Self-Avoiding Random Walks Yielding Labyrinths

Gary R. Greenfield
Department of Mathematics & Computer Science
University of Richmond
Richmond, VA 23173, USA
ggreenfi@richmond.edu

Abstract

We modify a self-avoiding random walk model based on curvature by Chappell. This will lead us to the discovery of autonomously constructed drawings that often yield labyrinths. Labyrinth formation occurs when the sensing and avoiding feedback loop that modulates curvature promotes curve following. We add rendering effects in order to stylize our labyrinths and visualize certain aspects of our self-avoiding random walk behavior.

Introduction

Points that execute self-avoiding walks occur in the physics and chemistry literature because of their application to protein folding and other polymer-related problems [8]. Such self-avoiding walks are often implemented using lattices [11], and in this case it is known that there are algorithms for infinite self-avoiding walks [6]. In order to generate smooth self-avoiding random walks, Chappell introduced a model for random walks in the plane that depends on curvature [3]. By incorporating rules based on readings from sensory apparatus — two sets of "feelers" attached to the point executing the walk — and adding rendering effects to the curve the point traces, Chappell generated many examples of drawings of points executing prolonged self-avoiding and self-following walks. In an effort to reverse-engineer certain aspects of the model Chappell implemented, we also developed a self-avoiding random walk model based on curvature, but one which to our surprise often produced labyrinths. In this paper we provide background and details as well as some labyrinth examples.

Chappell's Method

The random walk algorithm based on curvature introduced by Chappell [3] is parametrized by arc length s. This means the smooth curve the point traces while executing the random walk is approximated by successively drawing very short line segments of length Δs. Assume after traveling a distance s, the point has tangential angle (i.e., heading) $\theta(s)$, curvature $\kappa(s)$ and position $(x(s), y(s))$. To determine the position of the point when the curve length is $s + \Delta s$ first update the curvature and heading values using:

$$\begin{aligned} \kappa(s + \Delta s) &= \kappa(s) + \kappa_0 X(s), \\ \theta(s + \Delta s) &= \theta(s) + \kappa(s + \Delta s)\Delta s, \end{aligned}$$

where $X(s)$ is a stochastic random variable assuming values $+1$ and -1 and κ_0 is a "small" constant. Then determine the new position by letting:

$$\begin{aligned} x(s + \Delta s) &= x(s) + \cos(\theta(s + \Delta s))\Delta s, \\ y(s + \Delta s) &= y(s) + \sin(\theta(s + \Delta s))\Delta s. \end{aligned}$$

The example Chappell gives of the random walk produced using his model is shown in Figure 1. However, because the only update formula for θ that ever appears in his paper is:

$$\theta(s + \Delta s) = (1 - \eta)\theta_{RC} + \eta(\theta(s') + \varphi),$$

we can not be sure if the simplified version we have given here for the θ update should omit Δs as a multiplier or, possibly, use some other scale factor. Before returning to this question, consider Chappell's description of how he obtains a self-avoiding walk (SAW):

> To generate a self-avoiding curve, I place "antennae" on the moving point that sense when the path is about to be crossed. ... If the left antenna crosses the path, then the point executes a 180° reversing turn to the right. If the right antenna crosses the path, then it reverses direction by turning to the left. Reversing turns are followed to completion and are not interrupted or modified if one of the antennae strikes the curve again, although a second set of antennae described below can override the turn. The radius and angle of the reversing turn are each held constant to provide a unifying visual motif throughout the design. The walk is terminated when the the self-avoiding rule fails and the point collides with its path.
>
> In order to reduce the likelihood of the walk becoming trapped, the point is given a second set of smaller antennae that can override the longer antennae when more immediate dangers are detected. For example, if the walker is executing a reversing turn to the left and the the left overriding antenna touches the path, then a new reversing turn to the right is initiated, overriding the original turn. ...
>
> On each step of a SAW, the antennae must search the path for possible collisions. In an exhaustive search, the number of computations required to generate a SAW with N steps scales as N^2. ...

Although the description is not fully fleshed out, this excerpt motivates Chappell's example of the self-avoiding random walk shown in Figure 2.

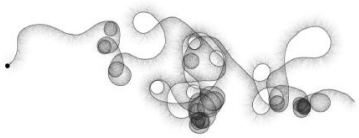

Figure 1: *Chappell's example of a random walk using a model based on curvature and parametrized by arc length. Printed with permission.*

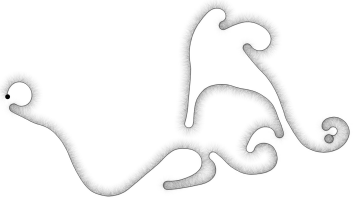

Figure 2: *Chappell's example of a self-avoiding random walk. Printed with permission.*

The reason Chappell uses such a complicated update formula for θ is to obtain a parametrized *family* of self-avoiding and self-following walks. Again, quoting:

θ_{RC} is the tangential angle produced by random-curvature motion, $\varphi = 0$ or π depending on whether the point follows the path "upstream" or "downstream," and the parameter $\eta \in [0, 1]$ controls the contribution of random motions. When $\eta = 0$, the point follows a random-curvature walk and is not influenced by the presence of its path. When $\eta = 1$, the point is "pinned" to the curve it is following.

We refer readers to Chappell's paper for further examples. Unfortunately, this description was too complicated to reverse engineer (see [3] for the definition of s'), and since several parameters for the model were not explicitly given and we wished to avoid searching a stored list containing every point previously visited in order to detect curve contact, we were forced to simplify and resort to our own devices. Our original goal was to develop a platform for simultaneously executing multiple self-avoiding random walks. While reproducing random walks similar to the one in Figure 1 was easy, it turned out that the incessant parameter tuning and never ending design modifications we labored over in order to try and keep self-avoiding random walks from prematurely terminating diverted us from our original goal, and instead led to a sequence of parameter settings and a design that, to our surprise, was capable of yielding *autonomously* constructed labyrinths. This is the model we give the details for below.

Our Method

We chose an update increment of $\Delta s = 0.4$ so that it takes multiple updates in order for a point to cross an underlying pixel of a $w \times h$ pixel canvas corresponding to the $[w, 0] \times [0, h]$ region of the plane where the point is executing its walk. In this way pixels can be "time-stamped" with the times (really update iteration counts) they were visited as well as the curvature and heading values they were using when those events occurred. Thus a point's feelers can examine pixels they come in contact with to see whether the point executing the random walk has previously visited them or not. The pixels at the boundary of the region are marked with a time-stamp of zero so the point will also be able to avoid them. These boundary pixels are given a visited curvature value of zero to reflect the fact that the boundaries are straight line segments.

The point starts out in the center $(w/2, h/2)$ of the region of interest with an initial heading $\theta(0) = 0$ so that it is headed in the positive x direction and an initial curvature $\kappa(0) = 0.02$. It moves until either it is halted or 86,250 Δs updates have occurred. If it is not in the midst of what we call either "turning" or "veering", then every *thirteenth* Δs update the curvature is modified by setting:

$$\kappa(s + \Delta s) = \kappa(s) + \kappa_0 Y(s),$$

where $\kappa_0 = 0.02$ and $Y(s)$ is a random number in the interval $[0, 1]$, otherwise $\kappa(s + \Delta s) = \kappa(s)$. The reason for spreading out the curvature updates is to allow the point time to draw a visible curve segment using each new value. When a turn or a veer completes, the curvature is reset using:

$$\kappa(s + \Delta s) = -\kappa(s)/3 + \kappa_t/2 + \kappa_0 Y(s),$$

where κ_t is a curvature value that was saved when the veer or turn commenced and κ_0 and $Y(s)$ are as before. Except for the first five Δs updates of a veer or turn, where the curvature remains fixed in order to allow the turn or veer to gain a foothold, after every Δs update the point senses to see if any avoidance measures are called for. This is done in two stages. First, sensory data from the point's antennae which consist of two feelers located 15° to either side of the point's heading and extending out for twenty units are considered. Second, a sensory field extended out in all directions to nearby pixels is considered.

The antennae can return sensory data from a distance five units away from the point out to twenty units away from the point. If the *first* pixel encountered that is marked as previously visited is sensed within seven

units of the point, the point halts because the walk has somehow gotten too close to a previously drawn part of the curve; if it is within fifteen units of the point, assuming a veer is not already in progress, it initiates a veer; otherwise, assuming a turn is not already in progress, it initiates a turn. To initiate a turn, the point saves the current curvature value $\kappa(s)$ to κ_t and sets $\kappa(s + \Delta s) = \pm 0.15$ where the sign chosen is *opposite* to that of the current value. Initiating a veer is more complicated because a turn may already be in progress! If a turn is in progress, whence there is already a saved value of curvature, the turn is canceled and the curvature is set so that $\kappa(s + \Delta s) = \pm 0.45$ where the sign chosen *matches* that of the current value. If a turn is not in progress, the point saves the *sensed* value of the curvature to κ_t and the curvature is set so that $\kappa(s + \Delta s) = \pm 0.45$ where the sign chosen is opposite to that of the sensed value. Turns complete after twenty-five Δs updates, veers after twenty.

The rationale for antennae is to detect the previously drawn curve in time to complete a gentle turn by turning away without intersecting it, but with the understanding that it may be necessary to increase the rate of turn (and try to follow along parallel to a previously drawn part of the curve) if the point still gets too close or otherwise encounters the curve at an odd angle. That being said, previously drawn parts of the curve can still be dangerously close to the point without being detected by antennae. Thus, in the second stage a sensory field extending throughout a 15×15 unit neighborhood of the point is invoked to search for previously drawn parts of the curve. Again, if a visited pixel is encountered within seven units the point is halted because it is deemed too close to allow the point to proceed. However, because the neighborhood is rectangular, rarely, a visited pixel may be encountered that is further away than that but in an odd location relative to the antennae. In this case one final evasive maneuver is attempted. The heading is changed so that $\theta(s + \Delta s) = \theta(s) \pm \pi/10$ where the sign is chosen to match that of the heading value saved at the sensed pixel.

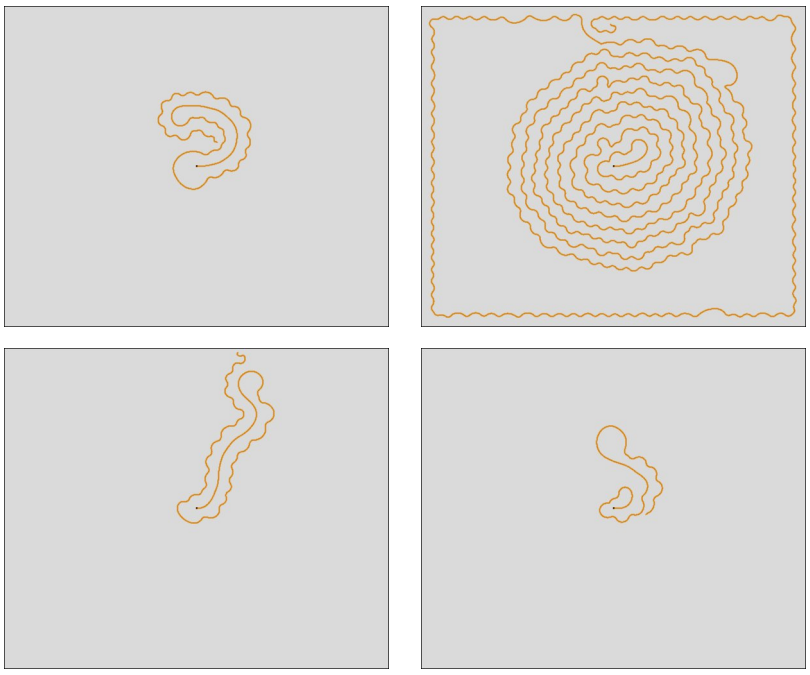

Figure 3: *Four examples of self-avoiding random walks on a* 600×500 *pixel canvas exhibiting the different types of "behavior" that can arise.*

Before giving higher resolution examples with added rendering effects that yielded labyrinths, we consider some lower resolution examples to help illustrate why it takes so much effort to induce an autonomous, self-avoiding walk using our self-avoidance curvature model. Figure 3 shows four examples of self-avoiding

walks executed on a 600×500 pixel canvas. They indicate that most often the walk is terminated because the point approaches a previously drawn section (or the boundary) by quickly curling back towards it and facing it almost head on. The exception is the drawing at the lower right where the point approaches a previous section almost parallel to it but still too close.

Labyrinth Examples

It would be sheer folly to try and survey the literature on mazes and labyrinths. Just to give an inkling of what has appeared recently in the mathematics and art literature, we cite an ongoing series of papers by Verbiese (the latest being [12]), tiling related work by Bosch et al. [1], visual mathematics related work by Fenyvesi et al. [4], and an artist's tie-in with the Jordan curve theorem by Ross and Ross [10]. Similarly, the non-photorealistic rendering literature is rife with examples [2] [9].

As our preliminary tests indicated, in some instances when the self-avoiding random walk is able to continue long enough, but halted before it reaches the boundary, the point will spiral out from the center and form a labyrinth. To generate further examples we enlarged the canvas to 1000×1000 pixels, extended the length of the antennae to forty units to get better separation between curve elements, and added rendering effects by thickening the curve and "feathering" it. Feathering was accomplished by drawing normals extending out from the curve after every few Δs updates whose length was proportional to the curvature. Successive positions of the point were used to construct a vector approximating the tangent vector to the curve whose perpendicular vector then provided the normal vector to the curve. Figure 4 gives two examples. We observe that longer antennae make it easier for the curve to reverse direction. This is evidenced by focusing on the thinnest sections of the curve where the point is wandering "freely" not subject to turning or veering.

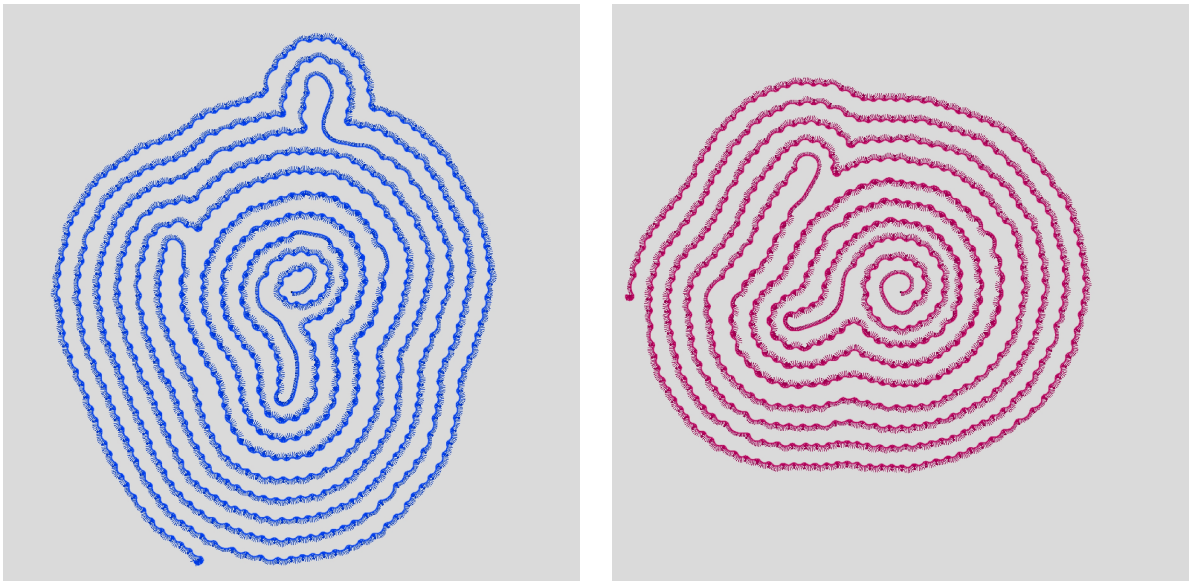

Figure 4: *Digital prints, 10" × 10", 2014. Left: SA Labyrinth #5223. Right: SA Labyrinth #8352. Copyright Gary Greenfield.*

Future Work

The rationale for using a random-walk model based on curvature is to draw smooth curves. To add self-avoidance the problem becomes to detect contact with the curve in time to turn in an appropriate direction. We are still interested in multiple points executing self-avoiding random walks while simultaneously avoiding each other. This is problematic using only two sets of antennae as Chappell does, or one set of antennae and a sensing field as we have done. However, we have been able to use our curvature set-up to make drawings where multiple points avoid each other but not themselves [5]. To add self-avoidance in this instance, it may be necessary to design sensory apparatus that can detect curve contact via a sensory field extending over an arc, or use sets of non-uniform, asymmetric feelers in a manner similar to the way Machado and Pereira have for their artificial ants [7].

References

[1] Bosch, R., Fries, S., Puligandla, M. and Ressler, K., From path-segment tiles to loops and labyrinths, in *Bridges 2013 Conference Proceedings*. G. Hart et al., eds., Tessellations Publishing, Phoenix, AZ, 2013, pp. 119–126.

[2] Bosch, R., Chartier, T. and Rowan, M., Minimalist approaches to figurative maze design, preprint.

[3] Chappell, D., Taking a point for a walk: pattern formation with self-interacting curves, in Bridges 2014 Conference Proceedings. G. Greenfield et al., eds., Tessellations Publishing, Phoenix, AZ, 2014, pp. 337–340.

[4] Fenyvesi, K., Jablan, S. and Radović, L., Following the footsteps of Daedelus: labyrinth studies meets visual mathematics, in *Bridges 2013 Conference Proceedings*. G. Hart et al., eds., Tessellations Publishing, Phoenix, AZ, 2013, pp. 361–368.

[5] Greenfield, G., Avoidance drawings evolved using virtual drawing robots, in *Proceedings EvoMUSART 2015*, A. Carballal, C. Johnson, and J. Nuno, eds., Springer-Verlag, Berlin, 2015, in press.

[6] Kremer, K. and Lyklema, J., Infinitely growing self-avoiding walk, *Phys. Rev. Lett.*, 54, 1985, pp. 267–269.

[7] Machado, P. and Pereira, L., Photogrowth: non-photorealistic renderings through ant paintings, in *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion — GECCO 2012*, T. Soule, ed., ACM, Press, New York, NY, 2012, pp. 233–240.

[8] Madras, N. and Slade, G., *The Self-Avoiding Walk*, BirkHauser, Boston, MA, 1993.

[9] Pedersen, H. and Singh. K., Organic labyrinths and mazes, in *NPAR '06 Proceedings of the 4th international symposium on non-photorealistic animation and rendering*, ACM Press, New York, NY, 2006, pp. 79–86.

[10] Ross, F. and Ross, W., The Jordan curve theorem is non-trivial, *Journal of Mathematics and the Arts*, 5:4, 2011, pp. 213–219.

[11] Vanderzande, C., *Lattice Models of Polymers*, Cambridge University Press, New York, NY, 1998.

[12] Verbiese, S., Amazing labyrinths, further developments IV, in *Bridges 2014 Conference Proceedings*. G. Greenfield et al., eds., Tessellations Publishing, Phoenix, AZ, 2014, pp. 483–484.