J., HARTNER, M., EBERT, D. S., AND HANSEN, C. D. 2002. Hardware-Accelerated Interactive Illustrative Stipple Drawing of Polygonal Objects. In *Proc. VMV*, Aka GmbH, 61–68.

LU, A., MORRIS, C. J., TAYLOR, J., EBERT, D. S., HANSEN, C., RHEINGANS, P., AND HARTNER, M. 2003. Illustrative Interactive Stipple Rendering. *IEEE Transactions on Visualization and Computer Graphics 9*, 2 (Apr.–June), 127–138. DOI: 10.1109/TVCG.2003.1196001

MACIEJEWSKI, R., ISENBERG, T., ANDREWS, W. M., EBERT, D. S., SOUSA, M. C., AND CHEN, W. 2008. Measuring Stipple Aesthetics in Hand-Drawn and Computer-Generated Images. *IEEE Computer Graphics and Applications 28*, 2 (Mar./Apr.), 62–74. DOI: 10.1109/MCG.2008.35

MCCOOL, M., AND FIUME, E. 1992. Hierarchical Poisson Disk Sampling Distributions. In *Proc. Graphics Interface*, Morgan Kaufmann Publishers Inc., San Francisco, 94–105.

MERUVIA PASTOR, O. E., FREUDENBERG, B., AND STROTHOTTE, T. 2003. Real-Time Animated Stippling. *IEEE Computer Graphics and Applications 23*, 4 (July/Aug.), 62–68. DOI: 10.1109/MCG.2003.1210866

MOULD, D. 2007. Stipple Placement using Distance in a Weighted Graph. In *Proc. CAe*, Eurographics Assoc., Aire-la-Ville, Switzerland, 45–52. DOI: 10.2312/COMPAESTH/COMPAESTH07/045-052

NI, A., JEONG, K., LEE, S., AND MARKOSIAN, L. 2006. Multi-Scale Line Drawings from 3D Meshes. In *Proc. I3D*, ACM, New York, 133–137. DOI: 10.1145/1111411.1111435

OSTROMOUKHOV, V. 2001. A Simple and Efficient Error-Diffusion Algorithm. In *Proc. SIGGRAPH*, ACM, New York, 567–572. DOI: 10.1145/383259.383326

RASCHE, K., GEIST, R., AND WESTALL, J. 2005. Detail Preserving Reproduction of Color Images for Monochromats and Dichromats. *IEEE Computer Graphics and Applications 25*, 3 (May), 22–30. DOI: 10.1109/MCG.2005.54

RASCHE, K., GEIST, R., AND WESTALL, J. 2005. Re-Coloring Images for Gamuts of Lower Dimension. *Computer Graphics Forum 24*, 3 (Sept.), 423–432. DOI: 10.1111/j.1467-8659.2005.00867.x

SALISBURY, M. P., ANDERSON, C., LISCHINSKI, D., AND SALESIN, D. H. 1996. Scale-Dependent Reproduction of Pen-and-Ink Illustration. In *Proc. SIGGRAPH*, ACM, New York, 461–468. DOI: 10.1145/237170.237286

SANTELLA, A., AND DECARLO, D. 2004. Visual Interest and NPR: An Evaluation and Manifesto. In *Proc. NPAR*, ACM, New York, 71–150. DOI: 10.1145/987657.987669

SCHLECHTWEG, S., GERMER, T., AND STROTHOTTE, T. 2005. RenderBots—Multi Agent Systems for Direct Image Generation. *Computer Graphics Forum 24*, 2 (June), 137–148. DOI: 10.1111/j.1467-8659.2005.00838.x

SCHMIDT, R., ISENBERG, T., JEPP, P., SINGH, K., AND WYVILL, B. 2007. Sketching, Scaffolding, and Inking: A Visual History for Interactive 3D Modeling. In *Proc. NPAR*, ACM, New York, 23–32. DOI: 10.1145/1274871.1274875

SECORD, A., HEIDRICH, W., AND STREIT, L. 2002. Fast Primitive Distribution for Illustration. In *Proc. EGWR*, Eurographics Association, Aire-la-Ville, Switzerland, 215–226. DOI: 10.1145/581924.581924

SECORD, A. 2002. Weighted Voronoi Stippling. In *Proc. NPAR*, ACM, New York, 37–43. DOI: 10.1145/508530.508537

SOUSA, M. C., FOSTER, K., WYVILL, B., AND SAMAVATI, F. 2003. Precise Ink Drawing of 3D Models. *Computer Graphics Forum 22*, 3 (Sept.), 369–379. DOI: 10.1111/1467-8659.00684

VANDERHAEGHE, D., BARLA, P., THOLLOT, J., AND SILLION, F. X. 2007. Dynamic Point Distribution for Stroke-based Rendering. In *Rendering Techniques*, Eurographics Association, Aire-la-Ville, Switzerland, 139–146. DOI: 10.2312/EGWR/EGSR07/139-146

XU, H., AND CHEN, B. 2004. Stylized Rendering of 3D Scanned Real World Environments. In *Proc. NPAR*, ACM, New York, 25–34. DOI: 10.1145/987657.987662

YUAN, X., NGUYEN, M. X., ZHANG, N., AND CHEN, B. 2005. Stippling and Silhouettes Rendering in Geometry-Image Space. In *Proc. EGSR*, Eurographics Association, Aire-la-Ville, Switzerland, 193–200. DOI: 10.2312/EGWR/EGSR05/193-200

ZAKARIA, N., AND SEIDEL, H.-P. 2004. Interactive Stylized Silhouette for Point-Sampled Geometry. In *Proc. GRAPHITE*, ACM, New York, 242–249. DOI: 10.1145/988834.988876