MAESTRO: Making Art-Enabled Sketches Through Randomized Operations

Subhro Roy[1], Rahul Chatterjee[1], Partha Bhowmick[1], and Reinhard Klette[2]

[1] Indian Institute of Technology, Kharagpur, India
[2] The University of Auckland, New Zealand
subhroroy.iitkgp@gmail.com, rahuliitkgp08@gmail.com, bhowmick@gmail.com

Abstract. Contemporary digital art has an overwhelming trend of non-photorealism emulated by different algorithmic techniques. This paper proposes such a technique that uses a randomized algorithm to create artistic sketches from line drawings and edge maps. A *curve-constrained domain* (CCD) is defined by the *Minkowski sum* of the input drawing with the structuring element whose size varies with the pencil diameter. Each curve segment is randomly drawn in the CCD in such a way that it never intersects itself, whilst preserving the overall input shape. An artist's usual trait of making irregular strokes and sub-strokes with varying shades while sketching, is realistically captured in this randomized approach. Simulation results demonstrate its efficacy and elegance.

1 Introduction

Non-photorealistic rendering, originated as a promising digital art about two decades back [CAS97, VG91, VB99], has gained significant impetus in recent times [Deu10, GG01, LMHB00, MG02, Mou03, RMN03]. The works are mostly based on simulating the physical model through frictional coefficient, viscosity, smear factors, force factors, etc. [KHCC05, KNC08, KCC06, OSSJ09, PSNW07]. The factors of irregularity and obscurity arising out of an artist's creative mind—which prevail in an artistic creation and hence differentiate it from a machine-generated product—are, however, seldom noticed in the existing approaches. In fact, unless some (artistic) randomization is imparted, it is practically impossible to simulate an artistic creation, since the mystical, fanciful mind of an artist can hardly be scientifically modelled.

To incorporate a randomization factor while sketching a figure out of a set S of (irreducible) digital curve segments, corresponding to a real-world object, a novel simulation technique is proposed (Fig. 1). The fact that an artist often uses irregular strokes and sub-strokes with varying shades and intensities while sketching is rightly captured in our randomized curve sketching. In particular, some sketched segments get lightly shaded in our technique compared to other heavily-shaded ones, and a sub-segment also may be lighter or deeper in shade compared to the rest of the segment—in a pseudo-random or mystic manner—which characterizes the novelty of our algorithm.

Preliminaries. A linear-time algorithm to generate random digital curves in a closed canvas is proposed recently in [BPR10]. The work proposed here rests on the same theoretical foundation, but extends the algorithm further for

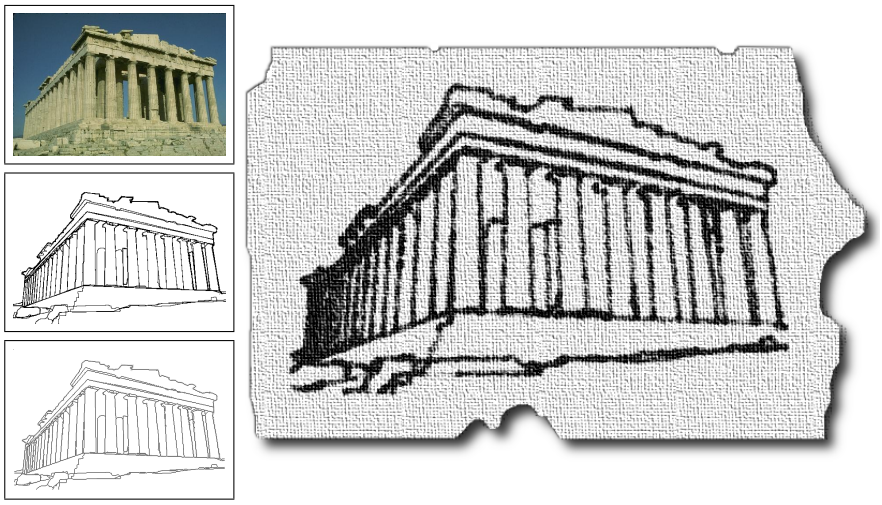

Fig. 1. Proposed algorithm. *Left-top:* Input image. *Left-mid:* After edge detection. *Left-bottom:* Skeletal image. *Right:* Final output by our algorithm, which resembles a crayon-drawn line sketch on a piece of handmade paper.

drawing random curves in CCD. Both the grid-point model and the cell model in $\mathbb{Z}^2$ are adopted in our work for their theoretical correspondence [KR04].

The input (thinned) digital image is first decomposed into a set $S = \{C_h\}_{h=1}^{N}$ of digital curves, each of which is simple and irreducible [KR04]. For each curve C_h, we prepare its *curve-constrained domain* (CCD), namely $\mathbb{D}_h$. The domain $\mathbb{D}_h$ signifies the region in which the segment (corresponding to C_h) sketched by the pencil will lie. Figure 2 illustrates a simple case where the segment starts from the point p and ends at the point q. Notice that the points p and q lie in two cells of $\mathbb{D}_h$. Vertices and centres of cells in $\mathbb{D}_h$ are assumed to be grid points in $\mathbb{Z}^2$, also simply called *points* for brevity in this paper. For each point $p \in C_h$, we take its Minkowski sum [KR04], namely $\mathbb{M}_p = \{q : q \in \mathbb{Z}^2 \wedge ||p-q|| \leq \lfloor t/2 \rfloor\}$, t being the width/thickness of the pencil-tip; then the domain $\mathbb{D}_h$ corresponding to C_h is defined by the union $\bigcup_{p \in C_h} \mathbb{M}_p$.

2 Curve Randomization in CCD

Contrary to polygon-generation algorithms [ZSSM96, AH96] that work with input vertices generated randomly but *a priori*, our algorithm generates new points *on the fly* (also called *online* in [KR04]) while creating a digital curve ρ. The curve ρ starts from $p = p_1$ and randomly chooses all the successive points, eventually ending at the destination point q (Fig. 2). The difficulty lies in making ρ *one pixel wide everywhere without intersecting itself*, thus becoming irreducible and simple. This calls for detecting every possible "narrow-mouthed" *trap* formed by the previously generated part of ρ, which, if entered into, cannot be exited without touching or intersecting ρ.

Principle of the Algorithm. A cell c (of a CCD, say, $\mathbb{D}_h$) is said to be *occupied* if and only if the generated part of curve ρ already passes through c; otherwise it is *free*. We use the following parameters for a cell c (see Fig. 2):

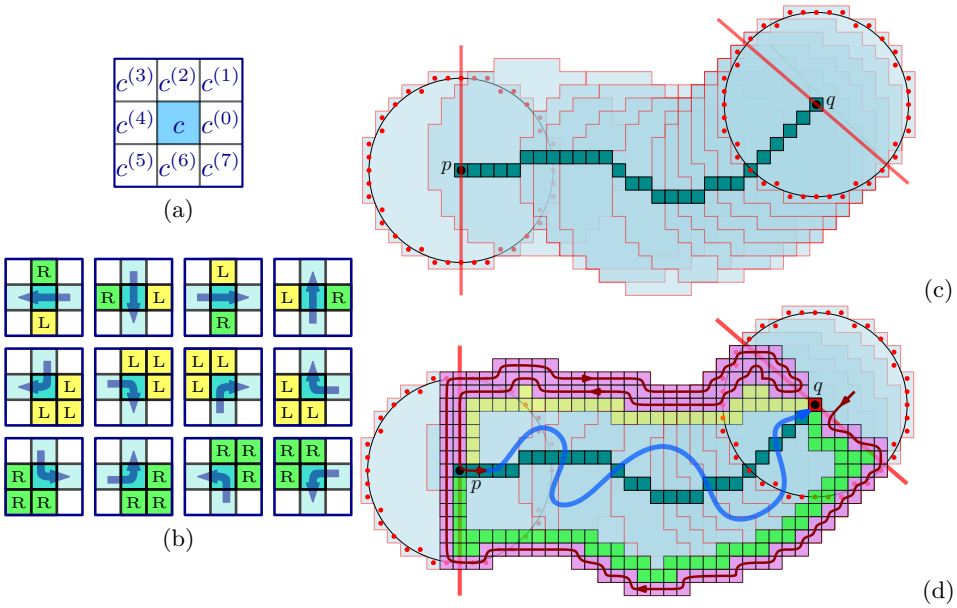

Fig. 2. The CCD and its initialization. (a) 8-N of a cell c. (b) Three types of turns with four combinatorial cases each. The current cell c_i is shown in blue, and the previous and the next cells in faded blue. (c) Minkowski sum (in blue) of a typical (simple and irreducible) digital curve (deep green) from p to q; red lines show the borderlines through p and q for CCD initialization. (d) Cells occupied ($\beta > 0$) by the initialized curve are shown in violet.

The *blocking factor* $\beta(c)$ is a 5-bit number given by the combinatorial arrangement of the occupied and the free cells in $N_1(c)$. The most significant bit of $\beta(c)$ corresponds to c itself, and the other four bits correspond to the four cells lying right, top, left, and below of c in that order. If a cell in $N_1(c)$ is occupied, then the corresponding bit of $\beta(c)$ equals 1, otherwise 0. Thus, $\beta(c) = 0$ implies that ρ is not (yet) passing through any cell in $N_1(c)$. If $0 < \beta(c) < 16$, then c is free but one or more cells in $A_1(c)$ are occupied. If $\beta(c) \geq 16$, then c is occupied.

The *directional label* $\delta(c)$ is used if $0 < \beta(c) < 16$ which takes its value then from $\{\mathtt{L}, \mathtt{R}, \mathtt{B}\}$, with the interpretation: $\mathtt{L}$ = left, $\mathtt{R}$ = right, $\mathtt{B}$ = both left and right, depending on the position of c relative to the direction of traversal of ρ in the cell(s) of $A_0(c)$. We use $\mathtt{X}$ for the initialized value. While the construction of ρ is in progress, blocking factors and directional labels have interim values, which are updated and become final values when ρ is finished.

Initialization of CCD. The initialization of a CCD is illustrated in Fig. 2. The cells c_p and c_q, corresponding to p and q, are obtained first. The initialized curve ρ is assumed to enter $c_q^{(6)}$ from its left edge, and then progresses through the border cells, to finally reach the cell c_p so that C_q has its label $\delta = \mathtt{B}$. By this initialization, c_q is free and has $\mathtt{B}$ as δ-value, whereas all other border cells are occupied, the (actual) random curve starts from p, and the free cells, adjacent to the border cells, have $\mathtt{L}$ or $\mathtt{R}$ as δ-value. While generating the random

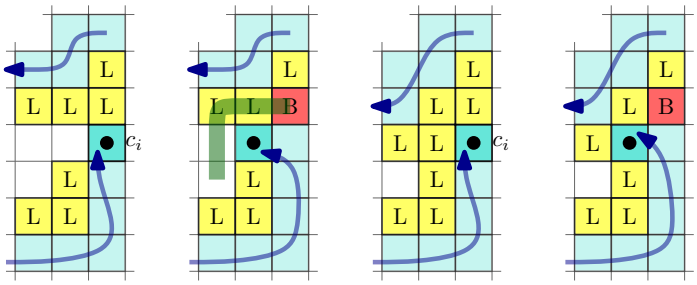

Fig. 3. Distinguishing the formation of a hole (a,b) from an ensuing hole (c,d). **Ensuing hole:** (a) Before formation, all the concerned cells have label `L`. (b) After formation, label of $b_i := c_i^{(2)}$ gets modified to `B`, and a free path exists from each free cell in $E_i \cap N_4(c_{i+1}) := \{c_{i+1}^{(2)}, c_{i+1}^{(4)}, c_{i+1}^{(6)}\}$ to b_i. **Hole:** (c) Before formation, cells have label `L`. (d) After formation, label of $b_i := c_i^{(2)}$ becomes `B`, and a free path to b_i is not possible from $c_{i+1}^{(4)}$ and $c_{i+1}^{(6)}$, as $c_{i+1}^{(3)}$ is blocked.

curve, if some cell c is visited which is adjacent to some border cell, then the corresponding parameters of c are updated accordingly. The initialized and the runtime parameters help advancing the curve in a random and yet 'safe' direction. Clearly, that *virtual* part of ρ lying in the border cells of $\mathbb{D}_h$ is not random, and hence not considered as being a part of the random curve.

Cell Parameters. The *current cell*, which ρ has currently entered, is denoted by c_i $(i > 1)$, unless mentioned otherwise. The cell c_i corresponds to the ith iteration of our algorithm. Parameters β and δ are updated in (appropriate cells of) $A_0(c_i)$, as shown in Fig. 2. Each current cell c_i has a *previous cell*, c_{i-1}, from where ρ has entered c_i, and a *next cell*, c_{i+1}, where ρ will enter next. The cells belonging to the region $\widetilde{N}(c_i) := A_0(c_i) \setminus (A_1(c_{i-1}) \cup A_1(c_{i+1}))$ are labelled in the ith iteration, as illustrated in Fig. 2.

Progressing the Random Curve. From the current cell c_i, the next cell c_{i+1} is (randomly) chosen in such a way that there exists at least one *free path* from c_{i+1} to the destination cell c_q. (A free path from a cell c_i to a cell c_{i+k}, $k > 1$, is given by a sequence of cells, $\rho(c_i, c_{i+k}) := \langle c_i, c_{i+1}, \ldots, c_{i+k} \rangle$, such that each cell in $\langle c_{i+1}, \ldots, c_{i+k-1} \rangle$ is free and distinct, and every two consecutive cells in $\rho(c_i, c_{i+k})$ are 1-adjacent.) A *safe edge* of c_i is a possible exit edge; the algorithm selects randomly one of the safe edges for exit. For the current cell c_i we have the *free region* R_i of all free cells c of the CCD $\mathbb{D}_h$ such that there exists still at least one free path from c to c_q. Similarly, a *blocked region* H is a maximal (connected) region of free cells such that there does not exist any free path from any cell of H to c_q. A cell in H is said to be *blocked*, and edges of a blocked cell are also *blocked*. There exists a free path from the current cell c_i to the destination cell c_q if and only if $A_1(c_i) \cap R_i \neq \emptyset$. (If $A_1(c_i) \cap R_i \neq \emptyset$, then there exists a free cell $c_i^{(t)} \in A_1(c_i)$ lying in R_i. Conversely, the existence of a free path from c_i to c_q implies that at least one cell of $A_1(c_i)$ is in R_i, thus $A_1(c_i) \cap R_i \neq \emptyset$.) As a result, the edge between c_i and $c_i^{(t)}$ is safe if and only if $c_i^{(t)}$ belongs to R_i.

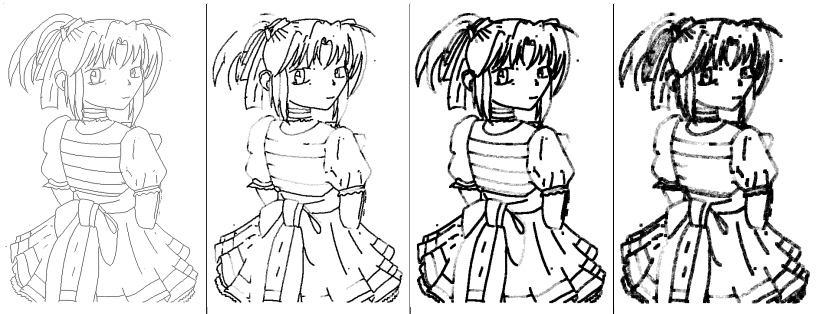

Fig. 4. Effect of varying m versus t on another set, S ($\gamma_{\max} = 255, \gamma_0 = 0$).

Ensuring Simple and Irreducible Property: ρ is allowed to enter and exit a cell at most once. Hence, an exit edge of the current cell c_i cannot be an entry edge of the next cell if the latter is already occupied (using $\beta(c_i)$). Furthermore, a blocked edge cannot be an exit edge. The crux of the problem is, therefore, to decide whether or not an edge of c_i is a blocked edge. Each event of forming a hole is detected based on (changes in the components of the cells in) $A_0(c_i)$. The advantage of detecting such a *hole event* is that, once ρ enters the next cell c_{i+1} from c_i by selecting a safe edge, it can never enter the hole H formed by c_i, since H gets surrounded by occupied cells after it is formed.

Further characterizations of cells in the local neighbourhood of c_i are required to distinguish whether there is a hole event or an event of an *ensuing hole* (Fig. 3). $E_i \subset R_i$ defines an ensuing hole corresponding to c_i if and only if
(e1) there exists $c \in \widetilde{N}(c_i)$ such that $\delta(c, i) = \mathtt{B}$,
(e2) for each $c' \in E_i$, we have that $\delta(c', i) \in \{\mathtt{L}, \mathtt{R}, \mathtt{X}\}$,
(e3) there exists a free path $\rho(c_{i+1}, c_q)$, and for any such path, c is on $\rho(c_{i+1}, c_q)$.
Note that, $\delta(c, i)$ denotes the label of cell c when the current cell is c_i.

Either a hole or an ensuing hole is created if and only if at least one free cell in $\widetilde{N}(c_i)$ gets the label $\mathtt{B}$ as c_i becomes the current cell. The current cell c_i gives rise to an ensuing hole E_i if and only if there exists a free cell $b_i \in \widetilde{N}(c_i)$ with
(E1) $\delta(b_i, i) = \mathtt{B}$;
(E2) there exists $\rho(a_i, b_i) \subseteq A_0(c_{i+1})$ for each $a_i \in E_i \cap A_1(c_{i+1})$. In particular, c_i gives rise to a hole H_i if and only if E1 is true and E2 is false. The proof follows from the combinatorial arguments given in [BPR10].

Final Sketch Creation. For each curve C_h in S, we create m random curves. Note that, S is obtained in our work by Canny edge detection [Can86] and thinning [RK82]. If p and q be the respective start and end points of the curve C_h, then each of these m random curves is made to start from p and end at q. Further, due to the curve-constrained domain, $\mathbb{D}_h$, corresponding to C_h, each random curve strictly lies in $\mathbb{D}_h$. The cells of $\mathbb{D}_h$ are always (re-)initialized for creating each instance of the m random curves corresponding to C_h.

Let $S_h = \left\{C_h^{(z)}\right\}_{z=1}^{m}$ be the set of m random curves corresponding to C_h. Let c_h be a cell of the domain $\mathbb{D}_h$. We maintain a counter, namely $\textsc{count}[c_h]$,

Fig. 5. Results on another image. Top-left: A photograph. Top-right: Product of our algorithm. Bottom: After overlaying on a canvas.

corresponding to each $c_h \in \mathbb{D}_h$. Each such $\textsc{count}[c_h]$ is initialized to 0 before generating the random curves in $\mathbb{D}_h$. Whenever a random curve $C_h^{(z)}$ visits c_h, $\textsc{count}[c_h]$ is incremented. Thus, after all m random curves are constructed in $\mathbb{D}_h$, we get $0 \leq \textsc{count}[c_h] \leq t \ \forall c_h \in \mathbb{D}_h$.

In order to create the artistic curve $\tilde{C}_h$ corresponding to C_h, we use the counter values $\{\textsc{count}[c_h] : c_h \in \mathbb{D}_h\}$. For each $c_h \in \mathbb{D}_h$, the corresponding image pixel is intensified to the value

$$\gamma_{\max} - \left(\gamma_0 + \frac{\textsc{count}[c_h]}{m} \times (\gamma_{\max} - \gamma_0)\right),$$

since we consider 8-bit intensity of the image (with $\gamma_{\max} = 255, \gamma_0 = 0$) as the final output corresponding to the input set S. To achieve an overall darker intensity (as in Fig. 4) in simultaneity with the randomized finish, we scale the colour spectrum to a smaller interval, namely $[\gamma_0, \gamma_{\max} - \gamma_0]$, and measure the pixel intensity by setting γ_0 to an appropriately high value.

3 Results and Conclusion

We have developed the software in Java™ API, version 1.5.2, the OS being Linux Fedora Release 7, Kernel version 2.6.21.1.3194.fc7, Dual Intel Xeon Processor 2.8

Fig. 6. Effect of using mixed pencils. Left: Input sketch. Middle: Product of our algorithm. Right: After overlaying on a canvas. Note that our algorithm uses thick curves in the relevant portion (e.g., nose) and thin curves for small details, which, in turn, creates the desired artistic touch.

GHz, 800 MHz FSB. It has been tested on several datasets containing various digital images of different shapes and forms. Snapshots on a typical set are already given in Fig. 1. The summary of results for a few images is presented in Table 1. From this table, it may be noticed that as the width of pencil-tip increases, the run-time also increases, since it needs a larger number of iterations to create sufficient stroke intensity.

The effect of number of iterations of randomized curve tracing with changing width of the pencil-tip is shown by a set of results in Fig. 4. It shows how the intensity of the pencil stroke increases with increase in m for a given value of t. A proper value of m has to be selected, therefore, for a given pencil to ensure the aesthetic quality of the sketch produced by our algorithm. Finer details, of course, can be captured with a fine-tipped pencil (i.e., having a low value of t), as evident in Fig. 4. Figures 5 and 6 show how our algorithm successfully produce the desired artistic impression—whether the type of input be a line-sketch or a photograph. Figure 6, in particular, shows the usage of mixed pencils (with varying widths of pencil-tips) depending on demand of the local region of interest. When a curve is fairly long, it signifies possibly a strong structural information of the underlying object—demanding a bold stroke from the artist—

Table 1. Summary of simulation results.

Image	w	h	n	t	m	T
houses	480	320	3441	8	10	10.871
houses	480	320	3441	5	5	6.257
nestle	320	480	2579	8	10	9.755
statue	481	321	6200	8	10	11.267
elephants	480	320	6256	5	5	10.306
vase	220	400	5048	5	5	3.333

w = image width; h = image height; n = number of curve points in the input image; t = width/thickness of pencil-tip; m = number of random curves; T = CPU time in seconds for the algorithm to produce the final output.

which is drawn by a thick and bold line in our algorithm. Nevertheless, it retains its non-uniformity of shade, thus giving a crayon-like appeal, wherefore the piece of output gets the artistic finish.

References

[AH96] Auer T., Held M.: Heuristics for the generation of random polygons. In *Proc. 8th Canad. Conf. Comput. Geom.* (1996), pp. 38–44.

[BPR10] Bhowmick P., Pal O., Klette R.: A linear-time algorithm for generation of random digital curves. In *Proc. PSIVT 2010*, 168–173.

[Can86] Canny J.: A computational approach to edge detection. *IEEE Trans. PAMI 8*, 6 (1986), 679–698.

[CAS97] Curtis C. J., Anderson S. E., Seims J. E., Fleischer K. W., Salesin D. H.: Computer-generated watercolor. In *Proc. SIGGRAPH '97*, 421–430.

[Deu10] Deussen O.: Oliver's artistic attempts (random line). http://graphics.uni-konstanz.de/artlike, 2010.

[GG01] Gooch B., Gooch A.: *Non-photorealistic rendering.* A.K. Peters Ltd., NY, 2001.

[KCC06] Kang H. W., Chui C. K., Chakraborty U. K.: A unified scheme for adaptive stroke-based rendering. *The Visual Computer 22*, 9 (2006), 814–824.

[KHCC05] Kang H. W., He W., Chui C. K., Chakraborty U. K.: Interactive sketch generation. *The Visual Computer 21* (2005), 821–830.

[KNC08] Kopf J., Neubert B., Chen B., Cohen M., Cohen-Or D., Deussen O., Uyttendaele M., Lischinski D.: Deep photo: Model-based photograph enhancement and viewing. In *SIGGRAPH Asia '08* (2008), pp. 1–10.

[KR04] Klette R., Rosenfeld A.: *Digital Geometry: Geometric Methods for Digital Picture Analysis.* Morgan Kaufmann, San Francisco, 2004.

[LMHB00] Lake A., Marshall C., Harris M., Blackstein M.: Stylized rendering techniques for scalable real-time 3d animation. In *Proc. NPAR '00*, 13–20.

[MG02] Majumder A., Gopi M.: Hardware accelerated real time charcoal rendering. In *Proc. NPAR '02*, 59–66.

[Mou03] Mould D.: A stained glass image filter. In *Proc. EGRW '03*, 20–25.

[OSSJ09] Olsen L., Samavati F. F., Sousa M. C., Jorge J. A.: Sketch-based modeling: A survey. *Computers and Graphics 33*, 1 (2009), 85–103.

[PSNW07] Pusch R., Samavati F., Nasri A., Wyvill B.: Improving the sketch-based interface: Forming curves from many small strokes. *The Visual Computer 23*, 9 (2007), 955–962.

[RK82] Rosenfeld A., Kak A. C.: *Digital Picture Processing, 2nd ed.* Academic Press, NY, 1982.

[RMN03] Rudolf D., Mould D., Neufeld E.: Simulating wax crayons. In *Proc. PG '03*, 163–172.

[VB99] Verevka O., Buchanan J. W.: Halftoning with image-based dither screens. In *Proc. Graphics Interface '99*, 167–174.

[VG91] Velho L., Gomes J. d. M.: Digital halftoning with space filling curves. *Proc. SIGGRAPH 91*, 81–90.

[ZSSM96] Zhu C., Sundaram G., Snoeyink J., Mitchell J. S. B.: Generating random polygons with given vertices. *Computational Geometry Theory and Applications* (1996), 277–290.