POINTILLIST HALFTONING

Alejo Hausner
Department of Computer Science
University of New Hampshire
Durham, NH, USA
ah@cs.unh.edu

Abstract

We explore an extension of current halftoning algorithms that aims to reproduce pointillism, a style of post-impressionist painting. Halftoning algorithms solve the problem of reproducing an image with many colors on a device with very few colors (usually only two, black and white). Pointillist painters also created images using a very limited number of paint colors. Both halftoning and pointillism use dot patterns to suggest intermediate colors not available on the image created.

In order to reproduce pointillism, we extend error diffusion, which is a commonly used halftoning algorithm. Error diffusion relies on an ordered grid of pixels, which appear as dots of color on the final image. If these dots are initially not on a grid, but are placed randomly, a new algorithm is needed. Our solution is to build a graph of pixels with links between neighbors on the image, and replace row-by-row processing with graph traversal.

This extension of error diffusion is the main contribution of the paper. Beyond simulating an artist's technique, our extension improves color reproduction, by producing alias-free dot patterns and more highly saturated colors for offset color printing.

1 Introduction

Pointillism is a style of painting introduced by the post-impressionist Georges Seurat (1859-1891). It deliberately uses a limited set of colors, which are stippled one dot at a time onto the canvas. The result has a curious effect on the human visual system. It is fooled into seeing an entire gradation of color, when in fact only a few colors are used–usually right out of the paint tube [20].

In order to reproduce pointillism with software, we can view it as a kind of halftoning. Halftoning also uses a limited set of colors, most often just two colors: black ink on white paper. One important difference between pointillism and halftoning, however, involves the arrangement of dots. Dots in current error-diffusion halftoning algorithms are always arranged on a regular grid, whereas the dots in pointillist paintings are placed randomly. Perhaps current halftoning algorithms use a regular grid because that is the configuration of imaging devices for which they are intended: CRT displays, laser printer imaging heads and uniform meshes for halftone images in offset printing presses. Of course, pointillist painters had no such restrictions; they placed their dots of color by hand, and did not use a regular grid of dots.

Computer simulation of pointillism would be possible if an algorithm were available for color halftoning on random dots. In the following pages, we present just such an algorithm.

In addition to simulating an artistic technique, the method described below extends to other problems of image reproduction. Randomly-placed dots are often preferable to dots arranged on a uniform grid. This is because uniform sampling can lead to aliasing artifacts when used to reproduce images with high-frequency features such as fine repeating patterns or sharp edges. By replacing uniform sampling with random sampling, we replace these possible aliasing artifacts with noise, which the human visual system often finds less objectionable and less noticeable.

One promising application of the methods developed here is improved color printing. In the usual four-color offset printing process, an image is broken down into four images, each of a different color. These images are then printed consecutively on the same sheet of paper. Each of these components uses a regular grid of single-color dots, and the grid orientations must be carefully chosen to minimize interference between color components, which creates moiré patterns. Placing the color dots randomly eliminates this interference.

1.1 Challenges

We face the challenge that existing halftoning algorithms assume the device dots are arranged on a regular grid. Furthermore, most algorithms process these dots in straightforward scanline order. If the output dots are randomly positioned, then these algorithms cannot help us. Ordered-dither algorithms, in particular, strongly depend on a regular grid of dots. Thus, we need a way to organize a collection of dots, and establish a processing order for them.

1.2 Overview

The rest of this paper is organized as follows: Section 2 presents an overview of halftoning techniques, both by computer and by artists. Section 3 describes our algorithm, which is based on graph traversal. We study several choices for our graph-traversal algorithm. Section 4 shows the results of applying this algorithm to pointillism and random-dot color printing. Section 5 concludes the paper, and presents avenues for future research.

2 Background

2.1 Halftoning Algorithms

"Halftoning" describes a set of techniques for reproducing images with a large number of colors on a device with a limited number of colors. For simplicity, we will assume that the device has only two colors (black and white), although the following discussion can be extended to devices with more colors.

On a two-color device, a naïve approach might choose, at each pixel, the device color closest to the image color. This would, however, lead to color banding, which is the appearance of distinct color boundaries where the original image has a continuous color transition. Thus, sharp edges on the original image might be preserved, but color variations will be lost. Halftoning restores these variations, but at the price of reducing the detail in the image. This is achieved by using patterns of dots on the output device: the eye blends these dots into a continuous tone which is closer to the original image's color. Such patterns, of course, unavoidably lead to loss of spatial detail: the image looks grainy.

The simplest form of halftoning improves the naïve algorithm by adding noise. The choice of "closest" color is biased at each pixel by a random amount. This degrades the image's sharpness, but creates some color tones. The power spectrum of the noise affects the quality of the dither. Simple uniformly random numbers yield white noise, which produces low-quality images due to clumping. When dots are arranged with a blue-noise power spectrum (which omits low-frequency components), clumping will be avoided and the dot patterns will be more evenly spread-out dots, which is more pleasing and less distracting.

Deterministic methods such as Bayer's [1] dispersed-dot ordered dither try to produce spread-out dot patterns by defining special matrices of thresholds. One such matrix, a small (eg 4x4) block of numbers, will be replicated across the image, and a white pixel will be produced only if the original pixel value exceeds the matrix's threshold value at that position. Because the same matrix is replicated, however, distracting regular patterns appear in the final image.

Error diffusion is another simple method that produces satisfactory halftoned images. As in all the methods just presented, the image is processed in scanline order. To process one pixel, the closest output color is chosen, just as in simple thresholding. What makes this approach different is that the error introduced by simple thresholding is not discarded, but is distributed to the pixel's as-yet-unprocessed neighbors. Various choices of neighbors and relative distribution weights have been examined, but Floyd and Steinberg's [8] choice is popular. The resulting pixel patterns are, effectively, blue noise patterns.

Some recent work on error diffusion has been devoted to color inkjet printers, which allow precise placement of color dots. Color dots can be made to fuse visually and substitute for black, thus reducing visible speckle due to isolated black dots [9]. Printers which allow variable-sized ink dots add more degrees of freedom to error diffusion [18].

In commercial offset printing, such variable-size dots are necessary, because the physical properties of ink cause adjacent dots to merge, limiting the resolution available. Such printing must use clustered-dot halftoning, which places variable-sized round dots on a regular grid. Such a regular grid may in some cases lead to aliasing if the image is resampled, and moiré patterns when several color passes are superimposed. One way to avoid this is to place the dots in a random pattern. This pattern will have a power spectrum with frequencies in the middle band ("green noise" [10]). The method is called *stochastic clustered dot dithering* and is presented by Ostromoukhov and Hersch [17]. However, they restrict themselves to greyscale images. Our approach can also use such variable sized random dots, but extends the method to color images, which are more difficult because in addition to size there are several more color degrees of freedom.

For more details on halftoning in general, please see Ulichney's excellent overview [22] or Lau *et al*'s recent paper [10].

2.2 Halftoning by Artists

All the above halftoning methods assume that the output pixels lie on a regular rectangular grid. This restriction arises from the structure of current output devices. Hand-made ink drawings, however, are free from this restriction. One hand-made technique that resembles halftoning is stippling, wherein the artist places black dots of ink on the image, aiming to suggest a tone of gray. The dots are spread apart, but otherwise do not follow a regular pattern. This technique has been reproduced with software, using voronoi relaxation, by Deussen [3] and others [21]. The dot patterns that result approximate a Poisson-disc distribution, equivalent to randomly-placed yet disjoint circles. Poisson-disc distributions have low discrepancy, and thus yield low variance sampling patterns. Low variance visually corresponds to faithful reproduction of tone.

Another artistic technique that uses random dots is pointillism, which was introduced by the post-impressionist painter Georges Seurat. This technique is similar to stippling in that the artist aims to reproduce an overall color tone using isolated dots of paint. However, the artist does not limit him/herself to black ink, but to a set of pure colors. The aim is for the human eye to blend the dots into colors, avoiding the muddiness that results if the pigments are physically mixed together on the palette before being applied to the canvas.

Many previous simulations of painterly effects, such as Meier's [13] and Hertzmann's [7], focus mostly on the placement of paintstrokes. Their paintstroke colors are chosen from the original image, and are not restricted to a limited palette. They ignore the fact that many impressionists, like Seurat, preferred not to mix their pigments, and used many colors straight out of the paint tube [20] to get more vibrant colors. One exception is the approach by Luong *et al*, [12] who choose pixel colors with similar luminance to reduce visual noise. However, they assume a regular grid, and furthermore unlike pointillist artists, restrict themselves to a fixed pallete of colors.

As mentioned above, the main challenge to pointillist simulation is the need for a way to color the dots of paint. It is this challenge that we take up next.

3 Approach

When painting a pointillist image, the artist chooses both the location and the color of each dot of paint. Our simulation simplifies this process, by choosing point locations independently of point colors. We place the points first, and color them later.

Point placement is done using voronoi relaxation [21]. This technique uses Lloyd's method [11] to build a centroidal voronoi diagram (CVD). The method starts with a random arrangements of points, which are spread apart by repeatedly building the voronoi diagram, then moving each point to its voronoi region's center of mass. A good review of properties of CVD's is given in [4]. One useful property of such CVD's is that they approximate ideal poisson-disc distributions of points.

To color the dots, we then adapt the error diffusion algorithm. Error diffusion algorithms involve an ordering of the pixels and a distribution of pixel quantization error.

3.1 Quantization

The ordering of pixels is usually simple scan-line order, which is not available in our case because our dots are arranged randomly. We will present our ordering below, in Section 3.2. The second part, pixel quantization, is general enough to be used in our problem. Error diffusion works by quantizing, *i.e.* reducing the color of each pixel to one of a limited set of colors. This causes the pixel to differ from its original color, and to be in error. The error must then be propagated to the pixel's neighbors. On a regular grid, pixels are usually quantized in regular row-scanning order, and the neighbors are usually adjacent on the same or nearby row of pixels. In an irregular pixel pattern, the only complication we face is locating these neighbors.

Research aiming to improve error-diffusion algorithms addresses two problems: 1) which neighbors should receive the error, and 2) how much error each one should receive. In this work, we solve the first problem by choosing the ring of nearest neighbors to receive a quantized pixel's error. We solve the second problem in the simplest way possible: by apportioning the error equally among the neighbors. Other techniques for apportioning error are left to future research. Our algorithm then, is:

```
1. Obtain a connectivity graph G, based on pixel
   neighborhoods.
2. Obtain an ordered list L of nodes of G
   (a graph traversal).
3. For each node V in L,
4.      quantize V to the nearest available color.
5.      add the quantization error equally to all its unvisited
        neighbors.
```

It should be noted that this algorithm works both with random pixel patterns and raster-grid arrangements. For random pixel patterns, the connectivity graph G can be obtained from the delaunay triangulation of the pixel centers.

3.2 Traversal Order

The connectivity graph G consists of a set of nodes (the pixels), and a set of edges (two nodes are joined by an edge if the corresponding pixels are neighbors). Thus an ordering of the pixels will effectively be a graph-traversal order.

We considered two graph-traversal algorithms: depth-first (DFS) and breadth-first (BFS). Both algorithms are given a graph and starting a node in the graph, and produce an ordered list of nodes visited. If we use a data structure F to represent the "frontier" of nodes that are candidates to be visited next, we can write both algorithms as follows:

```
1. Initialize: add a "seed" pixel to F.
2. while F is not empty,
3.      V ← next element in F
        (several choices are possible)
4.      mark V as 'visited'
5.      for each unvisited node N incident on V,
6.           add it to F.
```

In DFS, F is a stack, while in BFS it is a queue. We can implement both algorithms by using a list for F, with elements added to its tail. In line 3 of the algorithm, removing elements from the tail of the list yields DFS, while removing them from the head of the list yields BFS. After some experimentation, we found that artifacts are minimized by using a hybrid of DFS and BFS. This hybrid is implemented by choosing the next node randomly between head and tail of the list.

4 Results

We applied the methods described above to several applications. First, we sought to simulate the efforts of pointillist artists, since that was the original inspiration for the technique. The method also produces good stippled images, which can be used to improve four-color offset printing.

4.1 Pointillism

Figure 1a shows a detail from H.E. Cross' "Portrait of Mme Cross", and Figure 1b shows our simulation, using eight colors which were extracted from the original using k-means color quantization [2]. Using these colors, the simulation applies pointillist halftoning to a filtered version of Cross' image, obtained by blurring. The filtered image blends his dots and shows the overall colors he probably intended. The method begins by placing dots arbitrarily, and initializing each dot color with the average color of the pixels it covers. Then, error diffusion is applied. We are trying to repeat Cross' method with an algorithm, but we cannot reproduce his image exactly.

Part of the difference in the two images comes from our choice of position, shape and orientation of the paint strokes, which differ from Cross'. Our present focus is on dot color, not dot placement. There is another interesting difference. As mentioned, we are using the same eight colors as Cross. For any small region in the Cross image, the dot colors will be different from the dot colors in the corresponding region in our reconstruction. However, *if one steps back, the overall color of the region appears the same*. Two factors are responsible here. First, as described in the introduction, the human visual system will blend color dots into an overall color if one steps back. Second, given a set of more than three colors (we are using eight here), one can choose many different subsets that can be mixed to yield the same final color. The mixing happens in the human visual system when the image is seen from a distance.

Figure 2a is a detail from and from Luce's "The Seine at Herblay", and 2b is our reconstruction. In addition to colors extracted from the original, we added the CMYK primaries to the palette, which guarantees that all the colors will lie inside the available gamut. If the gamut is not thus contained, artifacts can arise because the quantization error is not diffused locally; it remains present over large regions, and shows up as solid-color patches.

4.2 Color Printing

Our final result presents an attempt at improving four-color offset printing. Color images are usually reproduced using four inks, Cyan, Magenta, Yellow and Black. An RGB image can be decomposed into four single-color component images, each of which is printed in a separate pass of the printing press. Each of these passes uses a traditional halftone screen, *i.e.*, a clustered-dot ordered dither, inclined at an appropriate angle. These angles are chosen to avoid moiré patterns when the colors are superimposed. Figure 3a shows a simulated enlargement of such a printed image. One limitation of this current technique is that the ink dots lie on precomputed grids, and it is impossible to prevent overlaps between dots of different colors. A close look at this figure shows that the color dots do frequently overlap. Such overlaps lead to reduced color saturation.

Pseudo-random halftone screens [16], which are also a form of random-dot halftoning, provide another way to avoid moiré patterns, by replacing them with noise. However, they are generated independently for each of the four color images, and thus provide no way to avoid overlap between different-colored dots. Pointillist halftoning eliminates both aliasing and dot overlap: first, dots are placed in a poisson-disc pattern, and then colored using error diffusion. The resulting image (Figure 3b) maintains higher saturation, and more spatial detail. It is worth noting that both images use the *same* total number of color dots, but the regular halftone screen loses fine detail (compare the windows on the left side of the images). This suggests that, in addition to yielding more highly saturated colors, pointillist halftoning can be used to print images at higher resolutions with the same dot density. Of course, it should be pointed out that, for this approach to be successful, the alignment of consecutive passes of the printing press must be very precise, and this may not be easily achieved.

4.3 Performance

As mentioned in Section 3, our approach separates the tasks of point placement and point coloring. We used a simplified form of Hausner's [6] CVD graphics-card based technique to place the dots, obtaining satisfactory dot patterns after 20 iterations. For 20,000 points this required about 40 seconds on a 900MHz PIII, and was dominated by graphics-to-CPU I/O time. By contrast, the dot-coloring phase (our graph traversal algorithm) for the same dots took less than one second. Thus the bulk of the cost of the algorithm is not in the halftoning itself, but in the point placement. If dot patterns can be pre-computed once for all time, pointillist halftoning will be fast.

A simple way to achieve faster performance would be to generate a CVD with a moderate number of points, with toroidal (wrap-around) boundaries, and to tile images with copies of this CVD. The graph structure of this tile could be determined once for all images, and the traversal could be made deterministic and table-driven. Thus the cost of point placement would be incurred once, and the graph traversal would be much faster.

5 Conclusion

A new technique for reproducing continuous-color images on reduced-color devices has been presented, which automates the techniques of pointillist painters. It emulates their use of a limited palette of colors, whereas many current painterly simulations do not take into consideration this method used by painters of the period. The use of a limited set of vibrant colors is key, for the end result has more sparkle and scintillation than if a larger color set was used. Beyond reproducing an artistic effect on a computer, the algorithm serves to produce alias-free and sharper color image reproduction.

5.1 Future Work

In the current implementation, the main drawback is the time needed to place the points. This must be performed for every image. Precomputed point patterns, discussed above (4.3), promise great speedups. We plan to use them in our future implementations.

Further effort also needs to be devoted to more sophisticated choices of error-diffusion weights. The current implementation assigns the pixel error equally to all neighbor dots, which, in the case of a rectangular array of dots, is not optimal. For example, the Floyd-Steinberg neighbor weights produce better halftoning results. Probably good neighbor weights may be derived by placing a filter on the quantized pixel and obtaining each neighbor's filtered area. To be feasible, such an approach requires a precomputed point pattern, with weights computed once offline.

References

[1] Bayer, B.E., "An optimum method for two level rendition of continuous-tone pictures", *Proc. IEEE Int. Conf. Commun., Conference Record*, pp.(26-11)–(26-15), 1973.

[2] Bishop, C.M., *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[3] Deussen, O., Hiller, S., van Overveld, C.W.A.M., and Strothotte, T., "Floating Points: A Method for Computing Stipple Drawings", Computer Graphics Forum, Vol. 19(3), pp. 40–51, Eurographics 2000 Conference Proceedings.

[4] Du, Q., Faber, V., and Gunzburger, M., "Centroidal Voronoi Tesselations: Applications and Algorithms", SIAM Review, 41(4):637–676, 1999.

[5] Glassner, A., *Principles of Digital Image Synthesis*, New York, Morgan Kaufmann, 1995.

[6] Hausner, A., "Simulated Decorative Mosaics", SIGGRAPH 2001, pp. 573-580.

[7] Hertzmann, A. "Painterly Rendering with Curved Brush Strokes of Multiple Sizes", SIGGRAPH 1998, pp. 453–460.

[8] Floyd, R.W., and L. Steinberg, "Adaptive algorithm for spatial grey scale", *SID Int. Sym. of Tech. Papers*, pp. 36–37, 1976.

[9] Klassen, R.V., Eschbach, R., and Bharat, K., "Vector Error-Diffusion in a Distorted Color Space", Proc. IS&T 47th Annual Conference, 1994, reprinted in *Recent Progress in Digital Halftoning* (Ed. R. Eschbach), IS&T, 1994, pp 63–65.

[10] Lau, D.L., Arce, G.R., and Gallagher, N.C., "Digital Color Halftoning via Generalized Error-diffusion and Vector Green-noise Masks", IEEE Trans. on Image Processing, 9(5), pp. 923–935, May 2000.

[11] Lloyd, S. "Least Square Quantization in PCM", *IEEE Transactions on Information Theory* 28(1982): 129–137.

[12] Luong, T-Q, Seth, A., Klein, A., and Lawrence, J, "Isoluminant Color Picking for Non-Photorealistic Rendering" Graphics Interface 2005.

[13] Meier, Barbard, "Painterly Rendering for Animation", SIGGRAPH 1996, pp 477–484.

[14] Naiman, A. and Lam, D., "Error Diffusion: Wavefront Traversal and Constrat Considerations", Graphics Interface '96 Proceedings, 19(1), pp. 78–86.

[15] Ostromoukhov, V., "A Simple and Efficient Error-Diffusion Algorithm", SIGGRAPH 2001, pp. 567–572.

[16] Ostromoukhov, V., "Pseudo-random Halftone Screening for Color and Black & White Printing", *Proc. 9th Int. Congress on Non-Impact Printing Technologies*, pp. 579–582, 1993.

[17] Ostromoukhov, V., and Hersch, R.D., "Stochastic clustered-dot Dithering", Journal of Electronic Imaging 8(4), Oct. 1999, pp. 439–445.

[18] Ostromoukhov, V., Emmel, P., Rudaz., Amidror, I., and Hersch, R.D., "Multi-level Color Halftoning Algorithms", Int. Symposium on Advanced Imaging and Network Technologies, Conf. on Imaging Sciences and Display Technologies, Oct. 1996, Berlin, SPIE Vol. 2949, pp 332–340.

[19] Rosenblum, Robert, *Paintings at the Musée D'Orsay*, New York: Stewart, Tabory & Chang, 1989.

[20] Russel, John *Seurat*, London, Thames & Hudson, 1997.

[21] Secord, A., "Weighted Voronoi Stippling", Non-Photorealistic Animation and Rendering 2002 (NPAR '02), Annecy, France, June 3–5, 2002.

[22] Ulichney R., *Digital Halftoning*, Cambridge, MIT Press, 1987.

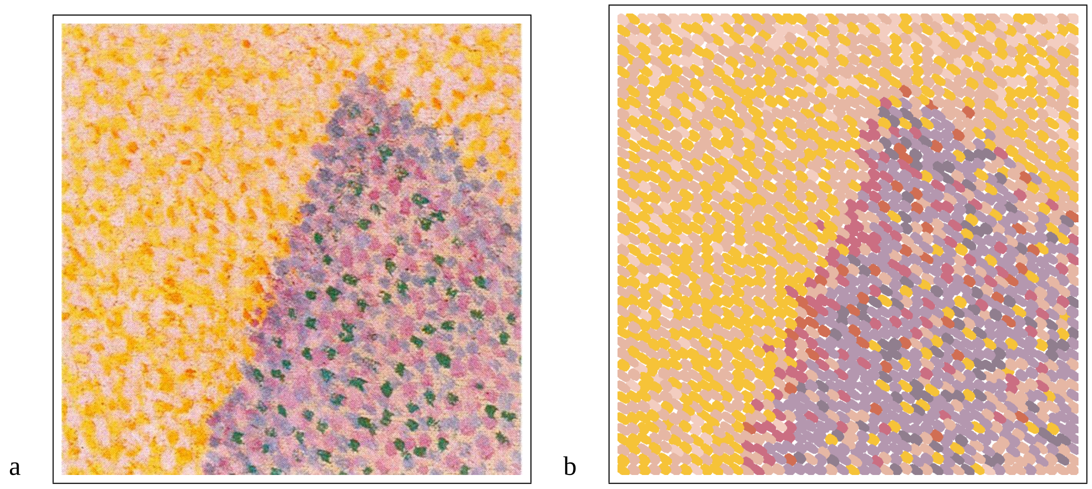

Figure 1. a) detail from H.E. Cross' "Portrait of Mme Cross", and b) our reconstruction.

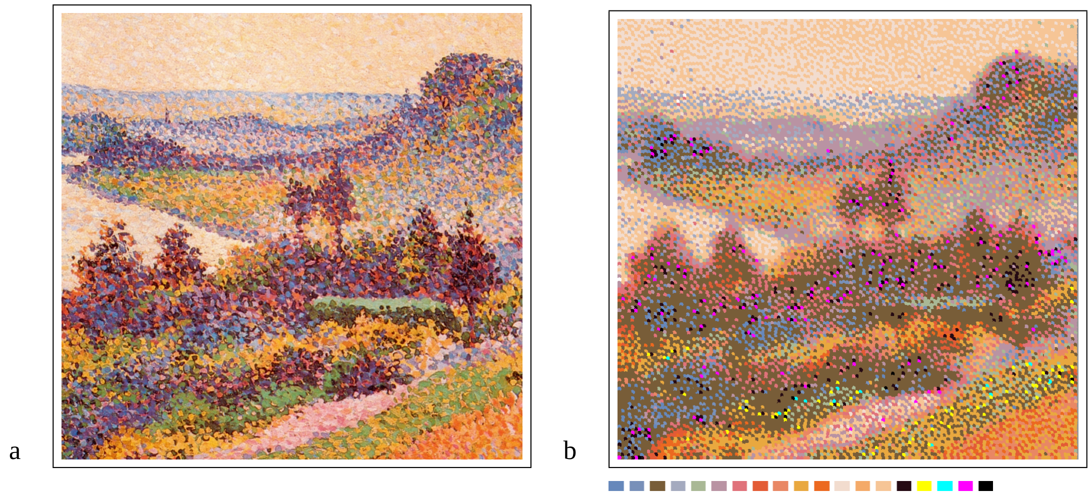

Figure 2. a) Detail from Maximilien Luce's "The Seine at Herblay", and b) our reconstruction.

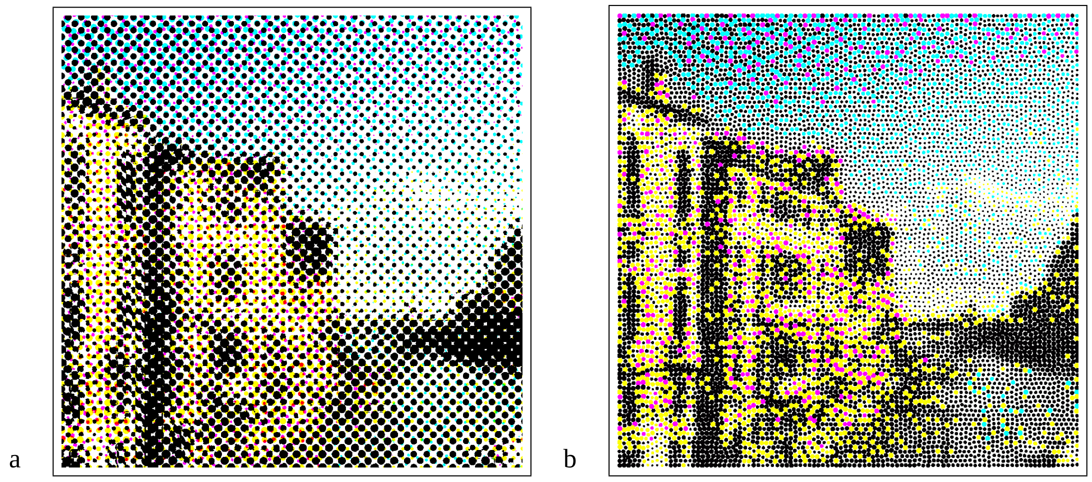

Figure 3. a) Standard four-ink reproduction of a color image. Notice the frequent overlaps between dots. b) Pointillist-halftone reproduction of the same image. The color dots are disjoint by construction.