Example-Based Stippling using a Scale-Dependent Grayscale Process
Martín, Domingo; Arroyo, Germán; Luzón, M. Victoria; Isenberg, Tobias

Published in: EPRINTS-BOOK-TITLE

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher’s PDF, also known as Version of record

Publication date:
2010

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license. More information can be found on the University of Groningen website: https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment.

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.
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 stipbles (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 losing spatial precision due to halftoning artifacts. This property and the simplicity of the dot as the main rendering element 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:

- 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 stiples 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.
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 stippledots 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 stipbles 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].
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. 4) instead of directly using a grayscale input image.

Shape 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 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 of the illustration and not necessarily inspired by hand-drawn stippling, see Section 2.
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. This means that the resulting image has roughly 14,000 × 10,000 pixels, with stipple dot sizes ranging from approximately 10 × 10 to 20 × 20 pixels. If we now were to produce an equivalent A4 output image at 1200 ppi, would hence use an 14,000 × 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 16x 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 shrunk down to the size of ca. one pixel.

Therefore, for a given output resolution reso, we compute the dot distribution using error-diffusion halftoning at a smaller halftoning resolution resht. Intuitively for our 1200 ppi example, one could suggest a halftoning factor fht whose size is 1/16 of reso, to compute resht, using the average stipple’s diameter of 16 pixels:

\[ res_{ht} = f_{ht} \cdot res_{o}; \quad f_{ht} = 1/16; \quad res_{o} = 1200 \text{ ppi}. \quad (1) \]

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 fht 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 fht 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 fp to be multiplied with fht to control the density of the stipples:

\[ f_{ht} = f_{p}/16. \quad (2) \]

For fp = 1 and thus, fht = 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 fht would only be 1/8. Thus, we can derive the halftoning resolution resht for a given output resolution reso in ppi based on the observations we made from scanning a sample at 1200 ppi as follows:

\[ res_{ht} = f_{ht} \cdot res_{o}; \quad f_{ht} = f_{p} \cdot 1200 \text{ ppi}; \quad res_{o} = 1200 \text{ ppi}. \quad (3) \]

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 \( \pm 100\% \) 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).
We begin the collection of computed stipple points by creating a
we first randomly determine the size class (using a normal distri-
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 1200ppi, 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 res-
olution, 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 ap-
propriate 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 infor-
mation 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 con-
tinuous tone printing processes such as dye-sublimation.

4.2 Stipple Dot Selection and Accumulation

We begin the collection of computed stipple points by creating a
gray-scale output buffer of the desired resolution, with all pixels hav-
ing 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 lo-
cation 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 orga-
ized by approximate stipple dot sizes, so that for each location
we first randomly determine the size class (using a normal distri-
bution 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 $i_d$ and the pixels previ-
ously placed into the buffer $i_b$ using $(i_d - i_b) / 255$. 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 es-
cential for our approach, it ensures the gradual merging of stipples
into larger conglomerates as darker tones are reproduced (see ex-
ample 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 res-
olution (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 pri-
mary). 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 con-
trast, 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

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

Figure 7: Merging of synthesized stipples at two tonal ranges.
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 is 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 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.
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 stipple’s 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 stipple for a 1200 ppi printer as there should be for a 100 ppi on-screen display. However, there need to be fewer stipple 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 algorithimic 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


