



HAL
open science

Interpolation of Point Distributions for Digital Stippling

Germán Arroyo, Domingo Martín, Tobias Isenberg

► **To cite this version:**

Germán Arroyo, Domingo Martín, Tobias Isenberg. Interpolation of Point Distributions for Digital Stippling. 2023. hal-04212086

HAL Id: hal-04212086

<https://inria.hal.science/hal-04212086v1>

Preprint submitted on 20 Sep 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Interpolation of Point Distributions for Digital Stippling

GERMÁN ARROYO, University of Granada, Spain

DOMINGO MARTÍN, University of Granada, Spain

TOBIAS ISENBERG, Université Paris-Saclay, CNRS, Inria, LISN, France

We present a new way to merge any two point distribution approaches using distance fields. Our new process allows us to produce digital stippling that fills areas with stipple dots without visual artifacts as well as includes clear linear features without fussiness. Our merging thus benefits from past work that can optimize for either goal individually, yet typically by sacrificing the other. The new possibility of combining any two distributions using different distance field functions and their parameters also allows us to produce a vast range of stippling styles, which we demonstrate as well.

CCS Concepts: • **Computing methodologies** → **Non-photorealistic rendering**.

Additional Key Words and Phrases: point distributions, digital stippling.

ACM Reference Format:

Germán Arroyo, Domingo Martín, and Tobias Isenberg. 2021. Interpolation of Point Distributions for Digital Stippling. *ACM Trans. Graph.* 1, 1 (July 2021), 18 pages. <https://doi.org/10.1145/1122445.1122456>

1 INTRODUCTION

Digital stippling [Deussen and Isenberg 2013; Martín et al. 2017] is a non-photorealistic rendering technique [Gooch and Gooch 2001; Rosin and Collomosse 2013; Strothotte and Schlechtweg 2002] that emulates hand-made stippling by means of distributing stipple dots based on target images. The traditional art form has been and continues to be used for illustrations in several scientific domains such as biology, entomology, or archeology. For artists, while it seemingly only requires to place dots on paper using an ink pen, the traditional technique is an arduous and extremely repetitive task. It requires of time to produce relatively small drawings (e. g., A4) because the artist must place thousands of dots and has to correctly reproduce tone, shape, and texture. Ultimately, this illustration technique is thus increasingly being replaced by others due to its high cost.

Research in NPR, however, has led to numerous approaches for producing digital stippling without such long production times [Martín et al. 2017], taking into account not only the actual dot distribution but also the image resolutions [Martín et al. 2010, 2011], the shape of the dots [Martín et al. 2019, 2015], and properties such as the introduced abstraction [Spicker et al. 2019, 2017]. One of the main driving forces of digital stippling has been the stipple dot placement without visual artifacts, as early approaches based on

Authors' addresses: Germán Arroyo, arroyo@ugr.es, University of Granada, , , , Spain; Domingo Martín, dmartin@ugr.es, University of Granada, , , , Spain; Tobias Isenberg, tobias.isenberg@inria.fr, Université Paris-Saclay, CNRS, Inria, LISN, , , France.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2021 Association for Computing Machinery.

0730-0301/2021/7-ART \$15.00

<https://doi.org/10.1145/1122445.1122456>

Centroidal Voronoi Diagrams [Lloyd 1982; McCool and Fiume 1992] lead to unintended chains of dots which stipple artists are trained to avoid [Hodges 2003]. While several approaches have since addressed this artifact issue (see the overview in Martín et al.'s [2017] survey), sometimes it is still important that the dots are, in fact, aligned to intentionally represent dedicated features. Consequently, some approaches (e. g., [Li and Mould 2011, 2017; Mould 2007]) focus specifically on placing dots in a structure-aware fashion.

The fundamental problem is now that any given point distribution technique usually either produces nice point distributions for filling areas of an image, or it produces adequate feature stippling. More generally, to produce a digital stippling one may want to combine any two dot distribution techniques that one chooses for a given visual goal. In this article we thus describe a new approach to realize such a smooth interpolation between two point distributions such that their respective properties are maintained. For this purpose we consider stippling algorithms as discrete probabilistic functions, so we can create a new algorithm based on the interpolated function of their distributions. Our approach to combine two distributions is general enough such that it not only allows us to solve our particular problem. Instead, it also allows us to produce a great variety of results by only selecting different distributions and/or adjusting the distance field, which we see as a novel way of controlling the stylization for digital stippling.

2 PREVIOUS WORK

A complete survey of digital stippling is beyond the scope of this section and has recently been presented elsewhere [Martín et al. 2017], so here we only briefly mention some main related work and those techniques that specifically apply to our approach. The reproduction of stippling using computers began in the late 1990s. Researchers initially tried to place dots appropriately to represent a picture. These initial solutions were based on the concept of Centroidal Voronoi Diagrams (also called Lloyd's method [Lloyd 1982; McCool and Fiume 1992]); the main idea is to equalize the energy between Voronoi regions by moving the Voronoi sites to the cell's centroids. The first implementation was a painting systems with brushes to modify the dot distribution [Deussen et al. 2000, 1999]. Secord [2002b] later extended this approach by using the tone of the original image as a weight to control the distance metric.

Original CVDs and their extensions have the mentioned problem of producing patterns due to the regularity of the distribution, which must be avoided for realistic reproduction [Hodges 2003]. Several solutions have been investigated to reduce or solve this problem. Schlechtweg et al. [2005], for instance, used autonomous agents, the Renderbots, to place stipples with some random processing. Another idea is to use distributions with blue noise properties. Kopf et al. [2006], for example, presented a method based on Wang tiles and Poisson disk sampling, which has blue noise characteristics and

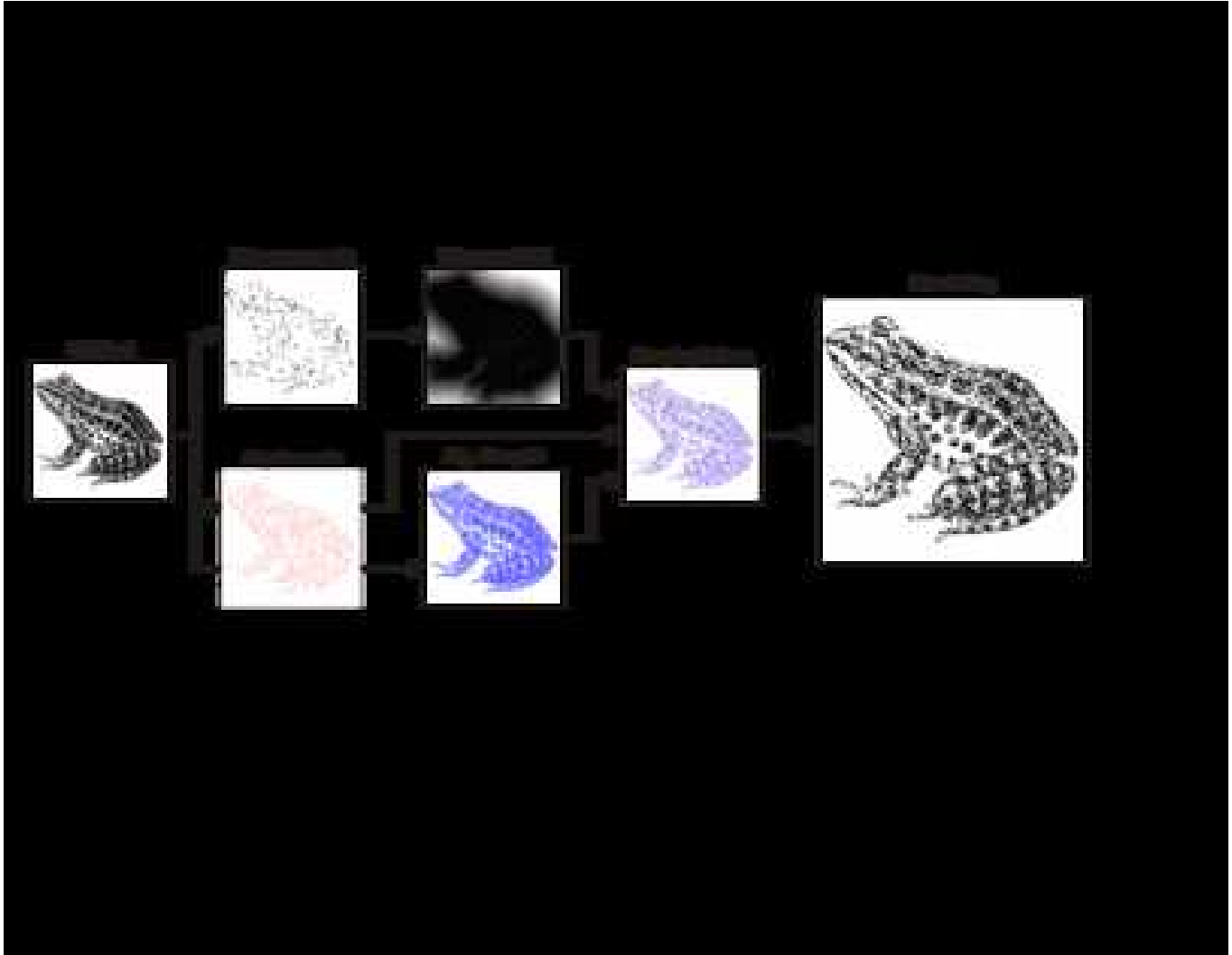


Fig. 1. Overview of our method (while we use color here to distinguish the distributions, the result only uses grayscale or black and white).

thus avoids patterns and, moreover, allows them to smoothly and indefinitely zoom into stipple images. Balzer et al. [2009] presented a capacity-constrained way to create point distributions based on Lloyd’s method that also possess blue noise characteristics. A generalization of the original CVD but using an optimization of the energy was described by Deussen [2009]. Another blue noise approach that is also fast was presented by Ascencio-Lopez et al. [2010]. Finally, Arroyo et al. [2010] used a Monte Carlo technique for sampling an adaptive probability density function.

Another way to avoid visible patterns is to use example-based techniques and hand-made stippling as input. For example, Barla et al. [2006a,b] synthesized different hatching and stippling styles using techniques from texture synthesis. Later, Kim et al. [2009] synthesized dot distributions using gray-level co-occurrence matrices (GLCM) for a statistical analysis of hand-made samples. Finally, Martín et al. [2010, 2011] showed that the use of resolution-dependent halftoning for dot distributions combined with scanned

dots can also achieve satisfactory results.

In contrast to the approaches discussed thus far that aim to avoid visual details, others attempt to use stipples that line up to form linear structures on purpose. One form is called hedcut stippling [Kim et al. 2008, 2010; Son et al. 2011] and is somewhat related to hatching. More related to our own work, however, are structure-aware techniques that aim to highlight particular sparse features in an image using aligned stipples, while reproducing the remainder of the image without such artifacts. For example, Mould [2007] used a graph whose edge weights recorded the magnitude of the local image gradient. He then used a version of Dijkstra’s algorithm to determine stipple dot positions, emphasizing particularly edges in an image. Later, Li and Mould [2011, 2017] based another structure-aware stippling technique on an approach for contrast-aware halftoning [Chang et al. 2009; Li and Mould 2010].

None of the techniques are ideal, however. For example, while Li and Mould [2011, 2017] are able to nicely emphasize linear structures,

the regularity of their dot placement is sometimes noticeable, in particular, in dark regions. In contrast, example-based approach such as those by Kim et al. [2009] and Martín et al. [2011] can produce convincing distributions for relatively dark or relatively light regions, but fail to properly reproduce linear structures.

3 OVERVIEW

Our visual goal is that of hand-made stippling. Here, artists avoid any visible artifact in dot placement for most of the stippled regions [Hodges 2003]. To represent features, edges, or borders, however, they align the dots or even use proper lines instead of chained dots (e. g., see the examples in Hodges’s [2003] book or in Figure 1 in Martín et al.’s [2017] survey). To be able to achieve a similar effect with digital stippling we need to address the mentioned problem of recent related work which either focuses on artifact avoidance or on feature representation. In this paper we thus demonstrate how we can interpolate between any given two distributions to be able to use them for those parts where they shine, and use others where they would not be ideal. We base our approach on a dedicated distance field that we use to determine which of the two distributions to use. Usually we derive this field using some edge detector, but other fields are possible. With this approach we can smoothly transition between any two given point distributions, without introducing new patterns to the stippling.

We show a graphical overview of our approach in Figure 1. In general terms, our method takes two stippling algorithms and a distance field such as one that is computed from an edge detector applied to the source image. We then construct a stochastic function from this information to get the Probability Density Functions (PDFs) of both algorithms. We then interpolate the two PDFs according to the distance field, and then rendering the result to obtain the final illustration. Next, we first explain the general method to combine any two point distributions, before we demonstrate how this approach allows us to solve the artifact issue for stippling.

4 DISTRIBUTIONS AND INTERPOLATION FUNCTIONS

Any stippling algorithm can be considered as a function that takes a source image (as well as certain parameters) as input and then determines dot positions on a virtual paper (ignoring the dot aspects [Martín et al. 2019, 2015] for now). We can consider the process of placing the dots as a random function that places a dot at a particular location. In the case of those stippling algorithms that use random numbers (e. g., [Arroyo et al. 2010; Li and Mould 2011, 2017; Schlechtweg et al. 2005]), the probability of placing the dots is given by the techniques’ respective Probability Density Functions (PDFs) directly. For the rest of deterministic stippling and point distribution algorithms, we can simulate a corresponding stochastic version simply by sampling the stippling result according to the density of stippling dots per area and deriving a respective PDF.

A PDF is a function that makes use of a random variable X and has a density function associated f_X , where f_X is a non-negative Lebesgue-integrable function. It can thus be defined as

$$\Pr[a \leq X \leq b] = \int_a^b f_X(x) dx .$$

The 2D nature of the virtual stippling canvas makes it convenient

to redefine the PDF in terms of area instead of some interval as

$$\Pr[X \in A] = \int_A f_X(x) dx .$$

In our case, X represents some random stippling dot so, intuitively, one can think of $f_X(x) dx$ as being the probability of some random stippling dot X falling within some infinitesimal area A .

The linear interpolation of two different PDFs (f and g) for the same area A can be written [Bursal 1996] as

$$I_X(\alpha) = (1 - \alpha)f_X + \alpha g_X, \text{ with } X \in A \quad (1)$$

for some scalar value $\alpha \in [0, 1]$. We thus have

$$\Pr[X \in A] = (1 - \alpha) \int_A f_X(x) dx + \alpha \int_A g_X(x) dx, \text{ with } X \in A .$$

For those stippling algorithms that directly use continuous, integrable functions it is enough to compute the interpolation of the respective two PDFs to merge both algorithms. But even those algorithms that use PDFs for the dot placement typically redefine the PDF in terms of time: the PDF is constructed while the dots are placed. Therefore, such algorithms are terribly slow due their need of re-adjusting the PDF for each iteration, changing the probabilities every time that a new dot is placed on the canvas (normally by reducing the probability in the area where the dot was placed).

For this reason, and for these particular cases, the interpolation between the PDFs cannot be performed until all dots have been placed and the PDF is finalized. An additional downside of this approach is that it increases the algorithm’s computational complexity. Moreover, it is impossible to combine such techniques with approaches that lack a well-defined PDF such as Voronoi-based algorithms. To address this problem, we propose to use a discrete probability function (DPF) instead of a regular PDF, we detail next.

4.1 Discrete Probability Functions

One possibility of converting the PDF into a DPF is by the discretization of the canvas, that would give us a random variable X that specifies cells of the resulting grid instead of the continuous positions in the 2D space of the stippling canvas. We can go a step further, however, and directly consider the grid cells to contain the respective probability value of the cell receiving the next stipple dot or not. This representation has the advantage that we can model any purely deterministic stippling algorithm, guiding the probabilities for each cell of the grid as if it was a DPF. Moreover, with a sufficiently fine resolution of the grid, for deterministic algorithms we produce virtually the same result as before the discretization.

Let us assume such a sufficiently fine-grained grid placed over the stippled image, with N stipple points to be placed. We then mark any cell that has no stipple point location in it as a white cell (W), and any cell that has one or more stipple point locations in it as a black cell (B). White cells have no probability to be stippled, so we assign them a probability of 0%. Black cells (their total number is M with $M \leq N$) have some probability greater than 0% of being stippled, so they get a non-zero probability. With the probability of a cell $\Pr[B_i]$ we refer to the chance that the next stipple dot that is placed is assigned to the given cell. We can thus state that $\sum_{i=1}^M \Pr[B_i] = 1$ because the next stipple has a 100% probability to be placed into one of the remaining black cells. We can derive

the real initial probability of each black cell by sampling the PDF for non-deterministic algorithms, by using some heuristic (e. g., considering the amount of dots per cell), or simply by assigning the same probability to all black cells ($Pr[B_i] = 1/M$).

This way, a simple stochastic algorithm gives us a dot distribution virtually identical to that the original algorithm as follows:

- (1) Cast a single stipple dot to the grid. Randomly determine one black cell into which it is placed, considering the probabilities of each black cell.
- (2) For the chosen cell, reduce the probability of this black cell getting future stipples by subtracting the probability of the current stipple $1/N'$, N' being the number of remaining stipples to be placed including the current one, from the cell, and distribute this probability equally to the rest of the remaining black cells. This means that these remaining cells now have a higher probability to get the next stipple point than before.
- (3) If a black cell reaches 0% probability it turns into a white cell. If it would become negative, only subtract the remaining positive amount in the previous step and distribute it.
- (4) Repeat from step (1) until there is only one black cell (with probability equal to 1), the last stipple dot is placed into this cell and the algorithm ends.

This algorithm produces a stipple result equivalent and virtually identical to that of a deterministic technique—up to the chosen grid resolution and with the possibility that the number of dots in a given grid cell may differ occasionally due to the probabilistic approach. Moreover, if we had some way of computing the actual probability (as opposed to deriving it from an input stipple distribution), we would be able to maintain the exact same distribution as described by the PDF. Our goal, however, is not to run the algorithm. Instead, we want to derive a DPF for any given stippling technique, which is simply the grid with black and white cells and their associated discrete probabilities. We now use this DPF to be able to smoothly interpolate between two arbitrary dot distributions.

4.2 Linear Interpolation

Given two DPFs f and g corresponding to two different dot distributions, we can express the interpolation of both DPF as

$$\text{interpolated_DPF}(\alpha) = f \cdot (1 - \alpha) + g \cdot \alpha; \quad \alpha \in [0, 1]. \quad (2)$$

We just derived the input DPFs as a set of cells with probabilities (denoted Ω as the set of all cells in the grid) and we assume that both DPFs use the same resolution and alignment. To be able to compute Equation 2 we thus need to do a pair-wise interpolation of the respective probabilities in the cells in both DPFs.

We do this interpolation also in a probabilistic way, such that the chance that the cell value from f has influence is $1 - \alpha$ (let us call this event F) and the chance that the value from g takes effect is α (which we call event G). We model the event of a cell's probability taking effect using a uniform/rectangular distribution U_{cell} in the range $[0, 1]$, which we can compare to $1 - \alpha$ or to α and which we need to define for each cell in the grid. We are interested in dots being placed, so we want to know the union of events F and G, so

we can specify the interpolation of a given cell as

$$\begin{aligned} Pr[cell \in \Omega, \alpha \in [0, 1]] = \\ Pr \left[\left(f_{cell} \cap (U_{cell} \geq \alpha) \right) \cup \left(g_{cell} \cap (U_{cell} < \alpha) \right) \right]. \end{aligned} \quad (3)$$

In probability theory, given two different probability events A and B , and if A and B are independent, the probability of having the two events happen at the same time is $Pr(A \cap B) = Pr(A) \cdot Pr(B)$. The probability that events A or B occur is the probability of the union of A and B and can be written as $Pr(A \cup B) = Pr(A) + Pr(B) + Pr(A \cap B)$, with the last term $Pr(A \cap B)$ being 0 in our case. In an example where f_{cell} has a 50% probability of placing a dot and g_{cell} has 10% and for an α of 0.1, we would thus get an interpolated probability $Pr[cell, \alpha = 0.1] = 0.5 \cdot (1 - 0.1) + 0.1 \cdot 0.1 = 0.46$.

We can thus apply the interpolation of the grid of probabilities by means of Equation 3, leading to results as demonstrated in Figure 2. Figure 2a–d show the direct application for the interpolation of two DPFs each, with a very coarse grid size to demonstrate the effect. Figure 2a shows the interpolation between DPFs derived from a random distribution and a normal distribution, while Figure 2b interpolates from the DPF of the normal distribution to the DPF of a procedural figure. Notice that, due to the coarse grid size of the example, the single stipple of each cell is placed in the resulting regular grid pattern. Next, Figure 2c–d show the DPF-based interpolation for equivalent distributions at a high resolution (sampled from the original PDFs). For comparison, Figure 2e–f then demonstrate the PDF-based interpolation for the same distribution (i. e., before their discretization to DPF) by following Equation 1 and then sampling it directly (i. e., using a Monte Carlo algorithm [Arroyo et al. 2010]). As one can see, the resulting DPF-based interpolation and the PDF-based interpolation produce visually similar results, with the exception of the density of the coverage of the ring image that we explain below.

This last example shows that two PDFs can be interpolated. However, as argued at the beginning of this section we do not actually have a PDF for all input distributions that we could use, in particular when we want to represent an arbitrary input image such as in stippling. Here the best approach is to represent the input resolutions as DPFs and to interpolate between them, as shown for a toy example in Figure 3a: while the distribution on the right is derived from a PDF, the distribution on the left is derived from an input image and Figure 3a shows the DPF-based interpolation between them. The argument holds even more for the interpolation between two image-based distributions as shown in the realistic example in Figure 3b, in which we interpolate using DPFs between two different stochastic algorithms (a halftoning algorithm on the left and a weighted Voronoi algorithm [Secord 2002b] on the right).

One important property of the DPF-based interpolation, in contrast to the use of PDFs, is that it is computed in terms of cells not in term of samples. This means that, while in the case of the sampling of a PDF the number of samples (or stipple points) can remain the same during the interpolation, in DPF-based interpolation we cannot guarantee that the number of samples remains constant during the interpolation process. We even cannot guarantee a constant sample count if the two algorithms to be interpolated each have the same amount of samples to start. Such situation is rare when dealing

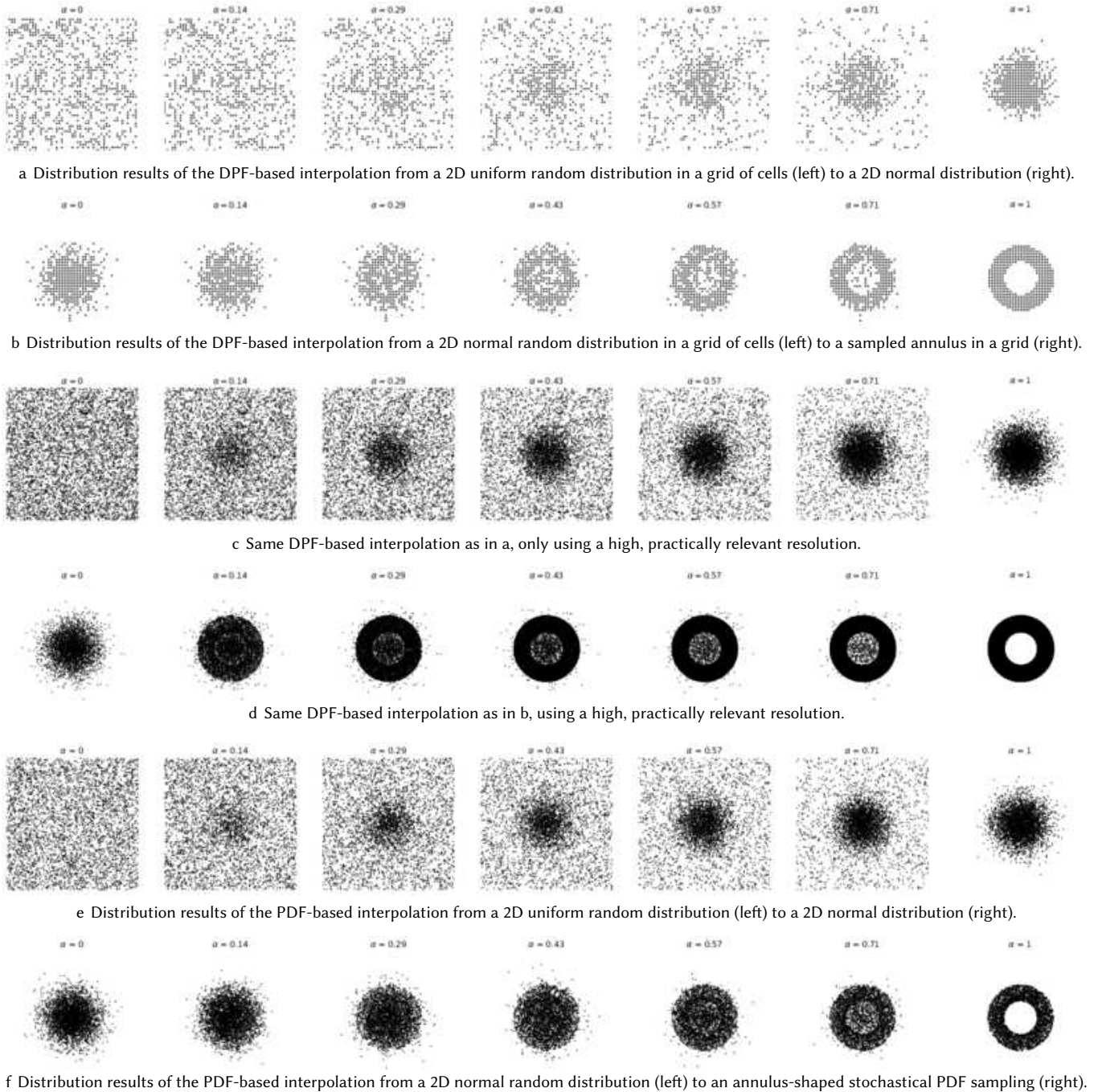


Fig. 2. Interpolation examples of two couples of DPFs, first in a–b at low and later in c–d at high resolution; e–f interpolation of their PDF counterparts.

with real algorithms, however, so this constraint has little practical implications such as when applying the DPF-based interpolation to digital stippling. Moreover, PDF sampling cannot guarantee that we get completely black areas are covered by the samples as shown in the last image on the right of Figure 2f: we instead get noisy images. The grid of the DPF, in contrast, forces us to always sample all the

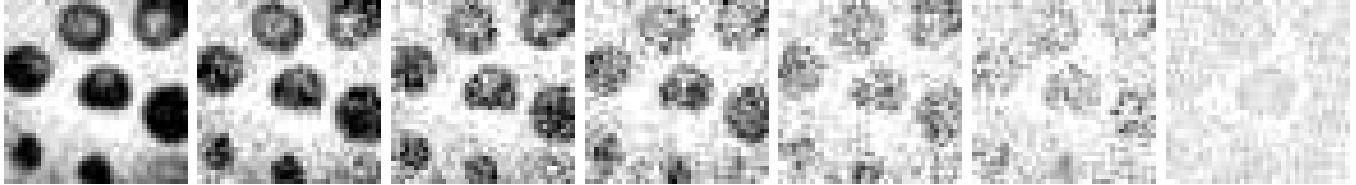
cells of the grid, which in turn allows us to get close to the original as shown in the last image on the right of Figure 2d.

4.3 DPF Interpolation based on a Distance Field

The problem with the linear interpolation of DPFs (or, similarly, PDFs) as discussed thus far is that the interpolation is applied glob-



a On the left, a sampled annulus in a grid; on the right, a sampled logo in a grid; in between, the distributions result of the interpolation of their DPFs.



b A more realistic example of the interpolation of two DPFs with a high-resolution grid, from $\alpha = 0$ on the left to $\alpha = 1$ on the right, with an increment of $1/7$ units per step. The left is a halftoned detail section of the frog in Figure 1, the right is the same picture after applying weighted Voronoi stippling.

Fig. 3. Examples of DPF interpolation for which PDF-based interpolation is unfeasible.

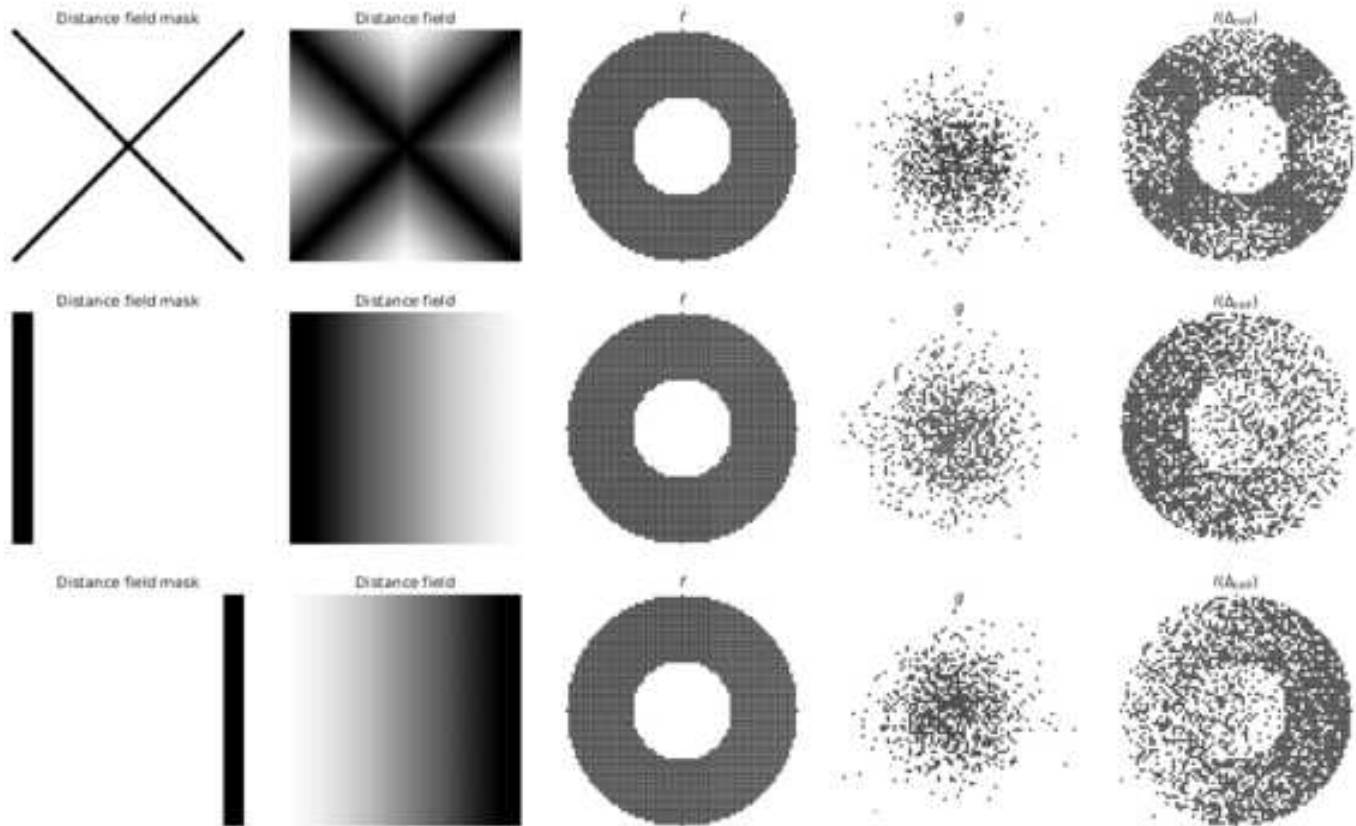


Fig. 4. Different masks, their respective distance field, two distributions (f and g), and their interpolation based on the distance fields.

ally for the entire canvas. We cannot yet control the interpolation depending on the original picture’s composition to combine different distributions as we motivated in Section 1. We thus now define a new function Δ that controls the interpolation based on the 2D

distance between a specific cell and a given subset of cells as

$$\Delta(x, \partial\Omega) := \frac{\inf_{y \in \partial\Omega} d(x, y)}{\max(\Omega)} ; x \in \Omega . \quad (4)$$

Here, Ω represents the complete space, i. e., the set of all the cells in the grid. This set can be considered to be a partially ordered set

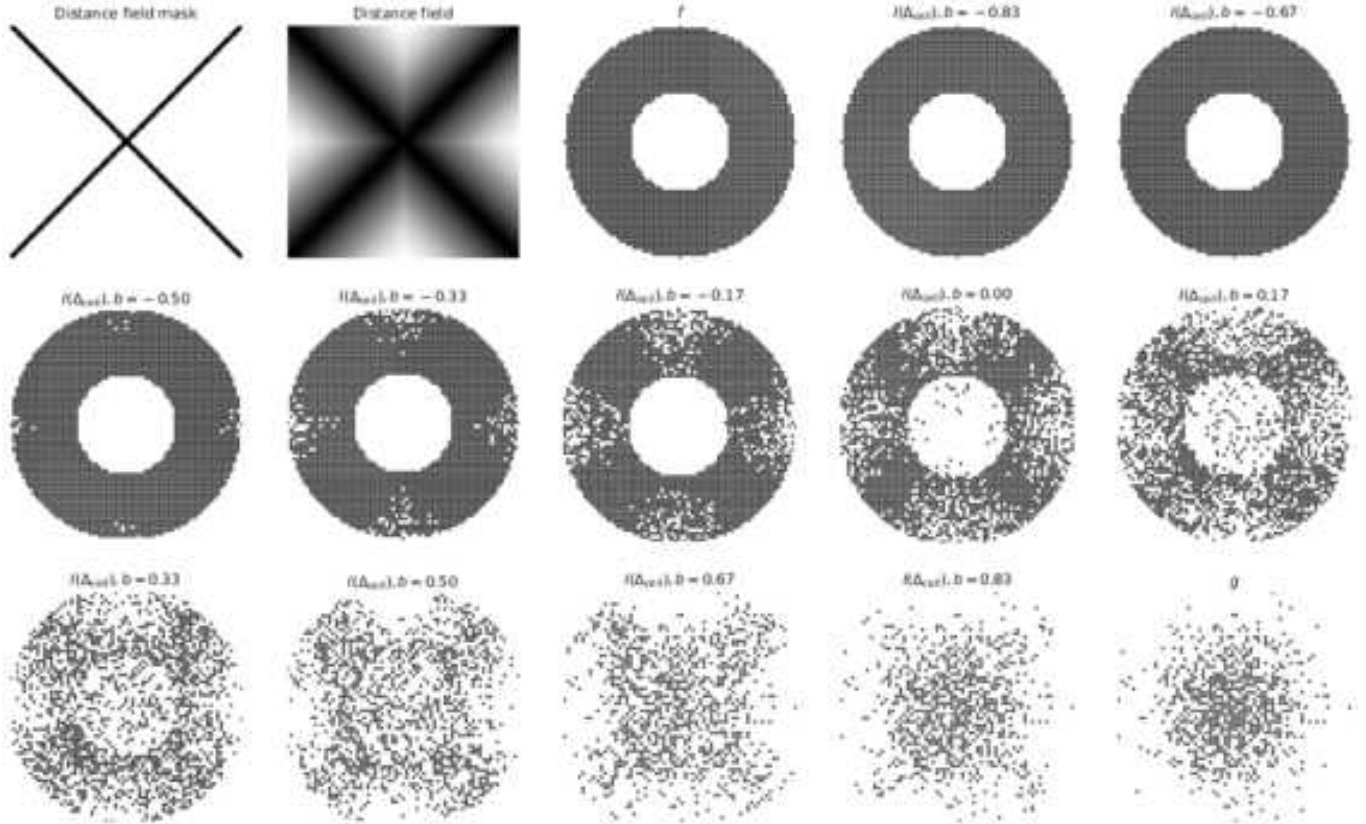


Fig. 5. Different masks, their respective distance field, two distributions (f and g), and their interpolation based on the distance fields.

because we can order the cells based on some arbitrary function such as the distance to a given, selected cell x . We can thus use the infimum to get the smallest value of possible distances $d(x, y)$ between x and any cell in a subset of cells $\partial\Omega$. The distance $d(x, y)$ could use any distance metric, we use a simple Euclidean distance. While $\partial\Omega$ could be equal to Ω such that it contains all its cells, in practice it will be just a part of the cells. The reason is that our goal is to construct a distance field from some mask that could guide the interpolation of the two distributions. We thus construct a mask by marking some of the cells of Ω as black cells, constructing the subset $\partial\Omega$. With the Euclidean distance $d(x, y)$, $\inf_{y \in \partial\Omega} d(x, y)$ gives us the distance from x to the closest black cell. We then normalize the result based on the maximum distance in Ω , namely $\max(\Omega)$. Overall, Δ_{cell} thus calculates the normalized distance field from each cell x to the closest black cell in the grid (Ω).

Figure 4 shows the effect of three different masks (i. e., different subsets $\partial\Omega \subset \Omega$) and their resulting distance fields on the interpolation between a uniform random distribution (g) and a deterministic algorithm drawing a 2D torus (f). In these examples the masks are independent from the used point distributions. Ultimately, however, we want to control the interpolation based on features in the original image, which also led to the two point distributions (in the case of digital stippling). We thus make the interpolated distribution I

depend on Δ instead of α and rewrite Equation 2 as

$$I(\Delta_{cell}) = f_{cell} \cdot (1 - \Delta_{cell}) + g_{cell} \cdot \Delta_{cell}.$$

In addition to this distance-based interpolation we also want to provide users with control for when f or g become dominant. We thus add a bias $b \in [-1, 1]$ to Δ_{cell} , so Equation 2 finally becomes

$$I(\Delta_{cell}, b) = f_{cell} \cdot (1 - \Delta_{cell} - b) + g_{cell} \cdot (\Delta_{cell} + b). \quad (5)$$

Figure 5 shows the effect of this bias b on the interpolation between a uniform random distribution (g) and a deterministic algorithm drawing a 2D torus (f). When $b = 0$ we get function f , whereas when $b = 1$ the returned values correspond to g .

So three functions are involved in the process of the interpolation. On one hand we have the two DPFs that can be implemented as simple masks (for deterministic algorithms these mask are binary). On the other hand we have the distance field mask. This is another binary mask that represents how the interpolation is performed. As noted before, one advantage of using DPFs instead of PDFs is that we can directly use the result of any stippling algorithm. We can thus take advantage of the parallelism of modern GPUs due to the cells in the masks being independent from each other. Another advantage is the simplicity of the computation of the distance field mask. The problem with the use of DPFs is the regularity of the resulting cells, but for that we can use some post-processing in the render stage, as we explain later.

5 ARTIFACT-CONTROLLED DIGITAL STIPPLING

As we had discussed, many existing techniques are successful in shading areas in images. While some early techniques based CVDs created unwanted linear features, most recent techniques produce artifact-free distributions. With our new distribution interpolation technique we could thus select any of these techniques and use it to shade larger regions, and combine it with another technique that can represent borders and features well. Next, however, we discuss specific choices for the algorithms and show how they efficiently facilitate artifact-controlled stippling.

5.1 Border Stippling

We begin with the stippling of borders, for which we could now use a dedicated structure-aware stippling technique (e. g., by Mould [2007] or Li and Mould [2011, 2017]). With our new ability to combine any two dot distributions, however, we can use the easier early stippling techniques, which are simpler to implement. Secord’s WCVD [2002b], for instance, produces results with chained stipples even in shaded areas, but as long as we only use the feature lines it produces for which we do want stipple chaining this does not pose a problem.

The key insight for the stippling approach we describe next is that, using our previously described interpolation technique for dot distributions, we would first express both as DPFs, assuming that each resulting black pixel represents only one dot of the stippling result. For some of them, including Secord’s WCVD [2002b], this process involves taking an image as an input, doing some image processing such as contrast adjustment or edge detection, and then running the actual dot distribution process, before extracting the DPF from the generated dot distribution. Second, we would interpolate the two generated DPFs as described in Section 4, and then place stipple dots for the interpolated DPF as described in Section 5.4. It turns out, however, that we can optimize the conversion from input image to filtered image to analytic dot positions to DPF (which is also essentially an image) by using the filtered image, after some additional (image) processing, directly as the input to our interpolation. Next we thus explain this process for generating the edge-representing DPF for our hybrid stippling process, before we discuss the shading of areas and the mixing of both approaches using a distance field.

One of our main goals is that we want those stipple dots that represent single lines in an image or that are borders of a stippled region to be aligned. Our first step is thus to compute an adjusted input image that emphasizes these edges. For this purpose we can employ established edge detection filters (e. g., by Canny [1986], by Sobel and Feldman [2014], by Prewitt [1970], the Laplacian of Gaussians—LoG [Marr 1982], or the Difference of Gaussians—DoG [Marr and Hildreth 1980]), and past NPR work [Hertzmann 1999; Isenberg et al. 2003] has discussed that some of these filters need to be combined to better cover those edges typically used in hand-drawn line images. Nonetheless, these filters are not always able to cover all lines a human artist would draw [Cole et al. 2008, 2009], and they also often require difficult fine-tuning or parameters to the chosen input image. In recent years, researchers have tried to address these problems by using deep neural networks (DNNs) (e. g., [Chen et al. 2015; Kokkinos 2015; Shen et al. 2015; Xie and Tu 2017]). Although these solutions can detect edges successfully, they



Fig. 6. Possible zig-zag pixel distribution due to a 2-pixel-wide edge.

need many training images and expensive human input to annotate the valid edges of the images. In addition, they require powerful hardware provide interactive response times. In this work we thus restrict ourselves to using traditional image filters as they are quite successful and because we combine them with stippled areas; yet they could be easily replaced by another technique in the future.

One essential prerequisite is to obtain edges of one-pixel thickness. We need this constraint to align stipple points later-on along the edge. For example, lines with a width of two pixels may end up being stippled in a zig-zag fashion as shown in Figure 6, for an average stipple distance of $\sqrt{2}$ pixels. For these reasons we use Canny’s [1986] filter for the examples in the remainder of our work to guide the placement of stipples along one-dimensional feature edge paths.

We thus now have a filtered input image for the first of the stippling techniques (Secord’s [2002b] WCVD stippling) that we use in our hybrid approach. As indicated above, similar to the binary image we ultimately need in our DPF-based interpolation of stippling distributions, this input image also essentially is a binary image, containing roughly the same information: black pixels that indicate locations for placing (much larger) stipple dots after the interpolation. The use of WCVD stippling just to then extract the DPF from the resulting distribution thus seems like an unnecessary detour, even with a fast GPU-based WCVD implementation.

In our approach for producing the edge DPF we thus do not actually run Secord’s WCVD but use image processing techniques to generate a similar result, one that also emphasizes edges and ensures an even distribution of stipple dots along them. Nonetheless, the result of first stippling and then extracting the DPF is not identical to the edge image we used as the input: the DPF image would not contain a consecutive series of pixels for every edge, in contrast to the filtered edge image. We thus need to add a controllable amount of space in-between the aligned stipples, a gap that would normally be ensured by the WCVD stippling, which limits the overall number of stipple dots in the image and thus also the number of black pixels that represent an edge in the DPF. Yet we can use a similar ink-constraining process and apply it directly to the edge image—in a way a simplified 1D case of WCVD stippling to distribute the remaining pixels evenly along edges.

For this purpose we determine all edge pixels and treat them as paths. Starting with a given black pixel, we find all its black neighbors in the 8-neighborhood. Given the direction that we came from the previous pixel, we then select that black pixel among the current pixel’s neighbors that best continues the path (or a random neighbor if we started a new path). We continue this stepping along pixel paths until a path cannot be continued anymore (i. e., there are no more black neighbors). Once this is completed, we start a new path tract by selecting a new random pixel that was not visited yet—until all pixels have been visited. To best emphasize long, outline

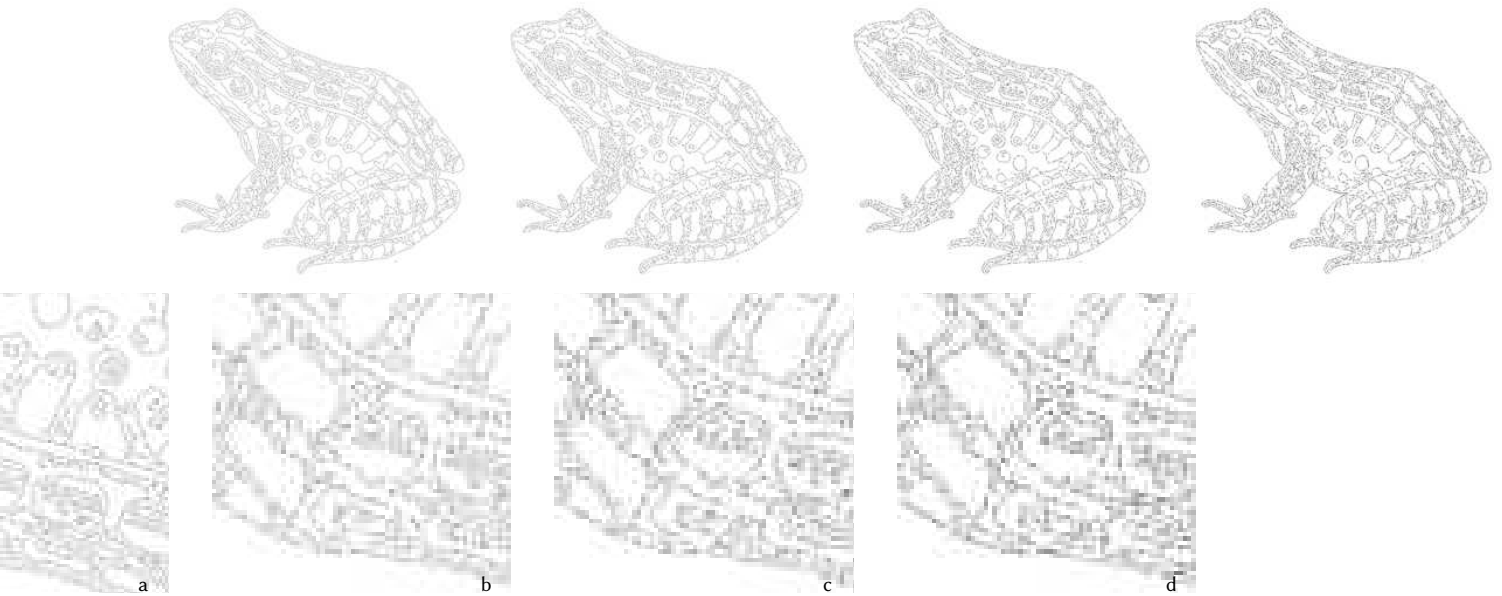


Fig. 7. Adding noise (full results and details), using $d_0 = 3.5$ pixels: a 0%, b +/-16.5%, c +/-33%, d +/-50%. Note that we clamp d to be at least 1 pixel.



Fig. 8. Corner problem produced by Canny's filter.

strokes, we begin by searching the image from top-to-bottom and left-to-right until we find the first black pixel—our initial starting pixel. We also use the heuristic of using the left-most pixel (w.r.t. our incoming direction) in the 8-neighborhood as we select the next pixel to ensure walking around shapes first. We also tested the addition of a backward search after the forward search to follow paths in both directions, but the results were visually equivalent to only doing a forward search, so we kept this simpler approach.

The process as we have described it so far simply yields all black pixels, and the result is identical to the edge buffer. To simulate the effect of WCVD stippling, we introduce sections of white pixels between the black ones. We thus only output a new pixel once we have covered a user-specified distance d along the path (counting $\Delta_d = 1$ for horizontal or vertical steps and $\Delta_d = \sqrt{2}$ for diagonal steps). To avoid regular patterns, we add noise to the distance. While we could have used a normally distributed noise offset around an average distance d_0 , our tests have shown that we can produce visually satisfying results by adding noise to d_0 in form of a random offset from $[-d_n, d_n]$. We demonstrate the effect of this noise being added to d_0 in Figure 7, for different amounts of noise.

One issue with this process is that the Canny filter, which we use, produces combinations of three black pixels (see Figure 8), which resembles a zig-zag line in the final image instead of a straight one or resembles a block. We thus remove these cases prior to stepping along the pixel paths. We show a more realistic example in Figure 9.

For an example edge input as the one in Figure 9a, this process then yields results as we show in Figure 10c. Compared to the WCVD

stippling result for the same input shown in Figure 10a, we argue that the result is visually equivalent. Figures 10d and 10b show detail sections for this comparison, respectively.

5.2 Stippling shaded regions

In shaded areas, it is well established that CVD and WCVD are counterproductive due to the regularity of these distributions. As we discussed in Section 2, other approaches produce similar blue noise distributions to avoid visual patterns and artifacts. To achieve our goal of realism in digital stippling, however, we base our solution on Martín et al.'s [2010; 2011] use of scanned grayscale dots and halftoning-based placement because their process considers the target image's physical size. The packing factor and the noise added to dot locations in this process produce good results for shaded areas (i. e., areas with a low gradient). Moreover, it is fast to compute because its dot distribution is based on a halftoned image [Ostromoukhov 2001], which also has blue noise properties, and its results resemble hand-made distributions according to the statistical analysis by Maciejewski et al. [2008].

The main problem of this approach is that it uses a grid to distribute points and adds random noise to remove the regularity. While this leads to good results in shaded regions and makes stippled edges appear fuzzy and unclear, with our distribution interpolation we can now combine it with the edge stippling discussed before. We thus only need to derive the needed mixing function α to control the interpolation, as we describe next.

5.3 Mixing functions

Now that we have the two distributions in place, we need to define the specific function that controls the interpolation between them. This function needs to encode our initial goal of using the edge-emphasizing stippling from Section 5.1 for the edges, and the

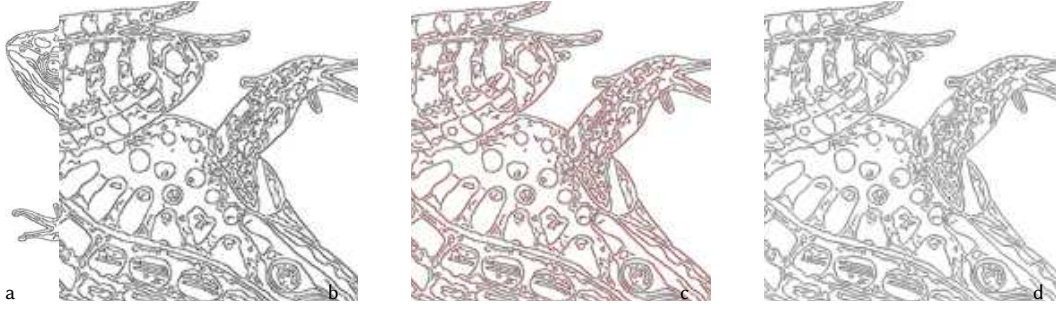


Fig. 9. Problems of the Canny filter and how we address them: a result of the Canny filter, b detail section showing the problems, c detail section with the problematic pixels in red, and d detail showing the corrected version.



Fig. 10. Comparison of WCVD stippling with our pixel-based heuristic: a WCVD, b detail from a, c our heuristic, d detail from c.

distribution mentioned in Section 5.2 for representing filled areas. In Equation 2 we initially controlled the interpolation by means of a parameter α , but later improved upon this concept in Equation 5 to a interpolation based on a distance field Δ and with a bias b .

The distance field Δ intuitively appears to be well suited to achieve our goal, provided that we use the initial edge Canny-filtered image generated in Section 5.1 as the distance mask $\partial\Omega$. We may want to, however, achieve a number of different goals as follows:

- show both border and the area stipples at the same time, with different densities depending on the distance from the edge,
- leave some white space around the border to enhance it,
- produce an inversion effect in which edges are represented by the lack of stipples, etc.

Our current linear computation of the distance Δ is too inflexible to achieve these goals. We thus update Equation 5 that we used for the interpolation to also employ a general function Γ that maps the distance field, the domain, to the same codomain but with a different shape to fulfill the desired goal:

$$I(\Delta_{cell}, b) = f_{cell} \cdot (1 - \Gamma(\Delta_{cell}) - b) + g_{cell} \cdot (\Gamma(\Delta_{cell}) + b). \quad (6)$$

By default, Γ is defined as a linear function ($\Gamma(cell) = \Delta_{cell}$). The possibilities of the Γ function can be shown with an example: If we want to control, for instance, the density of pixels for producing the combination of the borders and the area at the same time and to produce a band surrounding the borders, we could use this Γ

function, given the parameters L_1 and L_2 with $L_1, L_2 \in [0, 1]$:

$$\Gamma(\Delta_{cell}) = \begin{cases} 0 & : \Delta_{cell} \leq L_1 \\ \frac{\Delta_{cell} - L_1}{L_2 - L_1} & : L_1 \leq \Delta_{cell} \leq L_2 \\ 1 & : \Delta_{cell} > L_2 \end{cases}. \quad (7)$$

5.4 Stipple Rendering

We can now render the resulting interpolated distribution, which is represented as grid of black and white pixels and with black pixels to receive dots. Following our stochastic algorithm from Section 4.1 we can then place dots into black pixel regions, until we treated all black pixels. In contrast to recent approaches which freely rearrange stipple dots based on an initial positions (e. g., [Balzer et al. 2009]), we thus have to place dots based on the black pixels and consider the dot sizes and shapes. For the realistic capture of dot distributions such as in traditional stippling, in particular, the grid of the DPF has to be highly detailed, while a placed dot is ultimately much larger than a grid pixel.

As we based the stippling of shaded regions on Martín et al.'s [2010; 2011] halftoning-based dot placement, we also use it for the overall rendering of stipple dots. This allows us to take care of the target resolution of the output, the physical size of the stipple points, and overlapping of stipple dots. Martín et al. removed the regularity of the halftoning-based dot placement by adding noise, which worked great for areas but made their stippled linear features look fuzzy. In our new method, we resolve this problem by controlling the added noise individually for areas and linear features. In most cases, the added noise to linear features is zero, but it depends on

needs of the artist and the configuration of the pixels.

We use the same approach to control the range of sizes for dots in a discrete or continuous way. The discrete option allows us to simulate the use of real technical pens, which have a limited set of tip sizes and allow us to use scanned dots (i. e., textures). Both discrete and continuous control can be used to produce vector results in SVG format. The selection of the size for each individual pixel can be computed in a uniform random way, or by modulating it for the tone of the corresponding pixel in the original image. This control provides us with a great level of flexibility in the appearance of the results, as we discuss and showcase next.

6 RESULTS

To be able to illustrate the spectrum of possibilities with our new interpolated point distributions (henceforth IPD), we first compare them with established digital stippling approaches and, second, discuss examples to showcase the effect of different parametrization.

6.1 Comparison

For the comparison we selected three techniques from the literature that placed particular emphasis on either representing features or areas with digital stippling. First, we selected Secord's [2002b] weighted centroidal Voronoi stippling (henceforth WVS) because it is able to capture linear features well, despite creating line artifacts in areas due to being based on centroidal Voronoi diagrams. Second, we compare with Li and Mould's [2011; 2017] structure-preserving stippling (henceforth SPS) which placed particular emphasis on linear structures. Third, we compare with Martin et al.'s [2010; 2011] example-based grayscale stippling (henceforth EBG) because it managed to avoid linear structures in areas and came close to artistic examples, yet at the expense of only being able to produce fuzzy linear structures. These techniques differ from each other as follows:

Color: WVS and SPS are B&W only processes. EBG is a grayscale process that is inspired by the process of ink accumulation.

Dot shape: WVS and SPS use circles. EBG uses scanned dots, i. e., irregular shapes recorded as textures.

Dot size: WVS can use dots with a constant size, but can also modulate the dot size depending on the Voronoi region's area and the tone of the input image [Secord 2002a]. This modulation maximizes the perceived effect of each dot. SPS produces dots of constant size. EBG is based on scanning original hand-made stippling artwork and extracting individual dots, thus uses random stipple sizes within a given range.

Output size: EBG is a raster method with the goal of producing realistic stipple output for a given physical output size and pixel resolution, using matching scans of dots from stipple artwork. WVS and SPS are vector methods, their dot size can be controlled independently and continuously.

We selected four target images for our tests: *City* and *Headlamp* from Mould and Rosin's [2017] benchmark set: these showcase straight and curved linear features. We also use *Lenna* and Secord's [2002b] *Plant* because they were used in the past for studying structure-preserving stippling. For a fair comparison between the resolution-dependent EBG with the resolution-independent WVS, SPS, and IPD techniques we set a physical target equivalent of A5

output size at 300 ppi resolution. Based on these constraints, we used the original EBG code to produce stipple images, noting the stipple count. We then used Secord's [2002b] original implementation of WVS to produce results with the same target stipple counts (Figures 20 and 21), and with the help of Li and Mould's original implementation also obtained stippled versions for SPS (Figure 22), again with the same target stipple count per image.¹ For IPD we used the combination of WVD with EBG described in Section 5.

With all four techniques, we created several versions of the four benchmark images shown in Figures 11–13. To save space we do not show the whole A5 images (find them in the additional material in Figures 23–34), but only use a representative section. Specifically, we show vector images with constant, small, black, and circular dots in Figure 11 to show the distributions without any dot overlap. In Figure 12 we still use black circles and vector output, but modulate the dot size by the tone color of the corresponding pixel in the original image such that darker tones produce bigger and brighter tones produce smaller dots. Finally, Figure 13 shows a raster output with random dot sizes and scanned dots, with the actual dot size range being derived for all images using the EBG approach.

By using small circular dots with constant sizes, Figure 11 allows us to analyze the spatial character of the different dot distributions. We can observe, e. g., that SPS uses a regular grid alignment of the dots somewhat reminiscent of halftoning, which is not visible in its actual results as then the larger dot size leads to overlapping stippling (Figures 12 and 13). WVS, in contrast, is not bound by a grid but exhibits chain artifacts, also in normally dense regions where dot size attenuation normally would hide this fact (Figure 12). Nonetheless, the chain artifacts also play to WVS's advantage for stipples that should line up, such as the example in Figure 11b. EBG shows less chain artifacts in areas, but its edges and borders are not as clear due to the halftoning-based placement and the added randomness. EBG's halftoning grid itself is not visible because, for Figure 11, we used the EBG process as if with large dots and then used the final dot positions, without rounding them. Finally, we see that IPD combines the overall good distribution of EBG in areas with better edges from WVS, evident, e. g., in Figure 11o.

In a next comparison, we added variable, more realistic dot sizes that are also modulated based on the source image's brightness [Secord 2002a] (Figure 12). We computed the range of the actual dot sizes according to the considerations by EBG, which aimed to simulate realistic dot sizes. This change of dot sizes naturally makes all images much darker. The modulated dot sizes also make some artifacts such as the grid arrangement of SPS less apparent, but it does not completely hide them. Rather, in SPS it leads to completely black regions, while for WVS, EBG, and IPD we see more realistic stipple clusters forming. Note that this is not the same modulation as used by Secord [2002a] (and which we show in Figure 21); Secord did not only rely on the local image graylevel for deciding the modulation and also uses a range of dot sizes for a given graylevel. For the size of the line features in IPD we used a dedicated way to specify the dot size because we could not apply modulation—for linear features extracted by some edge detection mechanism there is no meaningful brightness to use. To avoid having to use a constant

¹*City*: 171,172, *Headlight*: 137,327, *Lenna*: 1,134,741, and Secord's *Plant*: 149,327.

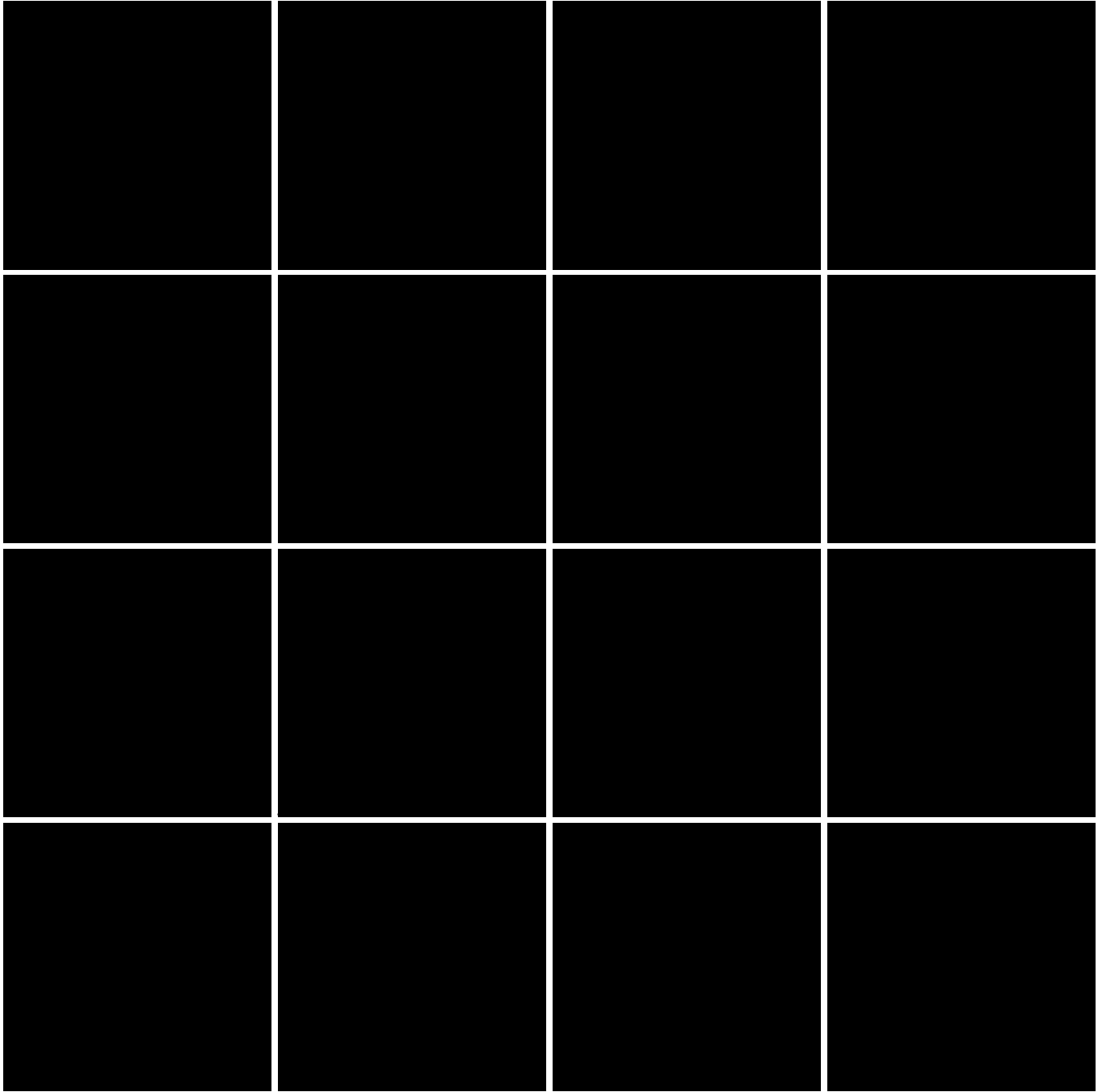


Fig. 11. Constant size dots, radius 0.5, B&W. First row WVS, second row SPS, third row EBG, and fourth row IPD.

size, we instead chose the mean value of the dot size range used for the areas, with an additional random scaling of $\pm 25\%$ to provide some variability. In comparing the line features, we see that they come out well for WVS and SPS, and, in contrast to EBG, also IPD shows much better aligned features. Unfortunately, however, IPD has problems with single lines of dots such as the highlight on the car headlamp as can be seen in Figure 12n. The root cause for these

double lines is our use of the Canny filter to control the interpolation. We discuss this effect and how to address it below in Section 6.2.

Finally, Figure 13 is based on the same modulated dot sizes as Figure 12, but with realistic stipple dot textures applied. For this figure, we used the normal scale-dependent process for EBG, and for WVS, SPS, and IPD we placed dot textures to the vector positions of Figure 12, with rounding of the resulting scale-dependent texels

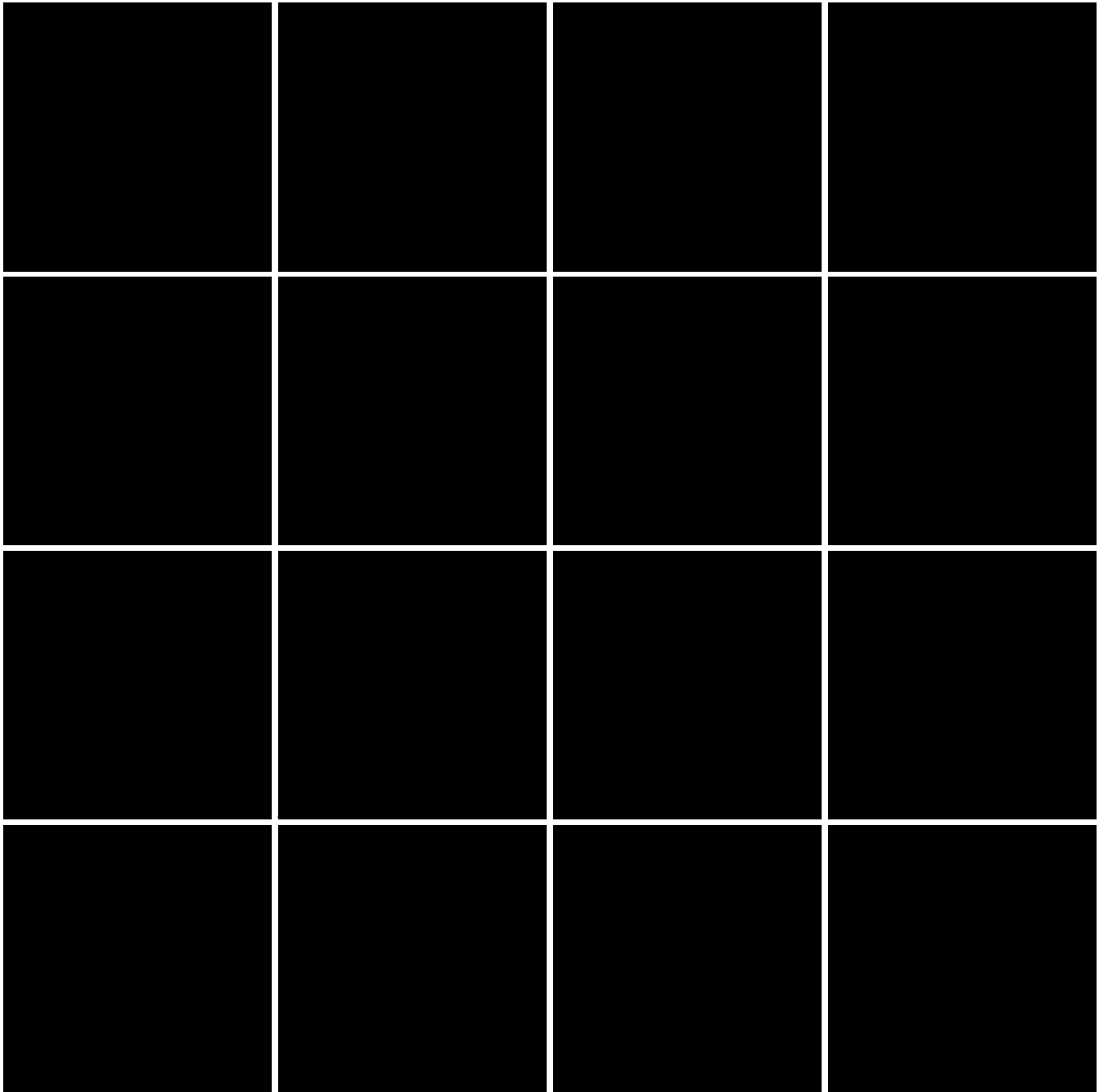


Fig. 12. Range between 2 and 4, continuous, modulated by tone, B&W. First row WVS, second row SPS, third row EBG, and fourth row IPD.

to avoid texture resampling. We enlarged the modulated dot sizes of Figure 12 by doubling their size to accommodate the fact that a texture of an irregularly shaped dot is not perceived as a black circle of the same size, but smaller due to the intensity change toward the center of the dot and the overall irregular shape.² As we can see

²We determined the scaling factor of $2\times$ empirically. In fact, the sizes of the dots in Figure 13 are mandated by the EBG process. In practice we thus adjusted the dot sizes

from the comparison of the images in Figure 13, the irregular shapes of the real dots even further hide the regularity artifacts of SPS, yet the areas of SPS still appear very dark and the images appear to have less tonal variety than the other techniques. Moreover, we see that the interpolation of IPD allowed us to create edges of similar

in Figure 12 to be half of the sizes of the textured EBG dots in Figure 13.

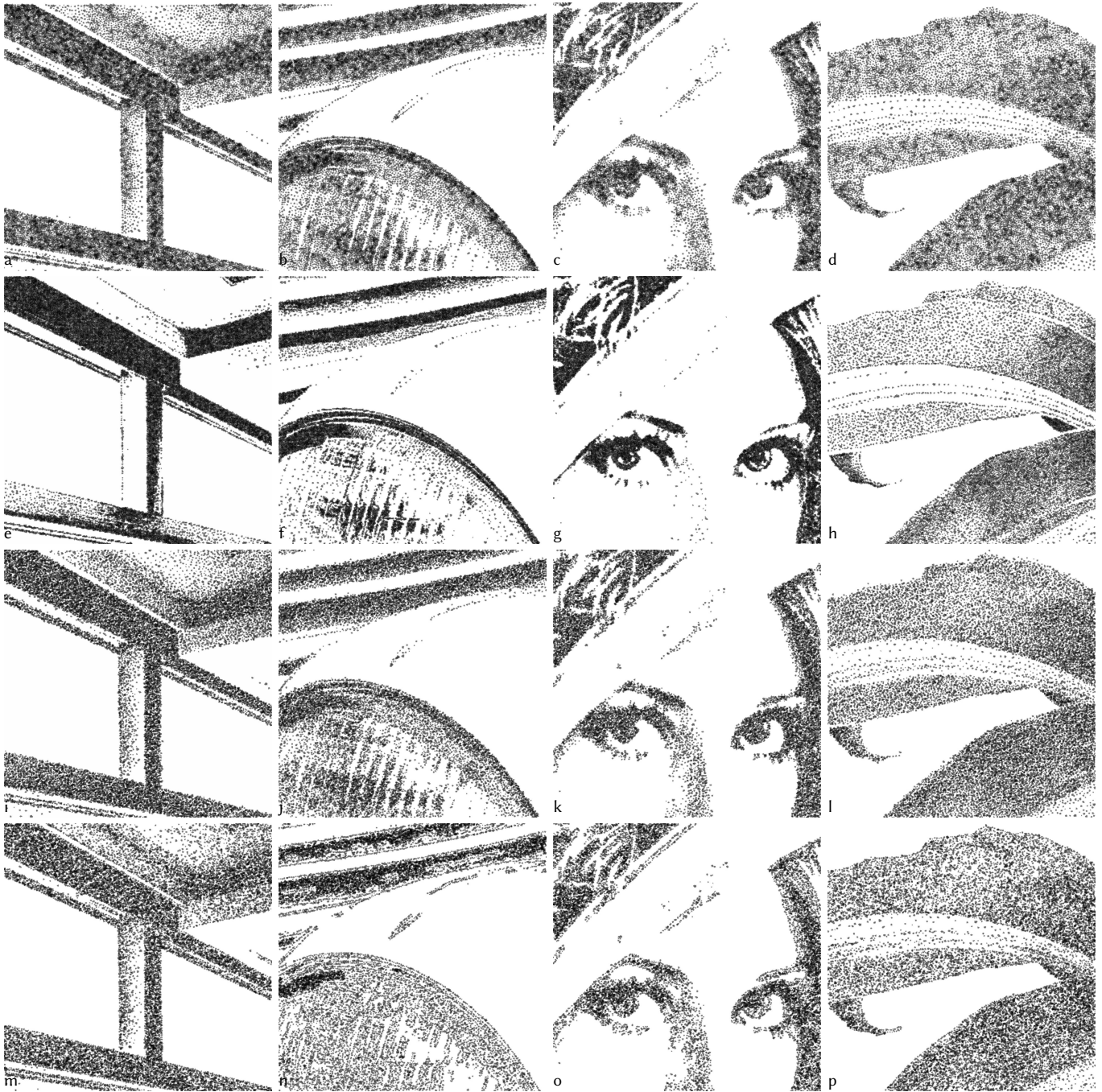


Fig. 13. Dot size range of 4–8; discrete, random, scanned dots at 1200 ppi. First row WVS, second row SPS, third row EBG, and fourth row IPD.

quality as those used in SPS—both for isolated lines and for edges of areas. This result is visible, in particular, in Figure 13p vs. h vs. l. Some specific edges, however, are still best portrayed by WVS, such as the lines on top of the headlight of the car—neither SPS nor IPD capture them quite as well as WVS, yet both are about the same and much better than WVS. With respect to the area portrayal, WVS shows some unnatural clusters, and EBG and IPD seem both better

than the other two—with IPD showing some more tonal variation than EBG. The reason for this effect is that IPD distributes slightly differently and places more points in borders than EBG.

These examples show that our new IPD approach indeed combines high-quality stippled areas with a good reproduction of edges, addressing the problem of EBG’s fuzzy lines. As noted before, however, the issue of isolated lines remains, as we discuss next. Moreover,

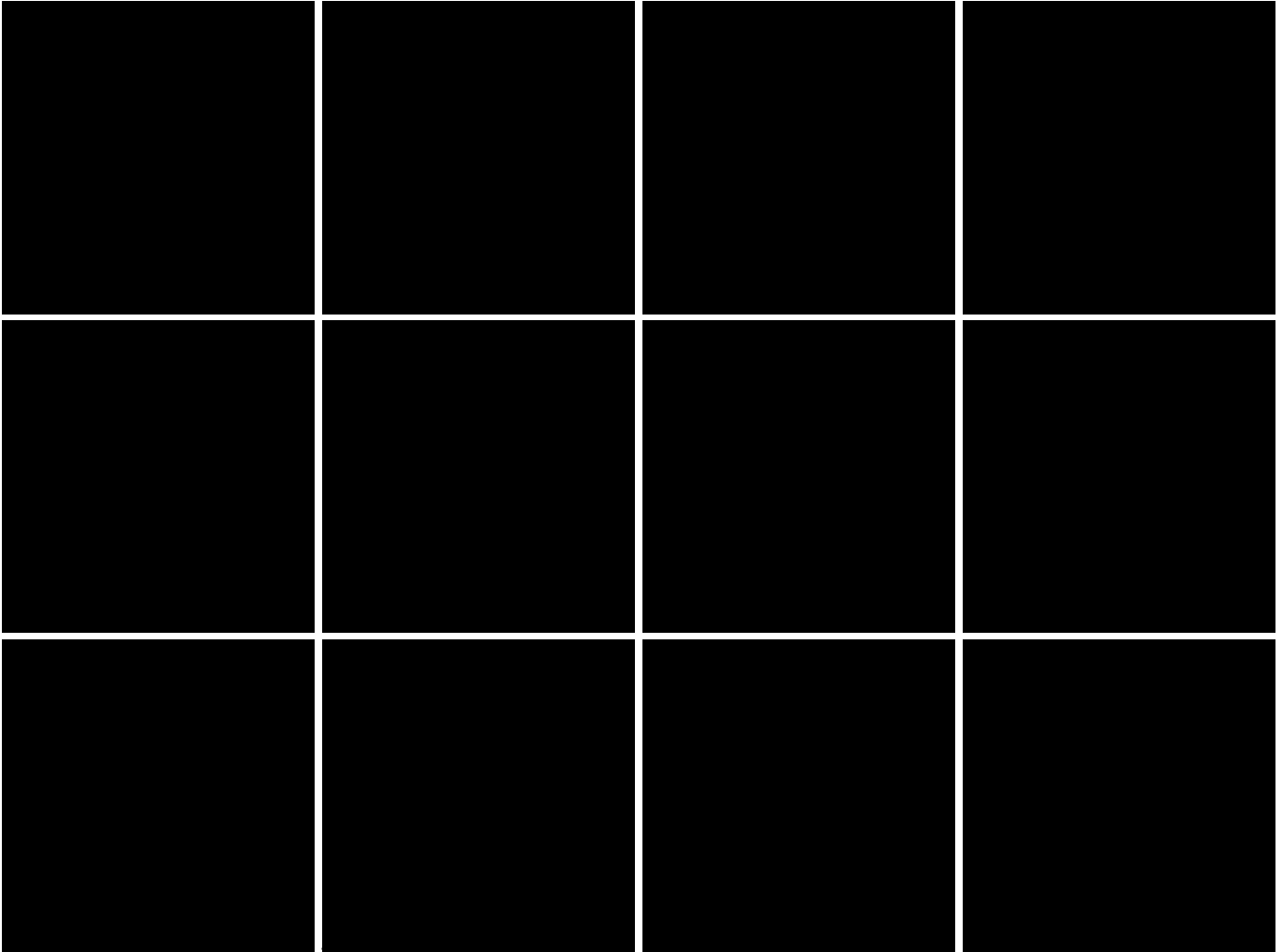


Fig. 14. Comparison of the use of different image filters for the IPD method. First row: Canny, second row: DoG, and third row: LoG. Also, we specifically highlighted the edge dots that originated from the WVD distribution. See the large versions in Figures 35–37 in the additional material.

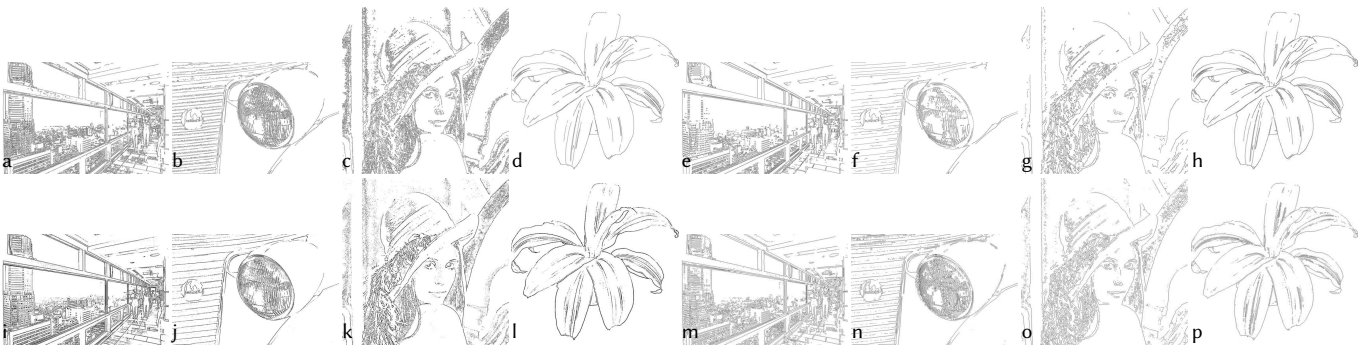


Fig. 15. Edge image inputs for the results in Figures 11–14. a–d: Canny-based edges for Figures 11–13; e–h: edges from filtered input with Canny for Figures 14a–d; i–l: edges from filtered input with DoG for Figures 14e–h; m–p: edges from filtered input with LoG for Figures 14i–l.

our comparisons used *two specific* source techniques with a *specific parametrization*, while IPD can combine the results from *any two*

techniques and with a *flexible parametrization*, as we also show next.

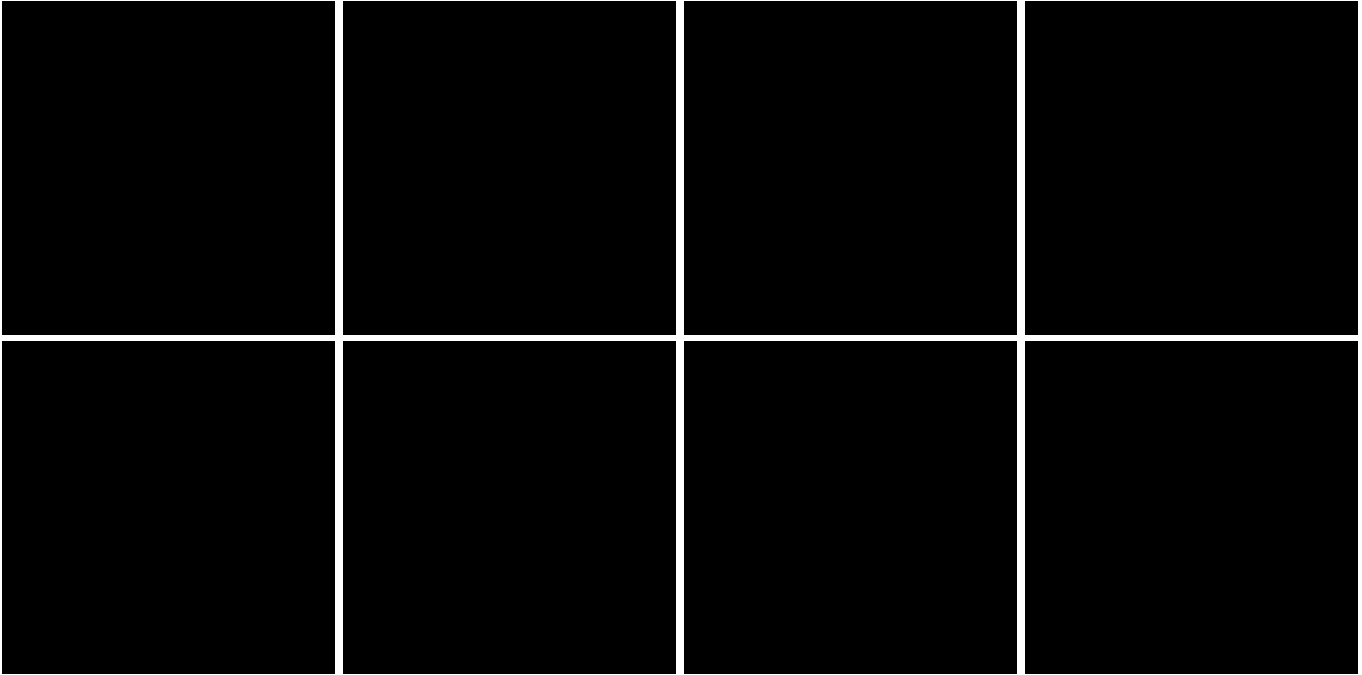


Fig. 16. Interpolation effects that can be achieved with our IPD method. See the large versions in Figures 38–39 in the additional material.

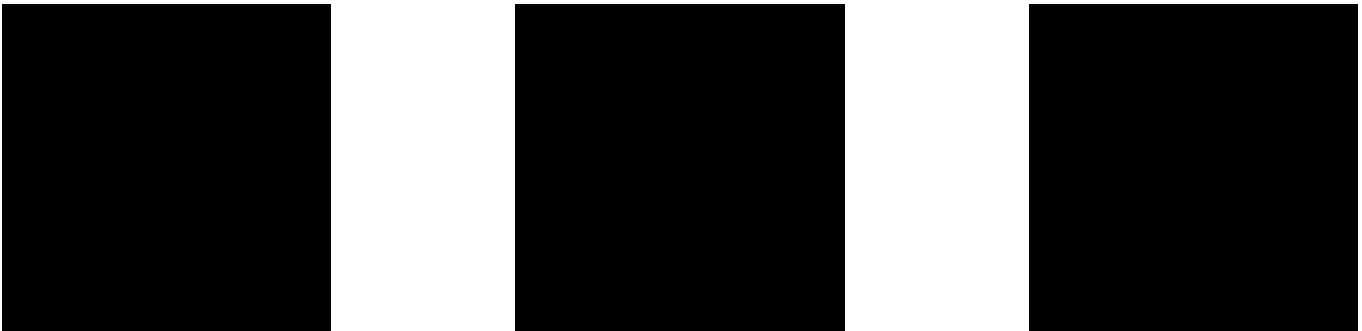


Fig. 17. Examples for effects that make use of masks to adjust the distance function Δ , to control the use of stipples from the input distributions.

6.2 Alternative results

For other results we can drop the limitations that we applied to the previous comparison, such as the fixed number of stipple dots used in Figures 11–13. These led to certain artifacts for IPD such as the duplicated rows of dots for isolated lines and the removal of dots around the linear isolated features of the plant example, which may or may not be desired. We thus first explore the use of alternative edge detection filters for controlling the interpolation as well as use different parameterizations for the dots, depending on from which of the two input distributions they arose.

Figure 14 shows the result for the use of Canny, Difference of Gaussian (DoG), and Laplace of Gaussian (LoG), with the additional use of filters to address the issue of double edges for single lines. To better highlight the properties of the generated edge stipples—and to produce an alternative visual result—, in Figure 14 we use

a constant small dot size of 2 for the stipples generated for areas, while showing edge stipples larger with size 4. For this purpose we manually used a combination of Gaussian blur as well as contrast and brightness adjustments, before applying the respective edge filters. We show the resulting edge images in Figures 15e–15p. With this process we can see that we can remove the double edges even for the Canny filter as shown in Figures 14a–d. The DoG filter is able to create single edges from single lines even without the use of other filters, but has the problem that its edges are somewhat noisier and may lead to zig-zags in places as can be seen in Figures 14e–h. Finally, the results for LoG (Figures 14i–l) show a slightly different portrayal of the edges, with some cleaner (e. g., *Secord's Plant*) and some noisier (e. g., *Headlight*) than DoG. Ultimately, it is thus upon the artist to select the most suitable edge detection process and to potentially apply additional filters to get the best results for a given visual goal, depending on the chosen input image.

We can also adjust the interpolation function to increase the distance around the borders such that area dots have less possibility of being drawn, as shown in Figure 16a–d. This leads to a white border around edges, which may be desired in some cases. Alternatively, we can also change the borders filter to use, for example, a DoG and remove the block that tries to align the strokes, producing that every black pixels is drawn as a dot as shown in Figure 16e–h. This effect emphasizes the linear edges particularly well through dark clusters of stipple dots, as can be seen particularly well in the full-size versions in Figure 39 in the additional material. If the process results in large, essentially black regions then a process to replace the stipples with solid polygons could be added [Azami et al. 2019].

Finally, we can combine our process also with additional masks as shown in Figure 17 to adjust the way the distance function is computed in Equation 6. For example, we can use the original edge image (for a chosen edge detection process) to generate offset lines similar to Kim et al.’s [2008] hedcut stippling, and then use this offset line image as the input to compute the distance function in Equation 6. This process generates an effect as shown in Figure 17a. We can also use dedicated masks to emphasize specific parts of the image as shown in Figure 17b. Here, we masked the region of the lamp in the image in white, leave the rest black, and then use this image as the input to compute the distance Δ . By then adjusting b such that edge stipples are always drawn we achieve the emphasis effect in Figure 17b because, in the white region of the mask (for the area distribution), we define Δ to be 0 and thus ensure that the region’s area stipples are always shown. Finally, with the same approach we can also use image-independent masks such as wavy lines to generate the background effects as we show in Figure 17c.

7 CONCLUSION

In summary, we described a method to smoothly interpolate between two point distributions. The fundamental contribution of our approach is that we do not interpolate the resulting images, but the underlying dot distributions by means of Probability Density Functions (PDFs). Our approach is general: it can deal with any input distribution by relying on the discrete version of the PDFs such that it can also deal with input distributions that are not expressed as analytic PDFs but as discrete sets of dots. Moreover, the discrete approach also allows us to take advantage of GPU-based implementations and raster image filters.

This new distribution interpolation allows us to combine traditional stippling techniques for different regions in an input image—those that excel in shaded areas with those that focus on structure preservation. We thus can now generate stippling images that express the best of these two worlds and smoothly interpolate both point distributions—without the artifacts that would arise from a purely pixel-based interpolation of final stipple image results. For this interpolation between the distributions we made use of a rasterized distance field, based on an edge image that expresses the features that one wants to preserve. Our overall process provides a lot of flexibility in that individual stages can be adjusted, to create additional visual effects and feature emphasis.

The main limitation of our method is its inherent use of discrete raster representations during the interpolation process. For example, this approach can produce “stair” effects, in particular, for linear

structures. Typical ways to address these problems is to increase the resolution of the image or to apply anti-aliasing, the former of which we can also use in our case. The discrete approach also leads to another limitation, which is that our results are probabilistic—meaning that we only reproduce the input distribution up to the resolution of the interpolation raster of the discrete probability function. With a sufficiently fine interpolation raster, however, we can address this limitation because the resolution of the interpolation raster is largely independent from that of the input images, provided that we can compute respective distances from edge features.

In the future we thus want to focus on ways to reduce rasterization artifacts and to better control the interpolation output. For instance, we plan to investigate the mentioned higher-resolution interpolation processing (e. g., using vectorization of the edge image), explore better edge detection for isolated linear features, and ultimately come up with a fully analytic way of interpolation. Independent of these efforts, we also want to explore ways in which our tool can best be given to illustrators, to provide them with detailed control yet without the need to deal with complex parameter settings.

ACKNOWLEDGMENTS

We thank the author of WVS for his tool and the authors of SPS for their comparison images. This work was partially funded by the Spanish Ministry of Economy and Competitiveness with FEDER funds through project TIN 2017-85259-R.

REFERENCES

- Germán Arroyo, Domingo Martín, and M. Victoria Luzón. 2010. Stochastic Generation of Dots for Computer Aided Stippling. *Comput. Aided Des.* 7, 4 (2010), 447–463. <https://doi.org/10.3722/cadaps.2010.447-463>
- Ignacio Ascencio-Lopez, Oscar Meruvia-Pastor, and Hugo Hidalgo-Silva. 2010. Adaptive Incremental Stippling using the Poisson-Disk Distribution. *J. Graph. GPU Game Tools* 15, 1 (2010), 29–47. <https://doi.org/10.1080/2151237X.2010.10390650>
- Rosa Azami, Lars Doyle, and David Mould. 2019. Stipple Removal in Extreme-tone Regions. In *Proc. Expressive*. Eurographics Assoc., Goslar, Germany, 123–132. <https://doi.org/10.2312/exp.20191083>
- Michael Balzer, Thomas Schlömer, and Oliver Deussen. 2009. Capacity-Constrained Point Distributions: A Variant of Lloyd’s Method. *ACM Trans. Graph.* 28, 3, Article 86 (Aug. 2009), 8 pages. <https://doi.org/10.1145/1531326.1531392>
- Pascal Barla, Simon Breslav, Lee Markosian, and Joëlle Thollot. 2006a. *Interactive Hatching and Stippling by Example*. Technical Report RR-6461. Inria, France. <https://hal.inria.fr/inria-00084569>
- Pascal Barla, Simon Breslav, Joëlle Thollot, François X. Sillion, and Lee Markosian. 2006b. Stroke Pattern Analysis and Synthesis. *Comput. Graph. Forum* 25, 3 (Sept. 2006), 663–671. <https://doi.org/10.1111/j.1467-8659.2006.00986.x>
- Faruk H. Bursal. 1996. On interpolating between probability distributions. *Appl. Math. Comput.* 77, 2–3 (July 1996), 213–244. [https://doi.org/10.1016/S0096-3003\(95\)00216-2](https://doi.org/10.1016/S0096-3003(95)00216-2)
- John Canny. 1986. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 8, 6 (June 1986), 679–698. <https://doi.org/10.1109/TPAMI.1986.4767851>
- Jianghao Chang, Benoît Alain, and Victor Ostromoukhov. 2009. Structure-Aware Error-Diffusion. *ACM Trans. Graph.* 28, 5, Article 162 (Dec. 2009), 8 pages. <https://doi.org/10.1145/1618452.1618508>
- Liang-Chieh Chen, Jonathan T. Barron, George Papandreou, Kevin Murphy, and Alan L. Yuille. 2015. *Semantic image segmentation with task-specific edge detection using CNNs and a discriminatively trained domain transform*. arXiv preprint 1511.03328. <http://arxiv.org/abs/1511.03328>
- Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. 2008. Where Do People Draw Lines? *ACM Trans. Graph.* 27, 3, Article 88 (Aug. 2008), 11 pages. <https://doi.org/10.1145/1360612.1360687>
- Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. 2009. How Well Do Line Drawings Depict Shape? *ACM Trans. Graph.* 28, 3, Article 28 (Aug. 2009), 9 pages. <https://doi.org/10.1145/1531326.1531334>
- Oliver Deussen. 2009. Aesthetic Placement of Points Using Generalized Lloyd Relaxation.

- In *Proc. CAE Eurographics Assoc.*, Goslar, Germany, 123–128. <https://doi.org/10.2312/COMPAESTH/COMPAESTH09/123-128>
- Oliver Deussen, Stefan Hiller, Cornelius van Overveld, and Thomas Strothotte. 2000. Floating Points: A Method for Computing Stipple Drawings. *Comput. Graph. Forum* 19, 3 (Sept. 2000), 41–50. <https://doi.org/10.1111/1467-8659.00396>
- Oliver Deussen, Stefan Hiller, Cornelius W. A. M. van Overveld, and Thomas Strothotte. 1999. Computer-Generated Stipple Drawings. In *Proc. VMV. infix*, Sankt Augustin, Germany, 329–338.
- Oliver Deussen and Tobias Isenberg. 2013. Halftoning and Stippling. In *Image and Video based Artistic Stylisation*, Paul Rosin and John Collomosse (Eds.). Computational Imaging and Vision, Vol. 42. Springer, London/Heidelberg, Chapter 3, 45–61. https://doi.org/10.1007/978-1-4471-4519-6_3
- Bruce Gooch and Amy A. Gooch. 2001. *Non-Photorealistic Rendering*. A K Peters, Ltd., Natick. <https://doi.org/10.1201/9781439864173>
- Aaron Hertzmann. 1999. Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines. In *ACM SIGGRAPH Course 17 on Non-Photorealistic Rendering*. ACM, New York, 7:1–7:14. <http://mrl.nyu.edu/publications/npr-course1999/>
- Elaine R. S. Hodges (Ed.). 2003. *The Guild Handbook of Scientific Illustration* (2nd ed.). John Wiley & Sons, Hoboken, NJ, USA. <https://www.gnsi.org/gnsi-handbook>
- Tobias Isenberg, Bert Freudenberg, Nick Halper, Stefan Schlechtweg, and Thomas Strothotte. 2003. A Developer’s Guide to Silhouette Algorithms for Polygonal Models. *IEEE Comput. Graph. Appl.* 23, 4 (July/Aug. 2003), 28–37. <https://doi.org/10.1109/MCG.2003.1210862>
- Dongyeon Kim, Minjung Son, Yunjin Lee, Henry Kang, and Seungyong Lee. 2008. Feature-Guided Image Stippling. *Comput. Graph. Forum* 27, 4 (June 2008), 1209–1216. <https://doi.org/10.1111/j.1467-8659.2008.01259.x>
- SungYe Kim, Ross Maciejewski, Tobias Isenberg, William M. Andrews, Wei Chen, Mario Costa Sousa, and David S. Ebert. 2009. Stippling By Example. In *Proc. NPAR*. ACM, New York, 41–50. <https://doi.org/10.1145/1572614.1572622>
- SungYe Kim, Insoo Woo, Ross Maciejewski, and David Ebert. 2010. Automated Hedcut Illustration Using Isophotes. In *Proc. Smart Graphics*. Springer, Berlin/Heidelberg, 172–183. https://doi.org/10.1007/978-3-642-13544-6_17
- Iasonas Kokkinos. 2015. *Surpassing humans in boundary detection using deep learning*. arXiv preprint 1511.07386. <http://arxiv.org/abs/1511.07386>
- Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. 2006. Recursive Wang Tiles for Real-Time Blue Noise. *ACM Trans. Graph.* 25, 3 (July 2006), 509–518. <https://doi.org/10.1145/1141911.1141916>
- Hua Li and David Mould. 2010. Contrast-aware Halftoning. *Comput. Graph. Forum* 29, 2 (2010), 273–280. <https://doi.org/10.1111/j.1467-8659.2009.01596.x>
- Hua Li and David Mould. 2011. Structure-Preserving Stippling by Priority-Based Error Diffusion. In *Proc. Graphics Interface*. CHCCS, Waterloo, ON, Canada, 127–134. <https://graphicsinterface.org/proceedings/gi2011/gi2011-17/>
- Hua Li and David Mould. 2017. Priority-Based Stippling and its Stylization Applications. *Int. J. Creat. Interf. Comput. Graph.* 8, 2 (July–Dec. 2017), 31–53. <https://doi.org/10.4018/IJCIG.2017070104>
- Stuart P. Lloyd. 1982. Least Squares Quantization in PCM. *IEEE Trans. Inf. Theory* 28, 2 (March 1982), 129–137. <https://doi.org/10.1109/TVT.1982.1056489>
- Ross Maciejewski, Tobias Isenberg, William M. Andrews, David S. Ebert, Mario Costa Sousa, and Wei Chen. 2008. Measuring Stipple Aesthetics in Hand-Drawn and Computer-Generated Images. *IEEE Comput. Graph. Appl.* 28, 2 (March/April 2008), 62–74. <https://doi.org/10.1109/MCG.2008.35>
- David Marr. 1982. *Vision*. The MIT Press. <https://mitpress.mit.edu/books/vision>
- David Marr and Ellen C. Hildreth. 1980. Theory of Edge Detection. *P. Roy. Soc. B-Biol. Sci.* 207, 1167 (Feb. 1980), 187–217. <https://doi.org/10.1098/rspb.1980.0020>
- Domingo Martín, Germán Arroyo, Vicente del Sol, Celia Romo, and Tobias Isenberg. 2019. Analysis of Drawing Characteristics for Reproducing Traditional Hand-Made Stippling. *Comput. Graph.* 80 (May 2019), 1–16. <https://doi.org/10.1016/j.cag.2019.02.001>
- Domingo Martín, Germán Arroyo, M. Victoria Luzón, and Tobias Isenberg. 2010. Example-Based Stippling using a Scale-Dependent Grayscale Process. In *Proc. NPAR*. ACM, New York, 51–61. <https://doi.org/10.1145/1809939.1809946>
- Domingo Martín, Germán Arroyo, M. Victoria Luzón, and Tobias Isenberg. 2011. Scale-Dependent and Example-Based Stippling. *Comput. Graph.* 35, 1 (Feb. 2011), 160–174. <https://doi.org/10.1016/j.cag.2010.11.006>
- Domingo Martín, Germán Arroyo, Alejandro Rodríguez, and Tobias Isenberg. 2017. A Survey of Digital Stippling. *Comput. Graph.* 67 (Oct. 2017), 24–44. <https://doi.org/10.1016/j.cag.2017.05.001>
- Domingo Martín, Vicente del Sol, Celia Romo, and Tobias Isenberg. 2015. Drawing Characteristics for Reproducing Traditional Hand-Made Stippling. In *Proc. NPAR*. Eurographics Assoc., Goslar, Germany, 103–115. <https://doi.org/10.2312/exp.20151183>
- Michael McCool and Eugene Fiume. 1992. Hierarchical Poisson Disk Sampling Distributions. In *Proc. Graphics Interface*. Morgan Kaufmann Publishers Inc., San Francisco, 94–105. <https://doi.org/10.20380/GI1992.12>
- David Mould. 2007. Stipple Placement using Distance in a Weighted Graph. In *Proc. CAE Eurographics Assoc.*, Goslar, Germany, 45–52. <https://doi.org/10.2312/COMPAESTH/COMPAESTH07/045-052>
- David Mould and Paul L. Rosin. 2017. Developing and Applying a Benchmark for Evaluating Image Stylization. *Comput. Graph.* 67 (2017), 58–76. <https://doi.org/10.1016/j.cag.2017.05.025>
- Victor Ostromoukhov. 2001. A Simple and Efficient Error-Diffusion Algorithm. In *Proc. SIGGRAPH*. ACM, New York, 567–572. <https://doi.org/10.1145/383259.383326>
- Judith M. S. Prewitt. 1970. Object Enhancement and Extraction. In *Picture Processing and Psychopictorics*, Bernice Sacks Lipkin and Azriel Rosenfeld (Eds.). Academic Press, New York, 75–149.
- Paul Rosin and John Collomosse (Eds.). 2013. *Image and Video based Artistic Stylisation*. Computational Imaging and Vision, Vol. 42. Springer, London, Heidelberg. <https://doi.org/10.1007/978-1-4471-4519-6>
- Stefan Schlechtweg, Tobias Germer, and Thomas Strothotte. 2005. RenderBots—Multi-Agent Systems for Direct Image Generation. *Comput. Graph. Forum* 24, 2 (June 2005), 137–148. <https://doi.org/10.1111/j.1467-8659.2005.00838.x>
- Adrian Secord. 2002a. *Random Marks on Paper: Non-Photorealistic Rendering with Small Primitives*. Master’s thesis. Department of Computer Science, The University of British Columbia, Canada. <https://www.cs.ubc.ca/labs/imager/th/2002/Secord2002/>
- Adrian Secord. 2002b. Weighted Voronoi Stippling. In *Proc. NPAR*. ACM, New York, 37–43. <https://doi.org/10.1145/508530.508537>
- Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. 2015. Deep-Contour: A Deep Convolutional Feature Learned by Positive-Sharing Loss for Contour Detection. In *Proc. CVPR*. IEEE Computer Society, Los Alamitos, 3982–3991. <https://doi.org/10.1109/CVPR.2015.7299024>
- Irwin Sobel and Gary Feldman. 2014. A 3×3 isotropic gradient operator for image processing. Correspondence. <https://www.researchgate.net/publication/285159837>
- Minjung Son, Yunjin Lee, Henry Kang, and Seungyong Lee. 2011. Structure Grid for Directional Stippling. *Graph. Models* 73, 3 (May 2011), 74–87. <https://doi.org/10.1016/j.gmod.2010.12.001>
- Marc Spicker, Franz Götz-Hahn, Thomas Lindemeier, Dietmar Saupe, and Oliver Deussen. 2019. Quantifying Visual Abstraction Quality for Computer-Generated Illustrations. *ACM Trans. Appl. Percept.* 16, 1, Article 5 (Feb. 2019), 20 pages. <https://doi.org/10.1145/3301414>
- Marc Spicker, Franz Hahn, Thomas Lindemeier, Dietmar Saupe, and Oliver Deussen. 2017. Quantifying Visual Abstraction Quality for Stipple Drawings. In *Proc. NPAR*. ACM, New York, Article 8, 10 pages. <https://doi.org/10.1145/3092919.3092923>
- Thomas Strothotte and Stefan Schlechtweg. 2002. *Non-Photorealistic Computer Graphics. Modeling, Animation, and Rendering*. Morgan Kaufmann, San Francisco. <https://doi.org/10.1016/B978-1-55860-787-3.X5000-2>
- Saining Xie and Zhuowen Tu. 2017. Holistically-Nested Edge Detection. *Int. J. Comput. Vis.* 125, 1–3 (Dec. 2017), 3–18. <https://doi.org/10.1007/s11263-017-1004-z>

A ADDITIONAL MATERIAL

Below we provide additional material to support our discussion.

A.1 Original images

Figures 18–19 show the input to our algorithm comparison in Section 6. Figures 20–22 show the original results produced by the WVS and SPS techniques as they were produced by the original tools.

A.2 Large, complete results

In Figures 23–37 we show the complete versions of the images in Figures 11–14 we created for our comparison in Section 6. In addition, Figures 38–39 show the complete versions of the examples in Figure 16. Finally, Figure 40 shows larger versions of the images in Figure 17 so that details are better visible. Note that we were not able to include them in their intended target size of A5 as this would have taken too much space. Instead, we show them at $\frac{2}{5}$ of A5 size.

A.3 Additional results worthy of note

A few additional results can further illustrate our approach. Figure 41 compares the approach from Figure 39 with a different filter setup that places more emphasis on the edges. We also show the possible control of emphasis in Figure 42 which compares the previous result in Figure 17b or Figure 40b with a version that uses less emphasis

by showing more area stipples.

A.4 Demo program and video figure

We provide a demo implementation of our approach that we submit as additional material. We based it on *StippleShop* (github.com/dmperandres/StippleShop), which uses a block-based paradigm and already provides basic methods that are necessary to construct our solution. We added the necessary blocks to implement our new solution including the distance field, the interpolation, and the rendering. We illustrate the use of this demo for IPD in an additional video figure that serves as a tutorial. The tool can also be used to create images for other stippling approaches for a comparison.



Fig. 18. Original versions of the images we used as the input for comparing the algorithms. Images a and b are part of Mould and Rosin's [2017] benchmark set for non-photorealistic rendering. Image a (Flickr image 13897692457) is by Richard Schneider, © (i) (f) (s) CC BY-NC 2.0. Image b (Flickr image 16453416352) is by Photos By Clark, © (i) (f) (s) CC BY-NC 2.0. Image c shows Lena (Soderberg)—a frequently used benchmark image from image processing, formerly part of the USC-SIPI Image Database, used under the fair-use clause. Image d is © Adrien Secord, permission will be requested.

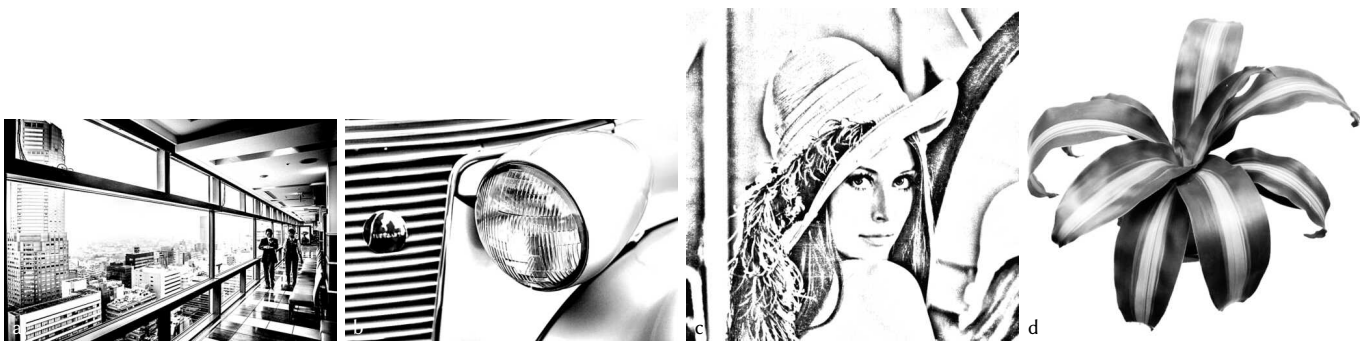


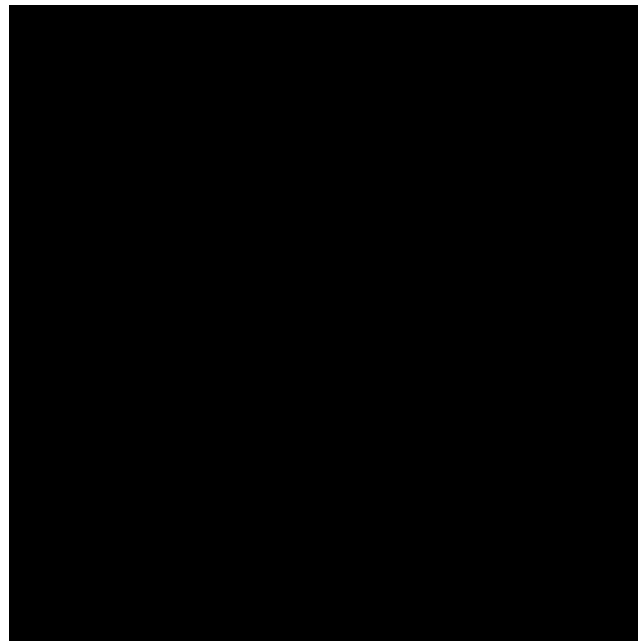
Fig. 19. Contrast-enhanced grayscale versions of Figure 18 (width of 1024 pixels) that we used as the input to our algorithm comparison in Section 6.



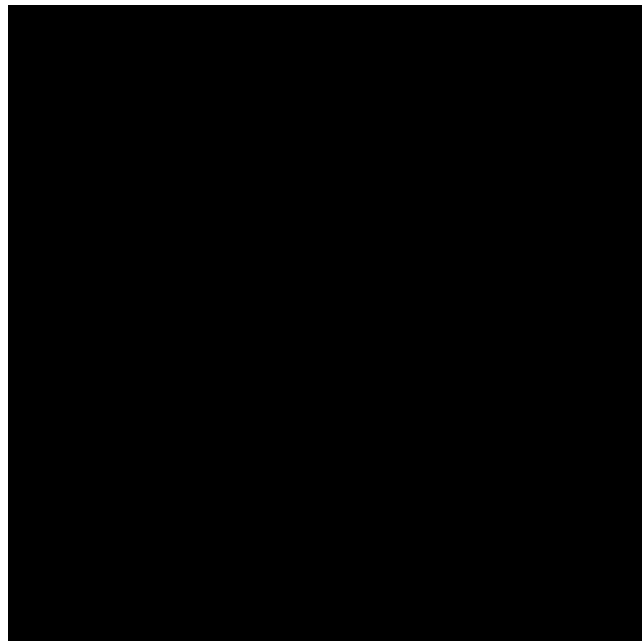
a



b



c



d

Fig. 20. Original results of Secord's [2002b] WVS method with dots of constant size. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).

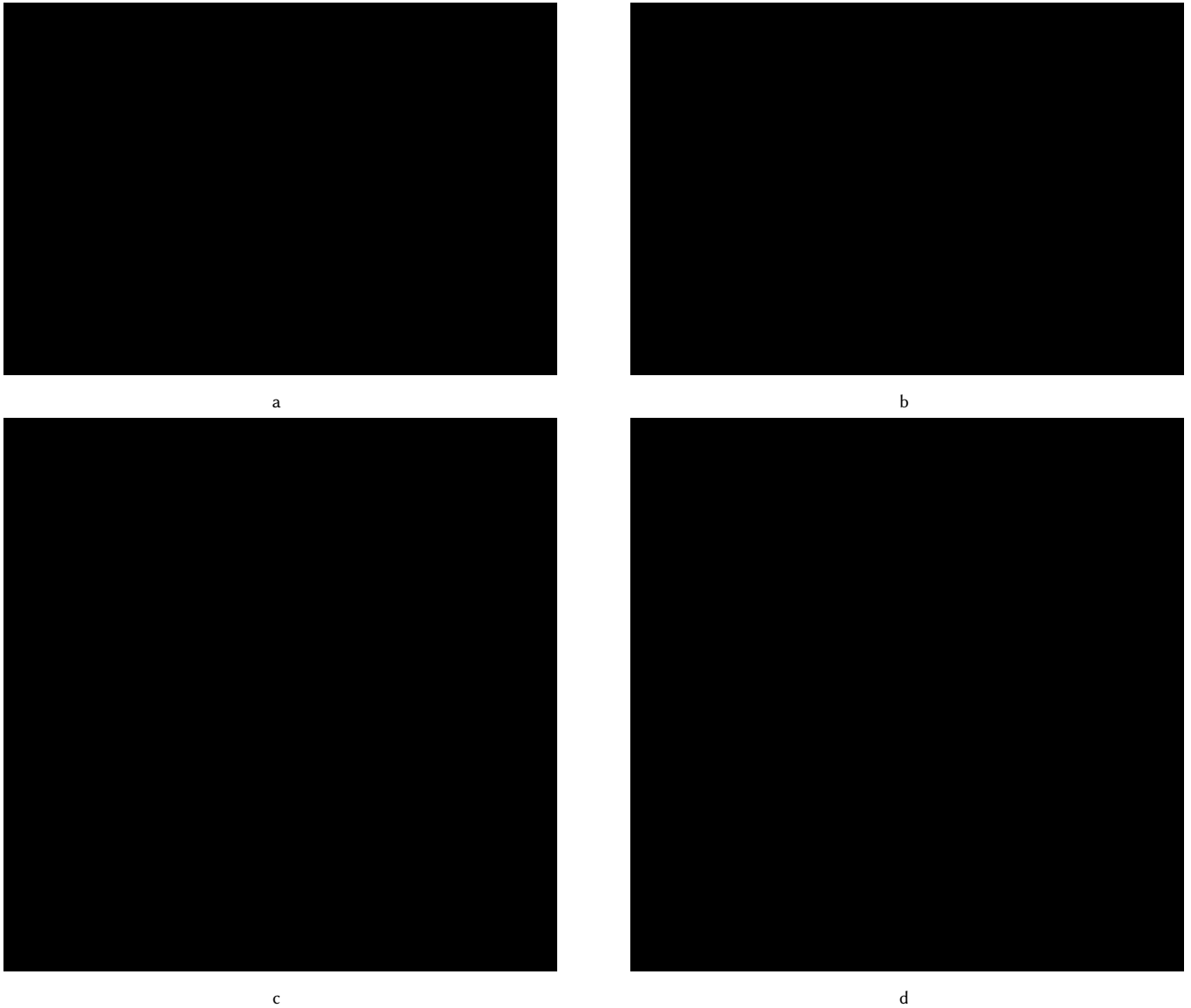


Fig. 21. Original results of Secord's [2002b] WVS method with its own modulation approach [Secord 2002a]. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space). The one unusually large stipple per image is an artifact (bug) of Secord's own original program which we used to create these images, we decided not to remove or adjust them—even if this would be possible.



a



b



c

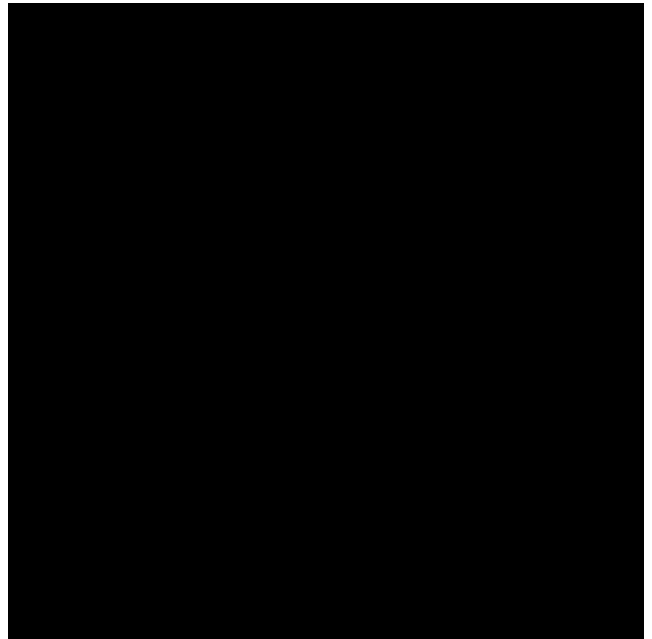


d

Fig. 22. Original results of Li and Mould's [2011; 2017] SPS method with dots of constant size. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



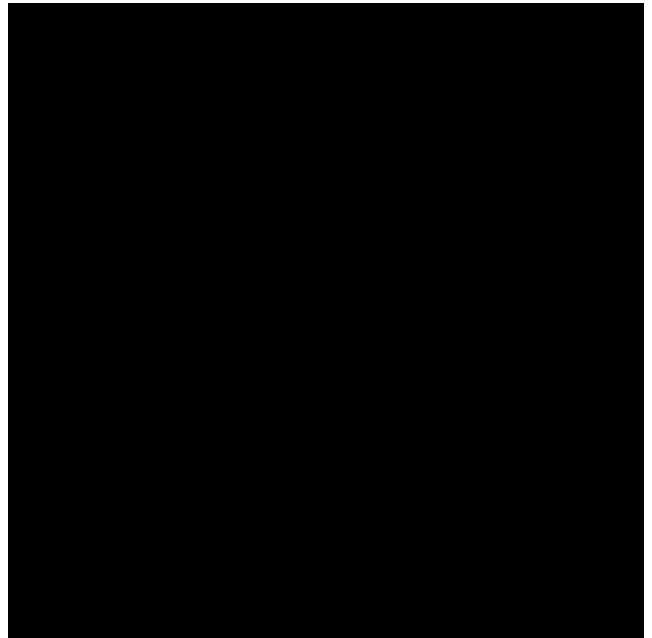
a



b



c



d

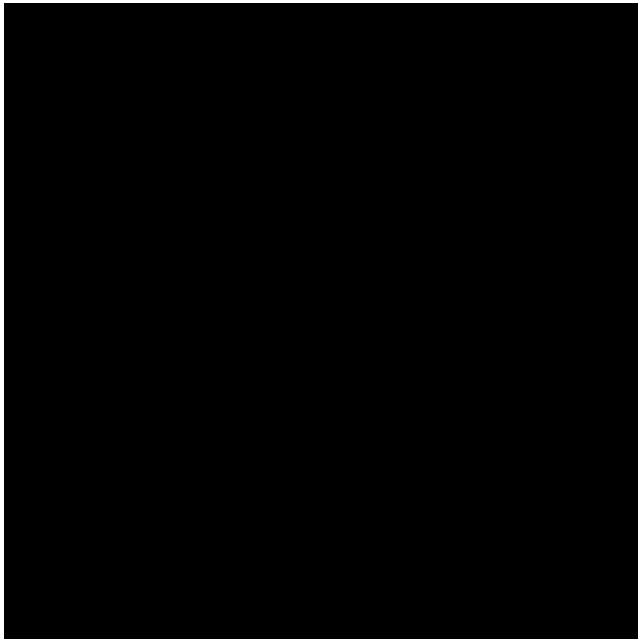
Fig. 23. Large and complete version of the detail sections in Figure 11a–d: Constant size dots, radius 0.5, B&W for WVS. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



a



b

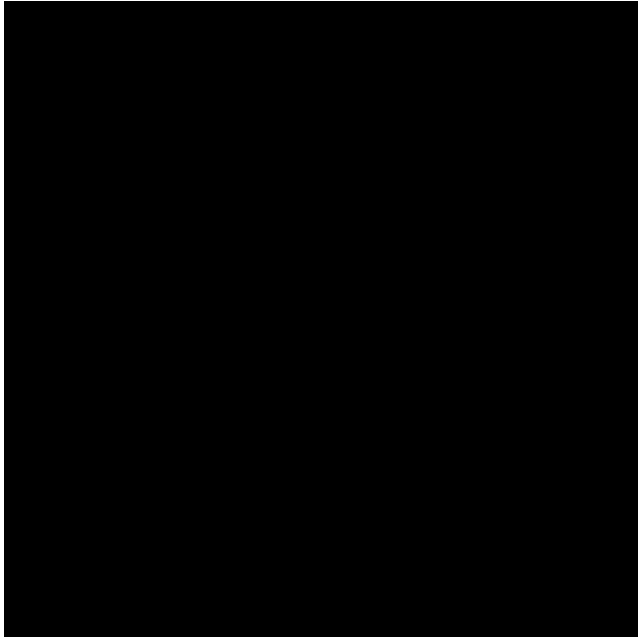


c

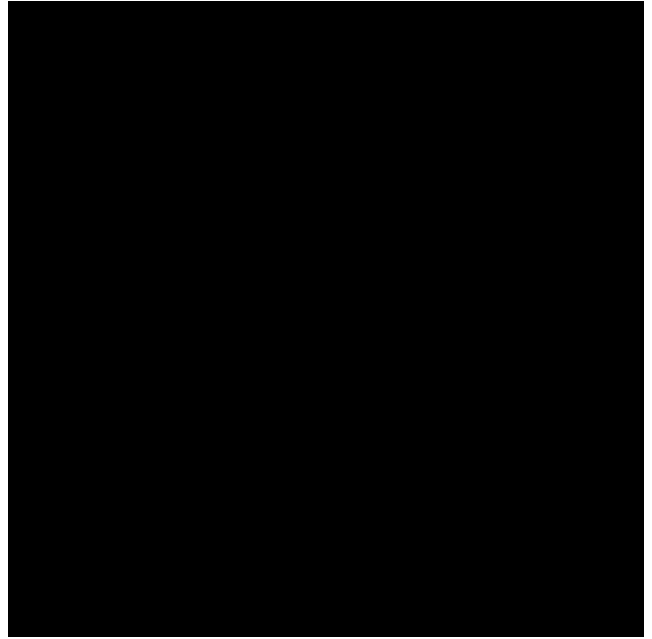


d

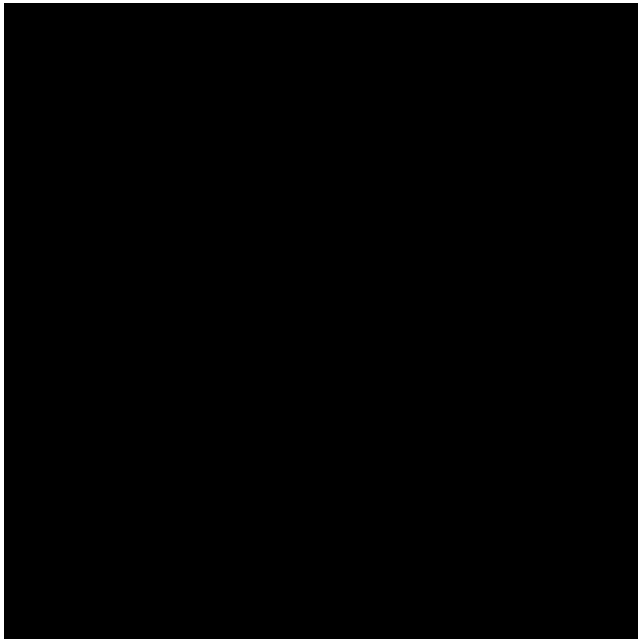
Fig. 24. Large and complete version of the detail sections in Figure 11e–h: Constant size dots, radius 0.5, B&W for SPS. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



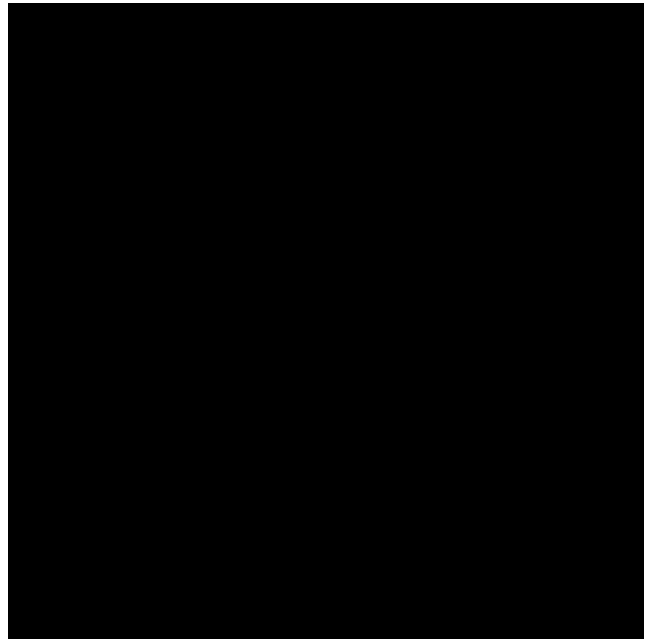
a



b



c

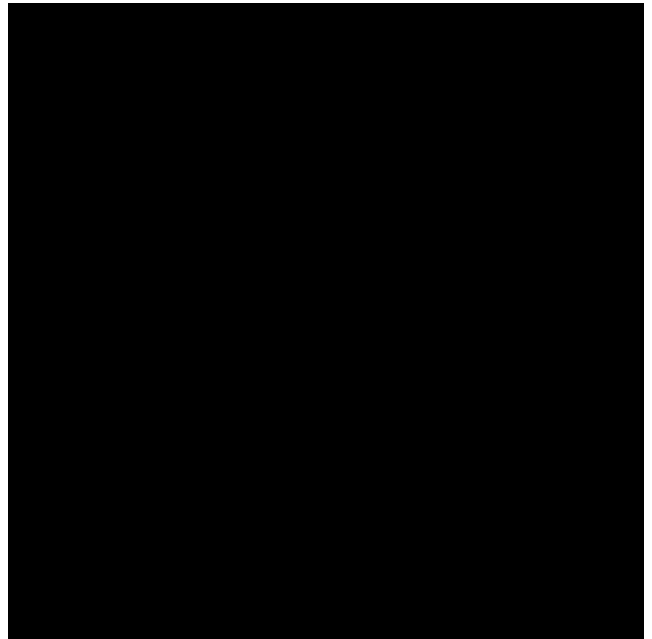


d

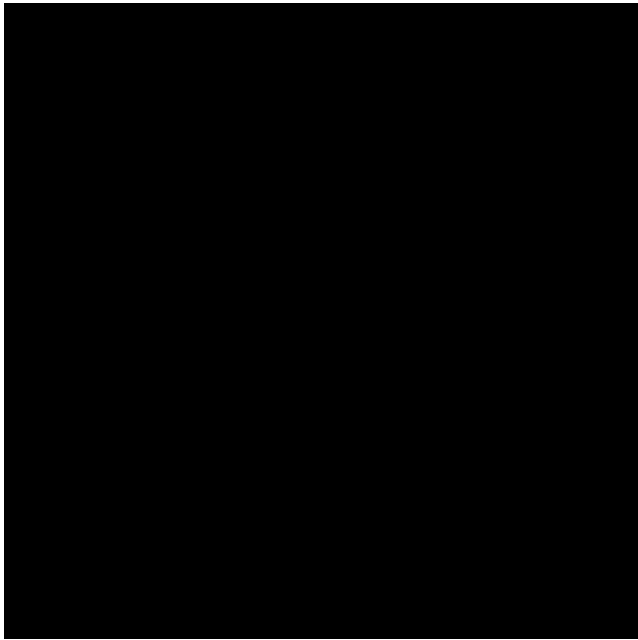
Fig. 25. Large and complete version of the detail sections in Figure 11i–l: Constant size dots, radius 0.5, B&W for EBG. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



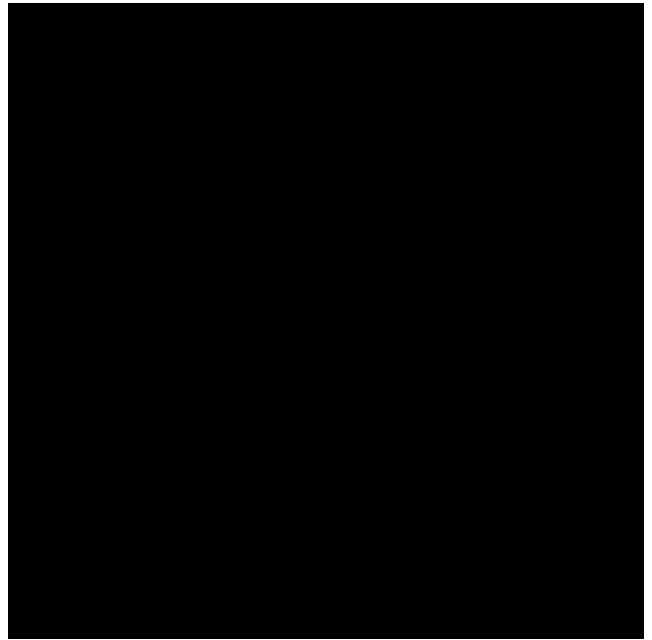
a



b



c

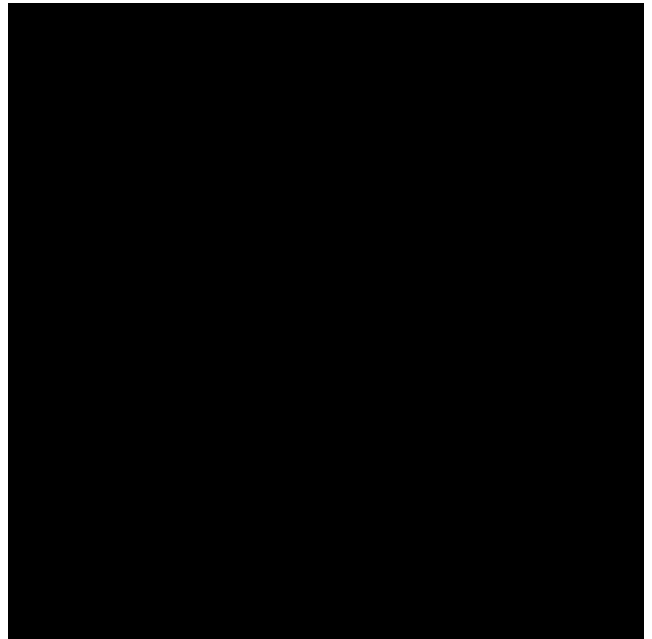


d

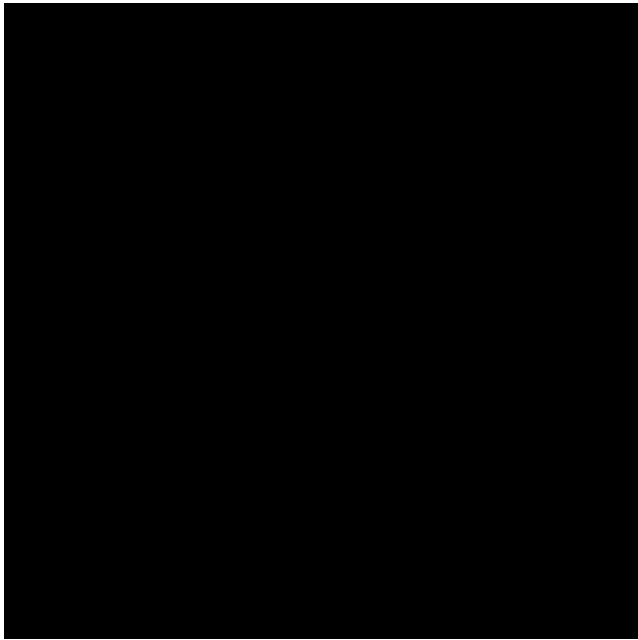
Fig. 26. Large and complete version of the detail sections in Figure 11m–p: Constant size dots, radius 0.5, B&W for IPD. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



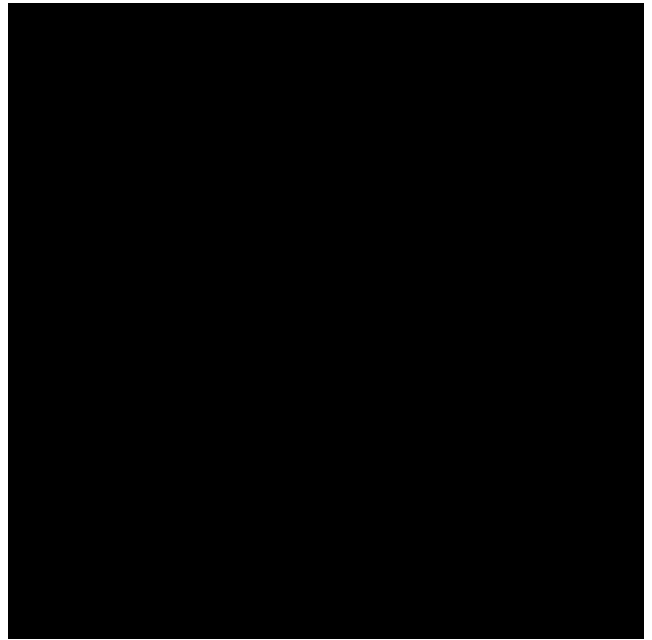
a



b



c

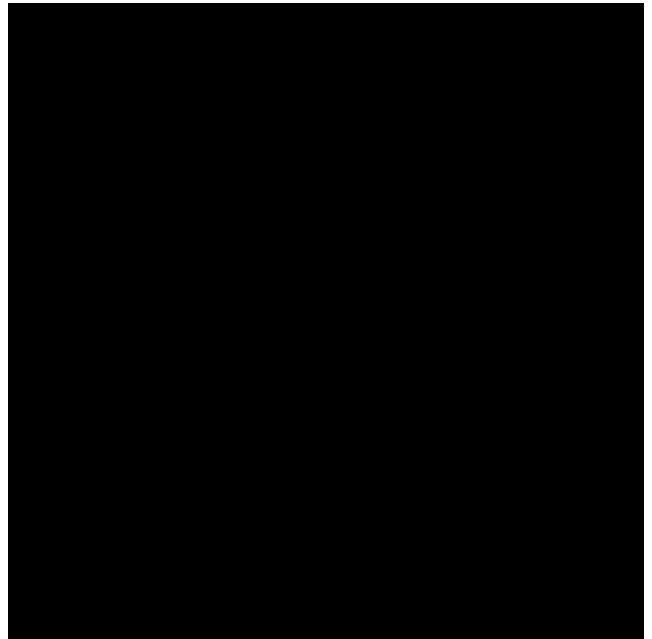


d

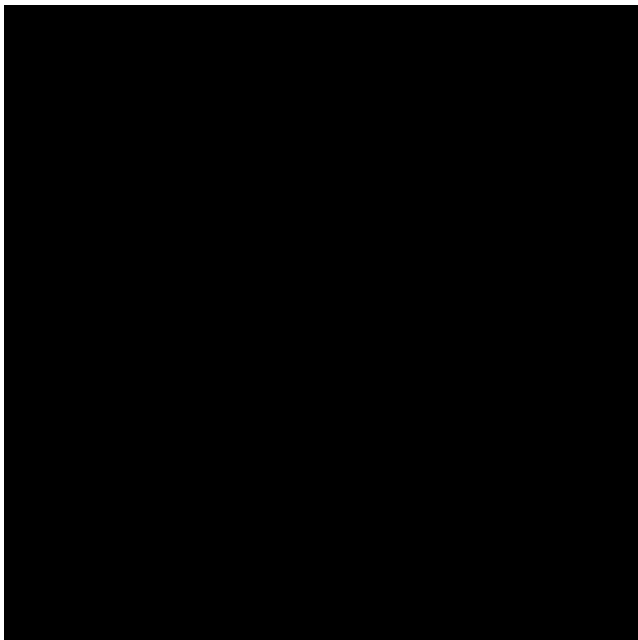
Fig. 27. Large and complete version of the detail sections in Figure 12a–d: Modulated size dots between 2 and 4, B&W for WVS. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



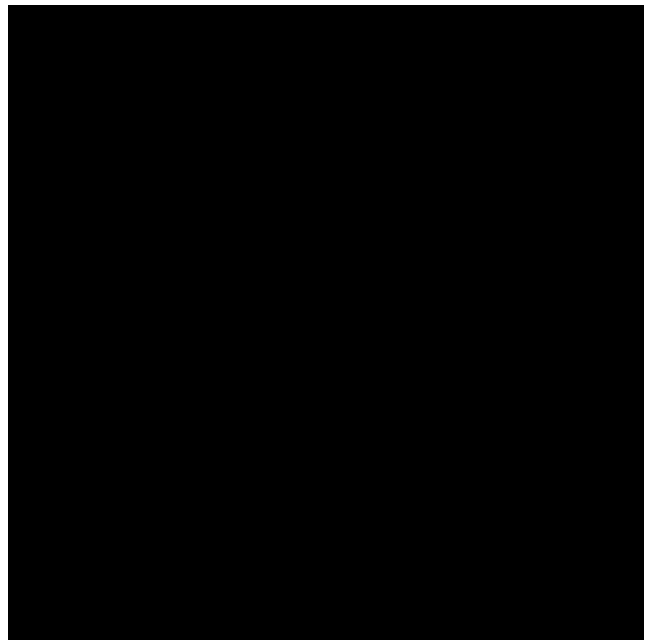
a



b

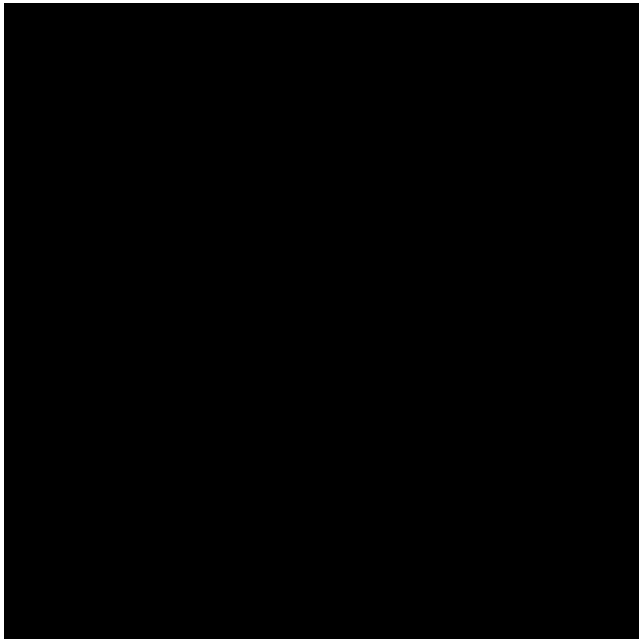


c

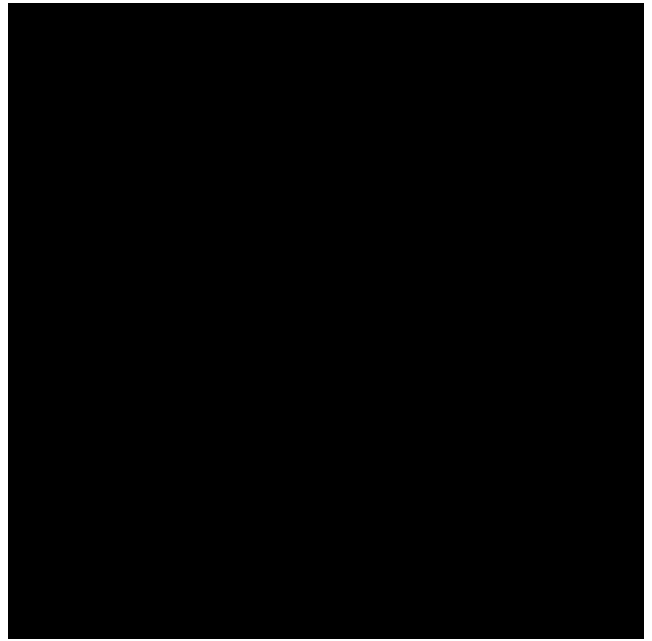


d

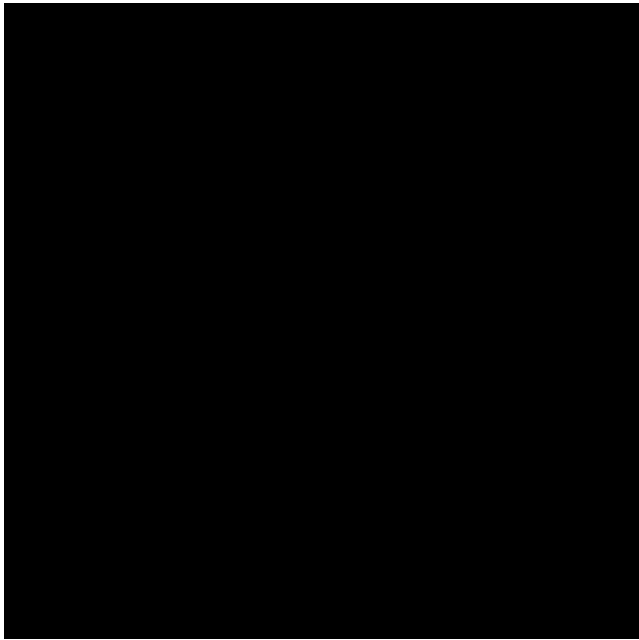
Fig. 28. Large and complete version of the detail sections in Figure 12e–h: Modulated size dots between 2 and 4, B&W for SPS. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



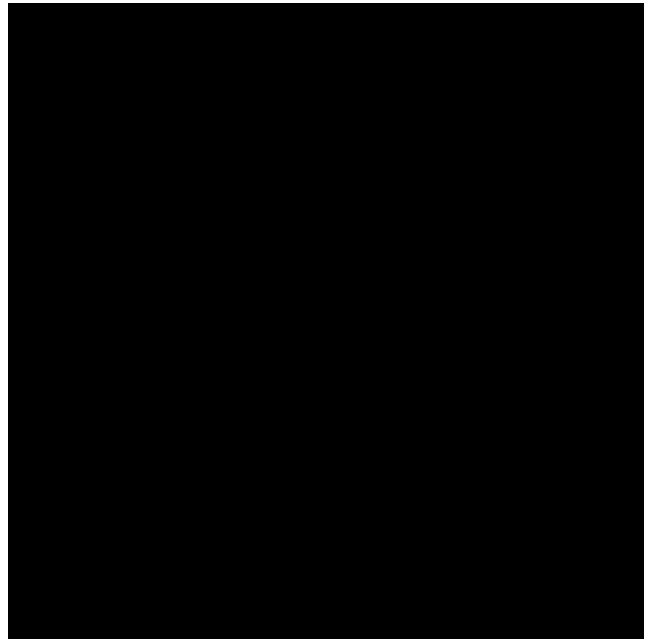
a



b



c

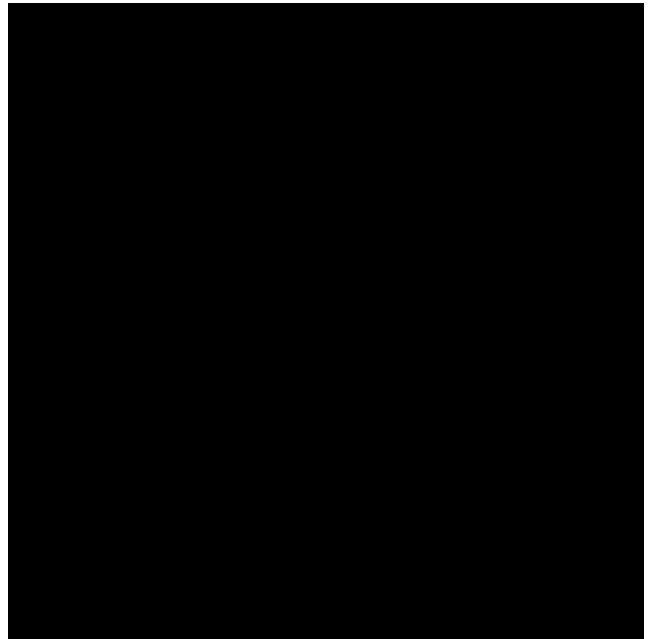


d

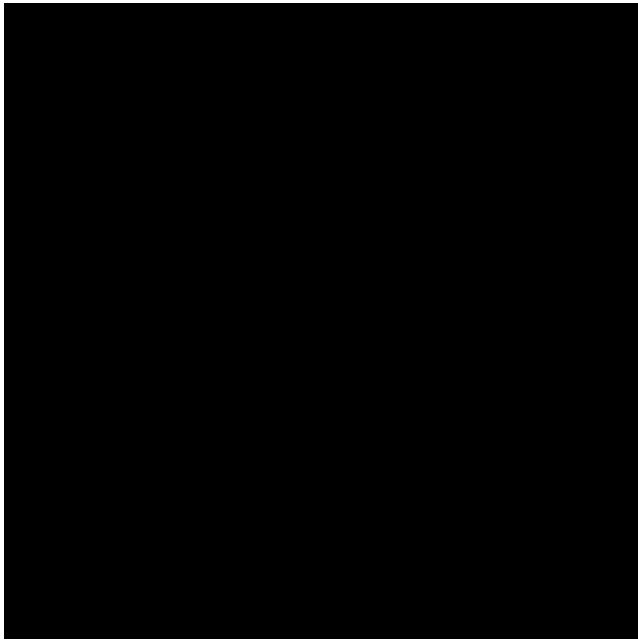
Fig. 29. Large and complete version of the detail sections in Figure 12i–l: Modulated size dots between 2 and 4, B&W for EBG. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



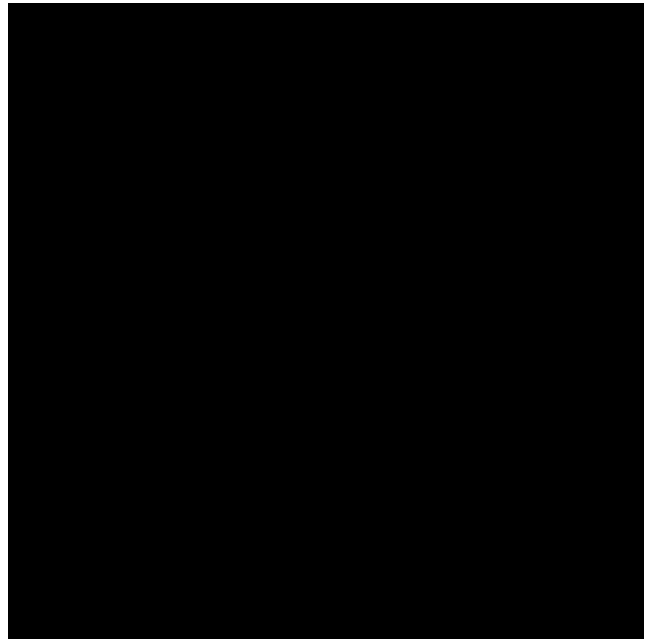
a



b



c



d

Fig. 30. Large and complete version of the detail sections in Figure 12m–p: Modulated size dots between 2 and 4, B&W for IPD. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).

a

b

c

d

Fig. 31. Large and complete version of the detail sections in Figure 13a–d: Scanned dots between 4 and 8, 1200 ppi, gray scale for WVS. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).

a

b

c

d

Fig. 32. Large and complete version of the detail sections in Figure 13e–h: Scanned dots between 4 and 8, 1200 ppi, gray scale for SPS. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).

a

b

c

d

Fig. 33. Large and complete version of the detail sections in Figure 13i–l: Scanned dots between 4 and 8, 1200 ppi, gray scale for EBG. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).

a

b

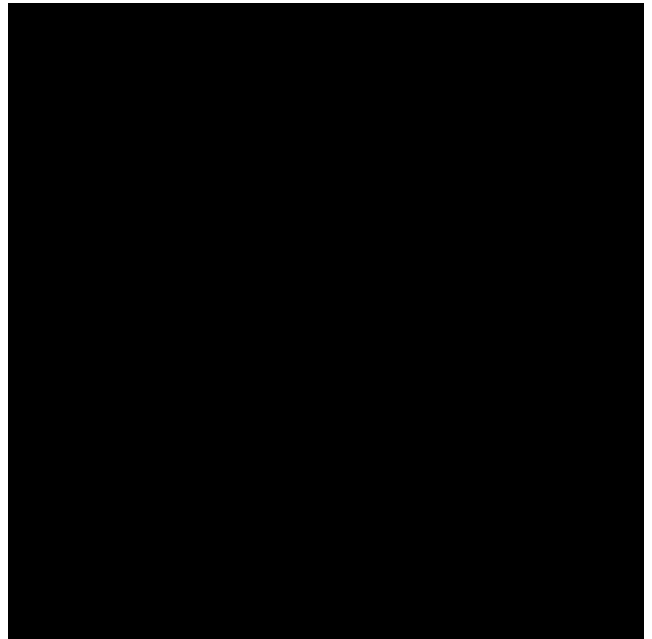
c

d

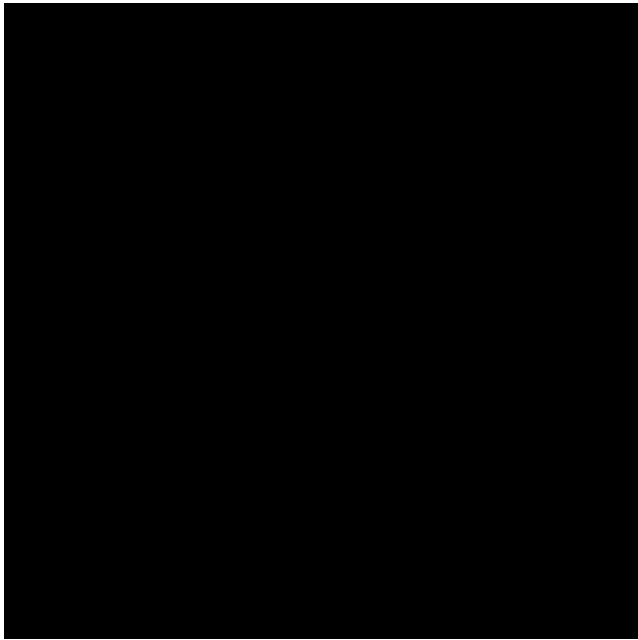
Fig. 34. Large and complete version of the detail sections in Figure 13m–p: Scanned dots between 4 and 8, 1200 ppi, gray scale for IPD. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



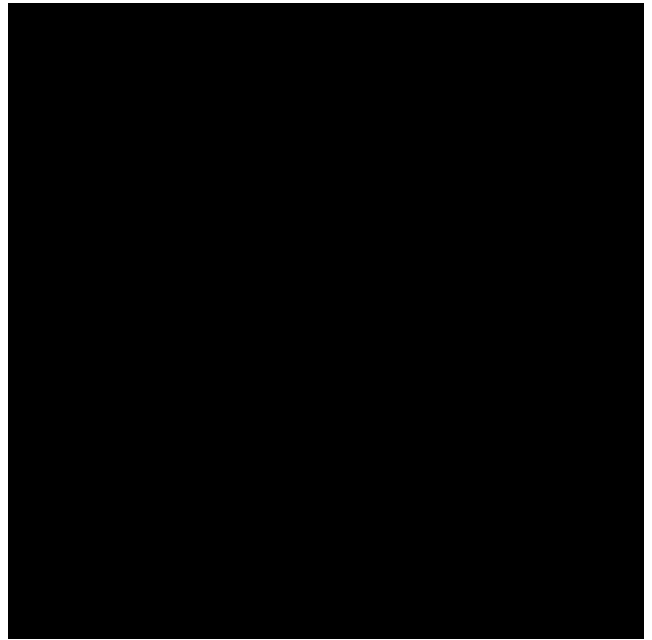
a



b



c

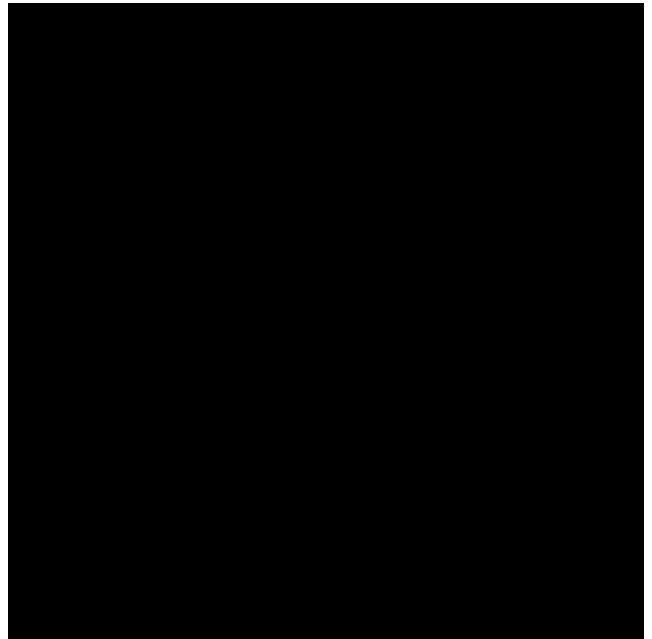


d

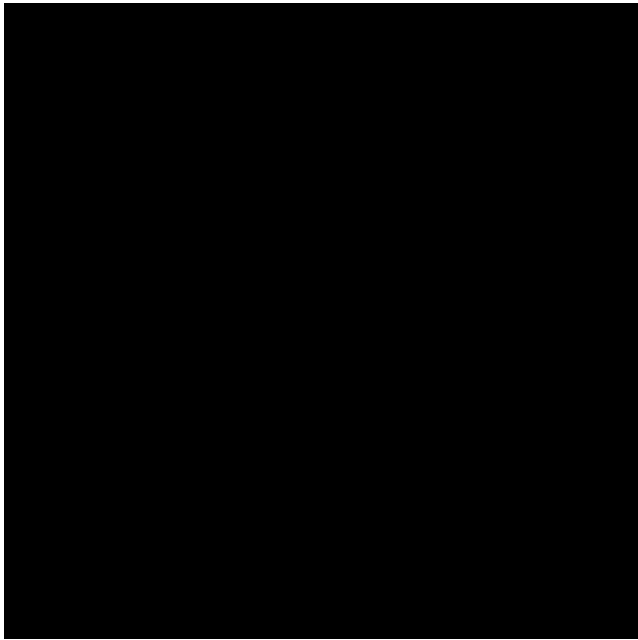
Fig. 35. Large and complete version of the detail sections in Figure 14a–d: Use of the Canny filter for for the IPD method. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



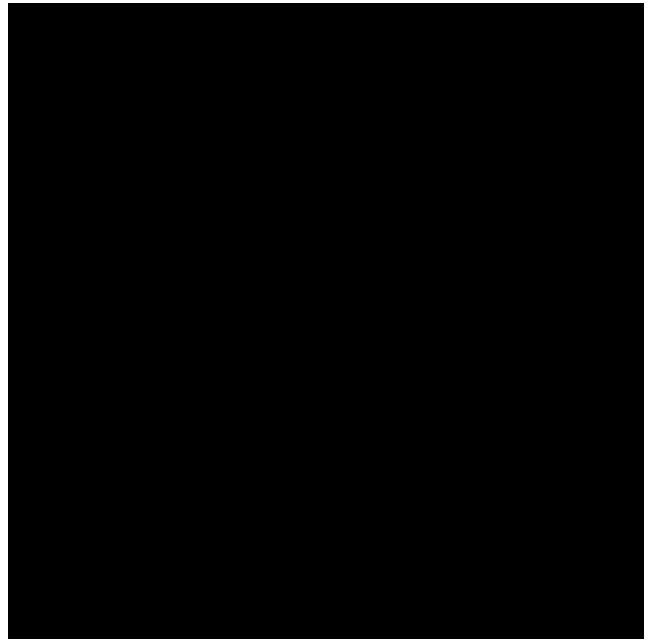
a



b



c



d

Fig. 36. Large and complete version of the detail sections in Figure 14e–h: Use of the DoG filter for for the IPD method. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



a



b



c

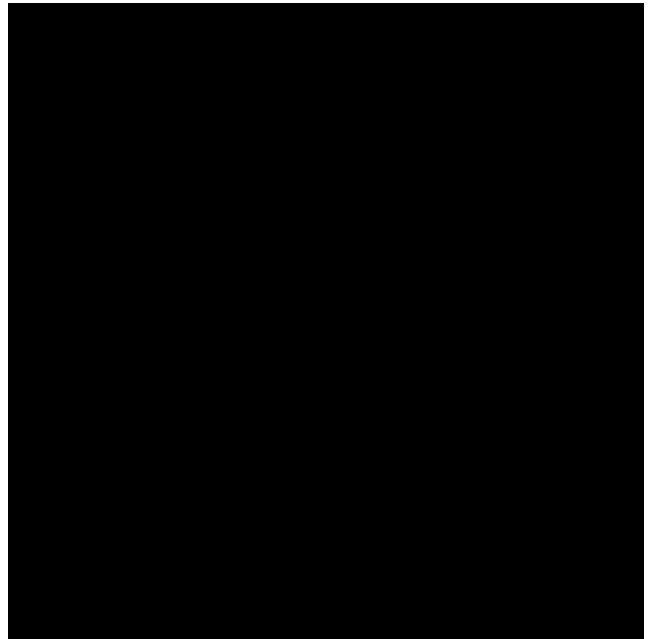


d

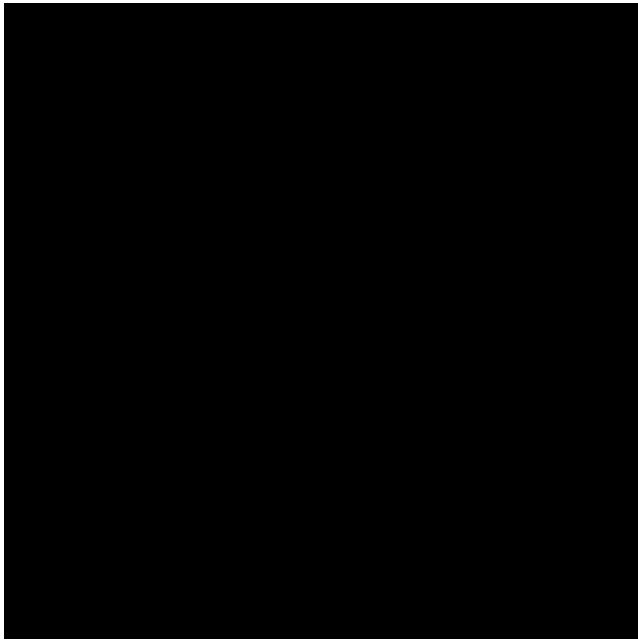
Fig. 37. Large and complete version of the detail sections in Figure 14i–l: Use of the LoG filter for for the IPD method. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



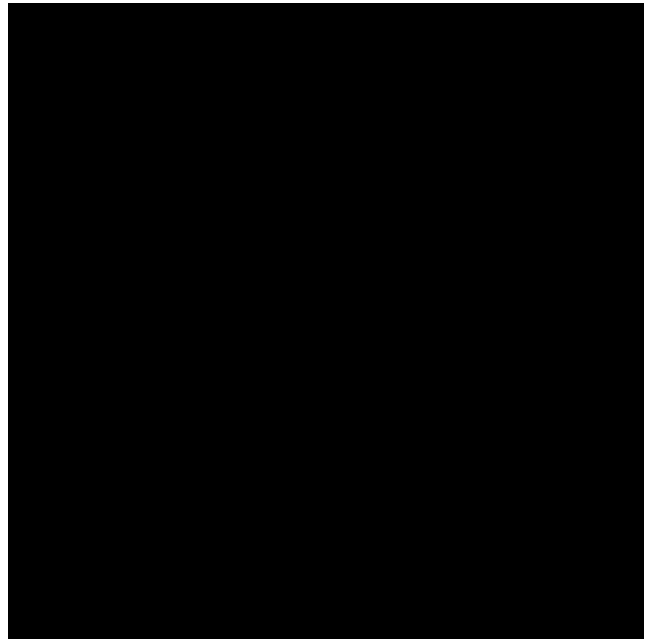
a



b

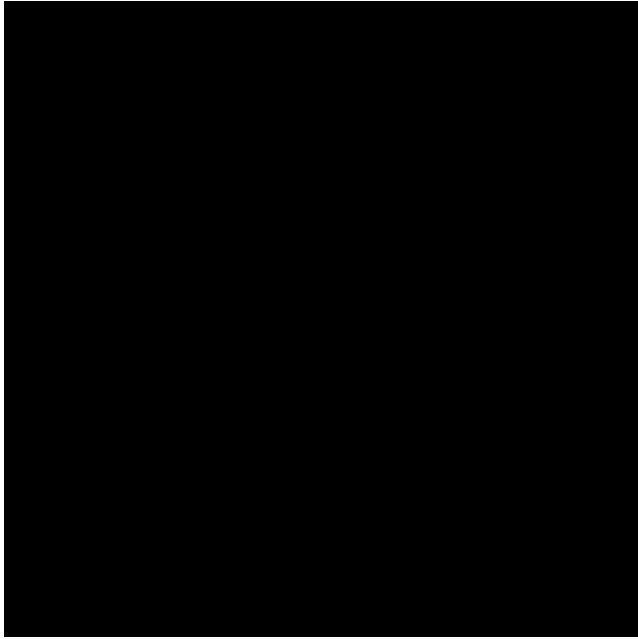


c

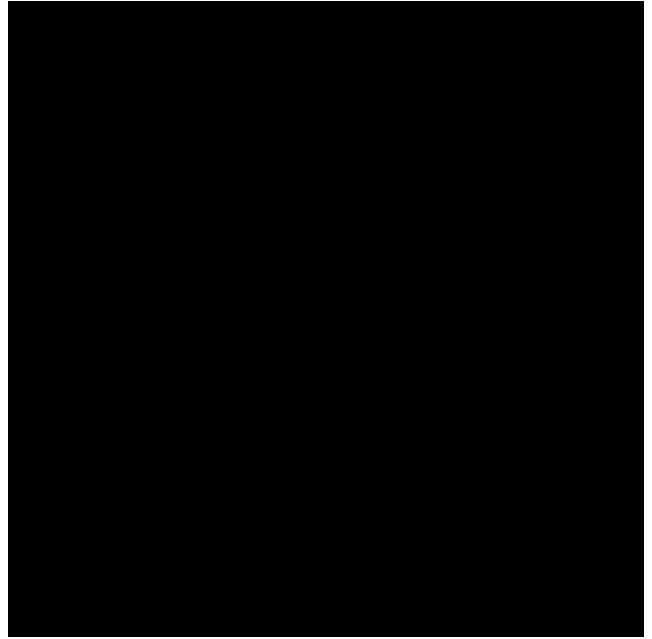


d

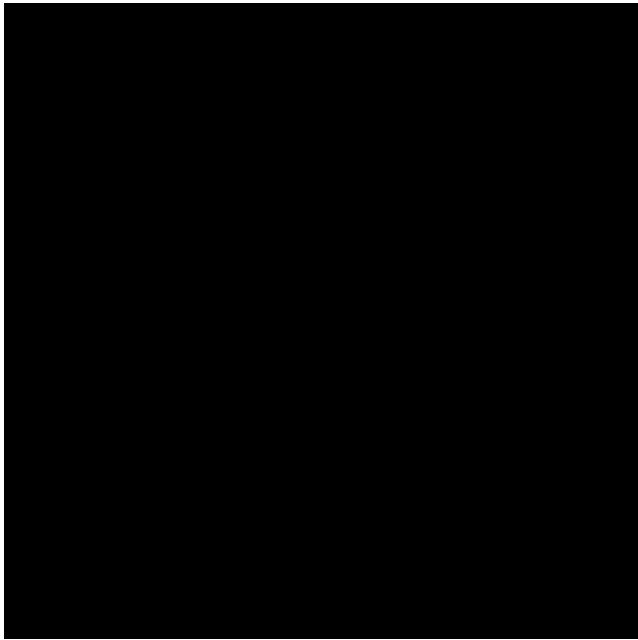
Fig. 38. Large and complete version of the detail sections in Figure 16a–d: Use of intentional margin around edges. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



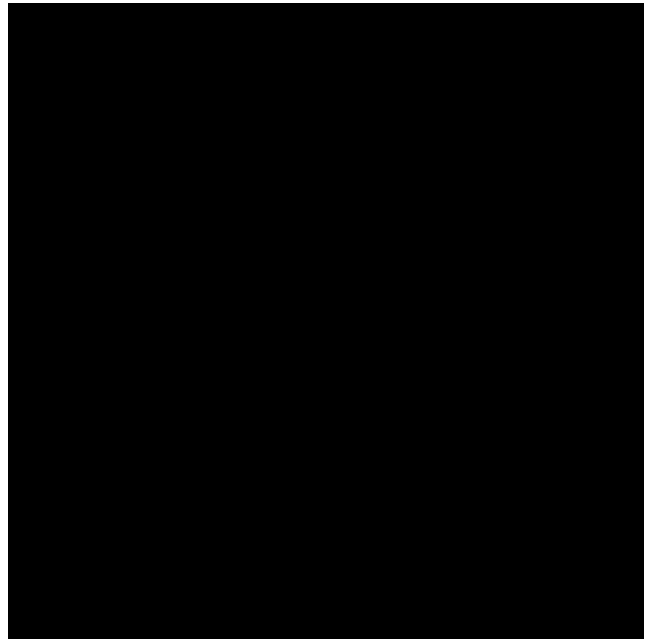
a



b



c



d

Fig. 39. Large and complete version of the detail sections in Figure 16e–h: DoG-based edge emphasis. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).

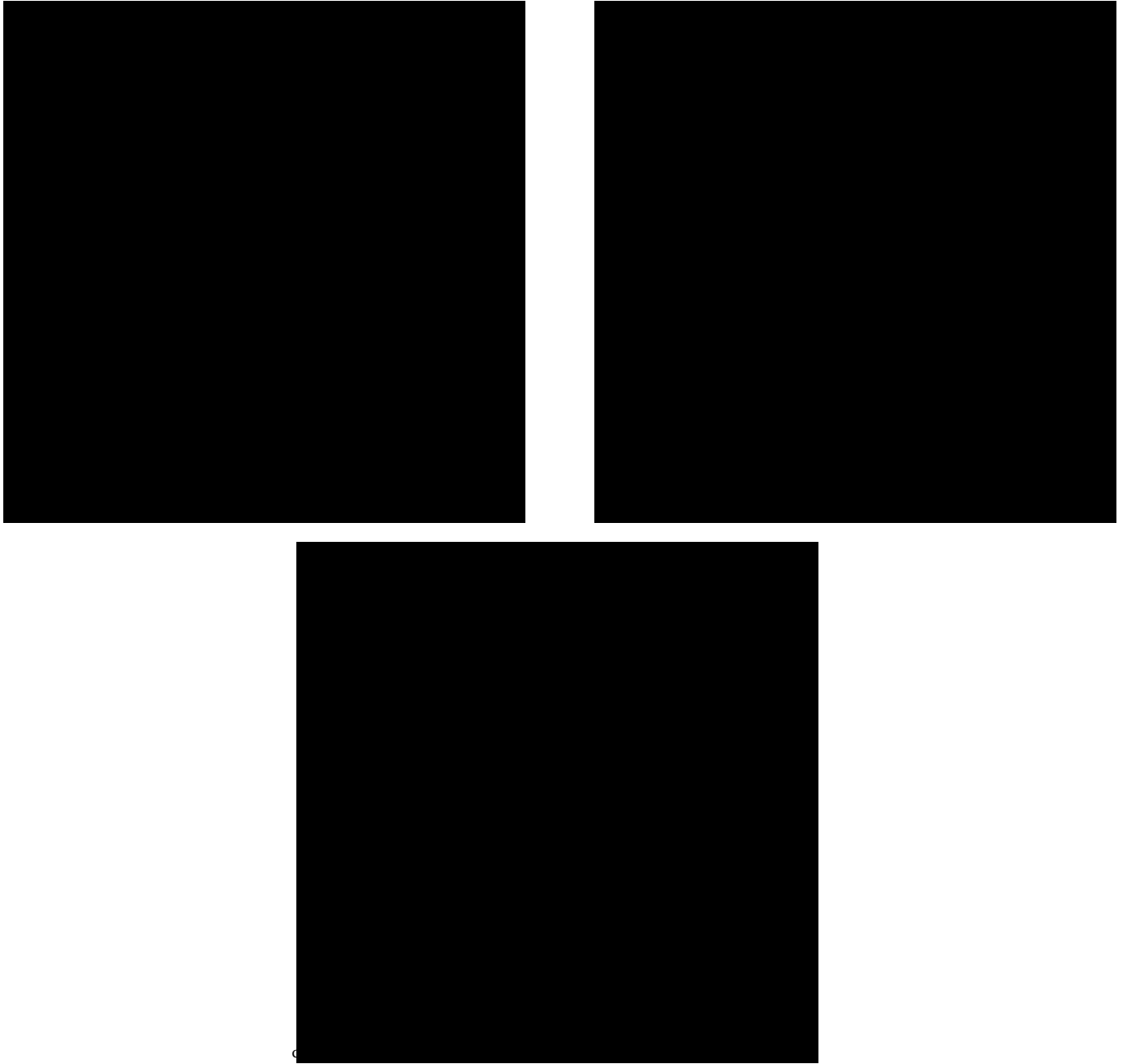


Fig. 40. Large versions of the examples in Figure 17: Examples for effects that make use of masks to adjust the distance function Δ , to control the use of stipples from the input distributions. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



Fig. 41. Comparison of Figure 39d (shown in a) with a different filter setup that places more emphasis on the edges (shown in b). We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).



Fig. 42. Comparison of the control of the degree of emphasis with the same mask, based on changing the b parameter of Equation 6. The image in a is the same as in Figures 17b and 40b. We show each image at $\frac{2}{5}$ of its intended target size of A5 (i. e., with base width of 84 mm instead of 210 mm, to conserve space).