Artistic Tessellations by Growing Curves

Hua Li* David Mould†
Carleton University

(a) *Tiffany glass* **(b)** *Craquelure* **(c)** *Mosaics* **(d)** *Macaroni art*

Figure 1: *From curves to artistic tessellations.*

Abstract

In this paper we propose to tessellate a region by growing curves. We use a particle system, which flexibly provides good control over the final effects by variations of the initial placement, the placement order, curve direction, and curve properties. We also propose an automatic image-based mosaic method which has good texture indication, using a smoothed vector field to guide particle movement. The final irregular tessellation simulates stained glass where the elongated curved tiles suggest the content of highly textured areas. We give some additional applications, some of which resemble naturally occurring irregular patterns such as cracks and scales. We also notice that stacking a set of curves in a structured way can produce the illusion of a 3D shape.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation;

Keywords: Tessellation, particle system, natural patterns, mosaics

1 Introduction

Human are very good at abstracting similar structure from natural textures such as fish or snake scales, cracks, or bark. This is because the human brain contains mechanisms that rapidly define regions having common texture and color [Regan 2000]. We call these "natural patterns" if they have irregular tessellations but significantly similar other structural properties in abstraction.

The existence of irregularities and randomness reminds us of the beauty of nature. Some artists also like to introduce unorganized elements into their work. Typical examples can be found in Tiffany glass or in stained-glass mosaics. Figure 2 (a) shows a Tiffany glass example that uses irregular tiles to represent objects. The curved boundaries are very jagged and rough. The artist Barbara Keith used a lot of uneven tiles of different shapes to express the hen in Figure 2 (b). In particular, the elongated tiles show the flow of the feathers beautifully. However, it is very difficult to automatically generate a tessellation that not only contains texture indication but also possesses organized irregularities.

(a) *Tiffany glass* **(b)** *Stained-glass mosaics*

Figure 2: *Two typical art work showing the irregular tessellation. (a) By an anonymous artist from Flickr.com; (b) By Barbara Keith.*

There is a very long history of research on creating regular tessellations. The mainstream methods are region-based methods, typically based on Voronoi diagrams [Okabe et al. 1992]. However, it is difficult to adapt such methods to represent natural patterns with elongated, irregular, or curved tiles; control over site placement is insufficient to create the desired tile shapes. It is common to see the use of regular tiles (e.g., square or hexagonal), especially in simulations of traditional mosaics [Hausner 2001; Elber and Wolberg 2003; Faustino and de Figueiredo 2005; Smith et al. 2005; Liu et al. 2007; Liu et al. 2010]. High-frequency textures are not often retained in earlier results due to tile shape constraints and style requirements.

In non-photorealistic rendering (NPR), researchers dealt with images either with region-based methods like mosaics or with stroke-based methods [Hertzmann 2003] separately. However, they rarely connect them together. The main contribution of this paper is an idea to tessellate a region by adding curves, either sequentially or in parallel, starting from a single point or from a distribution of points. The key is that, instead of tessellating with individual tiles (or regions), we build the boundary of each tile by the growth of curves. Our implementation employs a particle system where particle trails form curves. Control over the variations of the initial placement, the placement orders, curve directions, and curve properties provides

*e-mail: hli1@connect.carleton.ca
†e-mail:mould@scs.carleton.ca

a lot of possibilities for creation of irregular or curved tiles. Another contribution is to propose an automatic mosaic method with good texture suggestion. The application to an image shows similar effects to stained-glass mosaics. We also expand on the basic idea to present both abstract and natural patterns, introducing two variations, a splitting technique and a stacking technique. Figure 1 shows four irregular tessellations obtained by our system for different artistic styles: an abstract for Tiffany glass, craquelure on an oil painting, stained-glass mosaics with good texture indication, and an example of macaroni art.

2 Previous Work

Since Hausner first introduced Centroidal Voronoi Diagrams (CVDs) to simulate mosaics [Hausner 2001], numerous researchers [Elber and Wolberg 2003; Faustino and de Figueiredo 2005] used similar techniques to pack similar-shape objects to express image information. They were usually based on different emphasized features such as intensities or edges either manually or with user assistance. Recently, Liu et al. [2010] solved the mosaic simulating problem in a global energy optimization framework with graph cuts. However, usually square or hexagonal tiles are used in those applications and the goal for optimization based on shape constraints does not address texture. To introduce tiles of arbitrary shape into mosaics, some researchers treated the mosaic problems as a packing problem [Kim and Pellacini 2002] or area-based CVD [Smith et al. 2005]. However, they used regular or near-regular tiles (such as polygons). Less regular tileswere created by allowing tiles to overlap and then cutting the overlap [Di Blasi and Gallo 2005; Battiato et al. 2006; Orchard and Kaplan 2008]; the use of initially square tiles limited the irregularity. Different from Voronoi-based methods, Mould [2003], aided by operators from mathematical morphology, and later Brooks [2006], using a unique region-merging tool, represented stained glass by different segmentation shapes. However, they did not attempt to treat highly textured structure for regions containing, for example, hair or feathers.

Real stained-glass mosaics usually prefer showing the large-scale flow of the tiles for such textures. Most mosaic-making methods paid attention to object boundaries, usually aligning with edges, both in terms of position and orientation. Recently, Kyprianidis and Kang [2011] simulated similar directional image features very well, abstracting using directional shock filtering. We want our mosaics to have similar high quality in texture abstraction but with different irregular shapes.

Particle systems [Reeves 1983] are widely used in simulating natural phenomena in graphics. Miyata et al. [Miyata et al. 2001] performed particle simulation with proximity-based forces for square packing. They presented some organic texture such as reptile skin and scales. Work close to ours was done by Xie et al. [2010], who proposed an interactive sketch-based system for oriental brush strokes on complex shapes. Our curve tracing algorithm is similar to their work but is automatic.

3 Tessellations by Growing Curves

Instead of thinking of tessellation as packing individual primitives into a region, we could build a tessellation by filling a region with curves. The curves themselves are the tile boundaries. We describe our tessellation as follows.

Process 1. *Given a region and a direction field, and also given a starting point or a starting distribution, a tessellation is formed by the growth of a set of curves, growing either sequentially or in parallel. Each curve grows until it stops either by reaching another curve intersection or the region boundary. If the length of a curve is too short or the curve passes too close to previous curves, the curve is removed.*

The adjustments for the initial starting assignment, the order of the growth, the orientation of the growth, and the properties of the curves (such as curvature, arc length, and thickness) control the final effects of the partition. We strive to create tessellations that have good visual quality. Wong et al. [1998] demonstrated that repetition, balance, and conformance to geometric constraints are three elements to the perception of order. We take their advice and we also intend to introduce irregularities and randomness into the partition to make our simulation appear natural. The following are some general principles we adopt for our process.

- Curves have similar properties along their length, such as curvature; starting orientations are also similar. Enforcing this produces similar region shapes, providing a sense of unity.
- We do not allow short curves to be used and we do not allow the spacing of two curves to be very close.
- We enforce a minimum spacing when we assign starting points.
- Irregularities are attained by putting some random elements into the curve properties and the curve starting orientation; further apparent randomness derives from the uncertainty in intersection locations.

The curve growing process can be either sequential or parallel. In the sequential case (S-Method), we process curves one by one: another curve is created when the previous one is completed. In the parallel case (P-Method), multiple curves are initialized simultaneously and then grow incrementally in parallel.

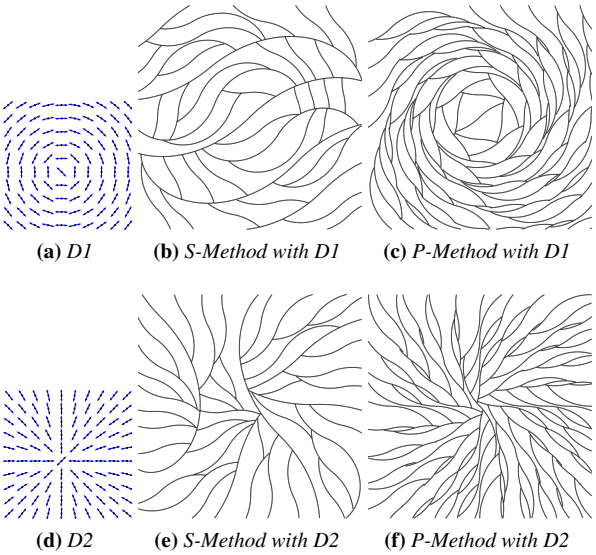


Figure 3: *Four examples from our basic idea. Left: Two direction assignments (D1 and D2); middle: the sequential method; right: the parallel method.*

We show examples from both our sequential method and our parallel method in Figure 3. The tiles in those tessellations are not regular, but curved and elongated. Globally, the tessellations have some similarities because they are using the same curve generation process and the same two direction fields. However, some differences appear because of the order of operations. The parallel method better preserves the large-scale trend from the initial distribution, since curves are generally shorter and hence do not have time to vary much from their initial direction. Spatial control is only possible through the initial distribution. Conversely, the sequential method

has more flexibility of spatial adjustment since logic can be applied at each individual curve placement, taking into account all previous curves. Doing this often makes the order of curve placement apparent, which is sometimes undesirable, although for some patterns such as cracking it is a useful effect.

Figure 4 shows a suggested process for the S-Method. The first curve begins at a random point and grows in two opposite directions. Next, we create and maintain a distance map, storing the distance of all locations to the nearest point either on a curve or on the region boundary. We next generate a curve beginning at the point of maximum distance; in this way we avoid the narrow spacing due to close placement. The process iterates, repeatedly growing the next curve and updating the distance map, stopping when the maximum distance value is below a threshold.

Figure 5 then shows an example of the P-Method. This example begins with a 5×9 grid of points. Each iteration, the curves are grown in a fixed time step. The curve will stop its growth when it meets with other curves or the region boundary. After all curves stop growing, the process ends.

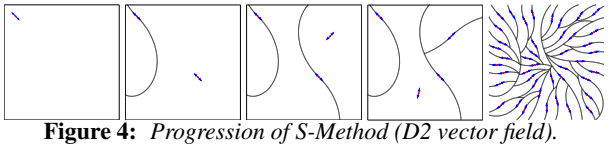


Figure 4: *Progression of S-Method (D2 vector field).*

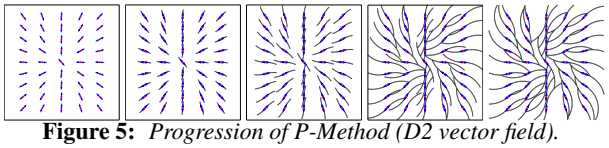
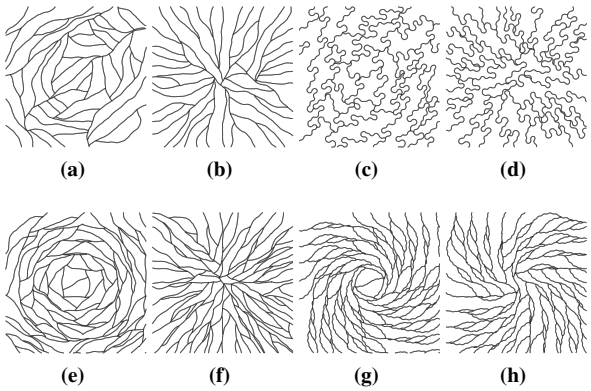


Figure 5: *Progression of P-Method (D2 vector field).*

In principle, any method for curve generation can work for this strategy. We propose to use a particle system [Reeves 1983] as our curve generator in this paper. We use a physical simulation implemented with forward Euler integration. Each time step ($\Delta t = 0.01$), the particle system updates a new position x from previous velocity v_0 and previous position x_0 based on the dynamics. The sequence of positions constructs a curve. The key calculations are as follows: $a = F/m$ for a force F, $v = v_0 + a \times \Delta t$, $x = x_0 + v \times \Delta t$, and we use unit mass $m = 1$. We use different force configurations for different purposes. For mosaic styles, we read $\overrightarrow{F}$ from a vector field; for other abstracts and natural patterns, we use the Lorentz force, previously used to generate magnetic curves [Xu and Mould 2009]. The examples in this section were generated using magnetic curves; details appear in the following sections. Figure 6 shows another group of tessellations from the same strategy as Figure 3 but using curve variations. They all globally maintain the same directional impressions as Figure 3. However, they contain further small-scale details due to the variations in curve properties.



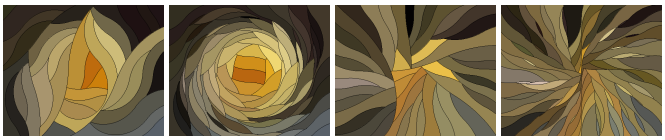
(a) **(b)** **(c)** **(d)**

(e) **(f)** **(g)** **(h)**

Figure 6: *Curve variations. Top: S-Method ; Bottom: P-Method.*

4 Texture Indication for Mosaics

Next, we will use an image to guide our tessellation. Simply using our existing tessellations from the previous section and assigning the average color to each tile, we can create interesting abstractions, shown in Figure 7. Despite using tessellations unrelated to the image content, the colors give some indication of the original lotus image, shown in Figure 8.



(a) **(b)** **(c)** **(d)**

Figure 7: *Abstraction from Figure 3. (a) S-Method with D1; (b) P-Method with D1; (c) S-Method with D2; (d) P-Method with D2.*

Figure 8: *A lotus and its vector field.*

These abstract images contain some visual interest. Nonetheless, we are more interested in presenting representational mosaics, and especially in using flow to indicate texture. The particle now approximately traces a streamline of the vector field. Spatially, particles are not generated if their starting positions are too close to previous curves, say within a distance s_{min}.

The vector field is calculated using the edge tangent flow (ETF) [Kang et al. 2007], in which the vectors are perpendicular to the image gradient. The initial force $\overrightarrow{F}$ is read from the vector field; two particles with opposite forces are created at each separate location with initial velocity zero. Possible initial point locations are generated using a structure-based stippling method [Mould 2007]. Particles proceed in discrete timesteps, with position and velocity updated with forward Euler. We maintain a separate timer (A) to tell us when the force should be updated by reading from the vector field (e.g., we might update after 400 timesteps when $A = 4$.). We maintain a discrete map to detect collisions between particles and other curves: a particle stops when its position is occupied, any other position in its 8-neighborhood is occupied, or it reaches the boundary.

We have an ambiguity about the force direction, resolved by inspecting the dot product of the current and previous force vectors. Given a current force $\overrightarrow{F}_1$ and a previous force $\overrightarrow{F}_0$, we check that $\overrightarrow{F}_0 \bullet \overrightarrow{F}_1 > 0$. If this is not satisfied, we flip the direction of $\overrightarrow{F}_1$. In this way, we prevent the curves from suddenly doubling back on themselves. For a similar reason, we do not use the original vector field from the image to do the calculations: we smooth the field to remove noise and also to strengthen the main trend in textured areas. The smoothing is done using Kang et al.'s bilateral approach [Kang et al. 2007], which smooths among the vectors with similar orientations under a 9×9 window. The goal is to maintain good texture structure. Figure 9 shows a comparison between different degrees of smoothing. Figure 9 (a) is a result without using smoothing, which is very noisy. We see progressively smoother curves from (a) to (c), and consider the smoother curves, and corresponding tessellations, to be of higher quality for our purposes.

(a) *Unsmoothed* **(b)** *Smoothed twice* **(c)** *Smoothed 5 times*

Figure 9: *The difference in smoothing ($A = 4$).*

Figure 10: *(a) Far away from the field. $A = 8$ with smoothing 5 times; (b) Son et al.'s line drawing.*

We also adjust A to control the degree of faithfulness to the field. With larger A, the particle diverts more from the original field. Figure 10 (a) shows this effect. Compared with Figure 9 (c), it looks like straight lines. However, even if it is not faithful to the field it can imply the underlying texture. Figure 9 (c) and Figure 10 (a) also remind us of line drawings; Son et al. [2007] used a similar vector field to draw line arts (shown in Figure 10 (b)) but ours fill the entire space to maintain texture information. Even without adding color information, we can clearly see the flower.

Different initial distributions also affect our final effects. Figure 11 compares a random distribution of 9,803 stipples with a better distribution from structure-aware stippling [Mould 2007] with the same stipple count. In Figure 12 we see that the results (b) and (c) are better than the one without consideration of structure details; (c) shows intensity indication by using $s_{min} = 2 + 5 * I/255$ to constrain the minimum spatial separation, where I is the intensity value.

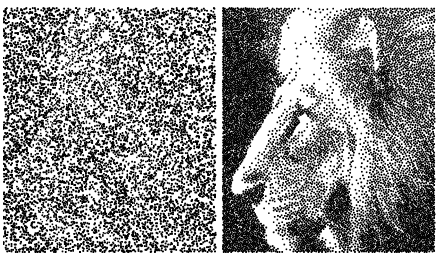
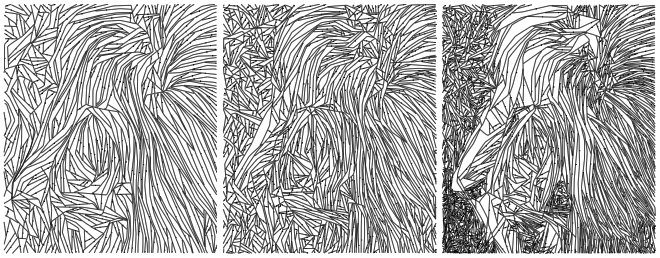


Figure 11: *A random distribution and a better distribution from Mould's stippling.*

In Figure 23 we color each tile by using the average colors of the original image region. Our results nicely show the feather texture for the eagle and the furry texture for the lion by using elongated curved tiles, which also resembles the Tiffany glass in Figure 2 (a). If we think the tiles are too big and want a similar effect to the stained-glass mosaics in Figure 2 (b), we can do further subdivision. After we have long parent curves, we emit new particles along these curves, provided that the arc length exceeds a threshold (we use 50 here.). Also to maintain the spatial characteristics, the generation is ordered from the start of the curve onward; the force directions for the child particles are always perpendicular to their parent force directions. The new particles alternate the sign of their initial directions, thus ensuring that tiles on both sides of the parent curve are subdivided. Two highly-textured examples are shown in Figure 28. Our results gracefully follow the flow trend of the texture.



(a) **(b)** **(c)**

Figure 12: *The difference in distribution with the same stipple count. $A = 8$ with smoothing 5 times. (a) Randomly; (b) Structure-aware stippling without intensity indication; (c) Structure-aware stippling with intensity indication.*

We also apply our method to macaroni art, an art style typically used by children, in which different objects (say, dry pasta) are pasted onto a canvas to form an image. Macaroni examples are shown in Figure 24, where 3D cylinders are placed along the curves, POV-Ray is used to render them, and the resulting images show their constituent curves very prominently.

5 Abstract and Natural Patterns

Our basic method is very appropriate for creating abstracts directly, as we showed in Section 3. In this section, we expand on the basic idea to create both purely abstract patterns and some textures resembling structures in nature such as cracks, scales, and rivers. We also exploit our method to create an arrangement of curves whose apparent occlusions produce an illusion of an ambiguous 3D scene. All the images in this section are built using the sequential method, since it is easier to control both global and local effects. We also make use of magnetic curves for the force calculation, as we are more interested in the artistic stylization and not in the preservation of an input image. A magnetic curve, essentially, is the trail of a particle with charge q moving in a magnetic field $\overrightarrow{B}$, thus experiencing the Lorentz force. We use a constant $\overrightarrow{B} = \{0, 0, -1\}$, which generates a 2D curve forcing the particle to move in the xy plane.

$$\overrightarrow{F} = q\overrightarrow{v} \times \overrightarrow{B} \tag{1}$$

The value of the charge q controls the curvature of the curve. If we flip the sign of the charge, the curvature is also reversed. The direction of the initial velocity provides our orientation control over the curve. We use $q = s * f(t)$ to control the overall curve shape, where s is a parameter to adjust the curvature magnitude. In the following, if we do not specify s explicitly, it means $s = 1$. The curve in Figure 3 uses $f(t) = (500 - t)^{0.8}$, called curve type I.

5.1 Splitting Technique

We introduce a splitting technique to represent both abstract and natural patterns, similar in spirit to the work by Federl [2003], but not based on physics. We randomly place an irregular trail in the region first. We propose two controls for the splitting. One suggestion is that if the curve exceeds the length threshold, it emits a few new particles evenly at Δl length (see Figure 13). The new particles are spawned on alternating sides of the parent curve. Their direction is rotated from the tangent direction of the parent curve with the same or similar angles. Each particle travels for an interval, until a sufficient length of the curve is attained. Another suggestion is that, after having the first curve, the process repeats for a known number of stages, each time emitting a fixed number (n) of

particles along the previous curve and only stopping when the maximum number of stages is reached. The random irregular curve (as curve type II) is generated by randomly choosing the sign of the charge q while $f(t)$ returns a random value uniformly taken from the interval $[0.00001, 0.1]$. This curve type is used in Figure 6 (a), (b), (e) and (f) as well.

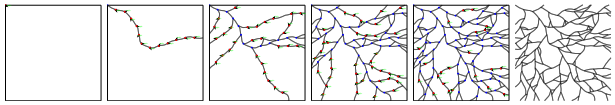
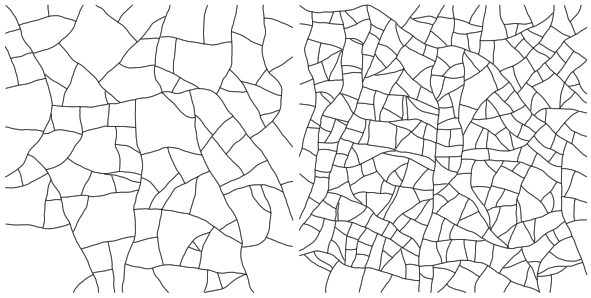
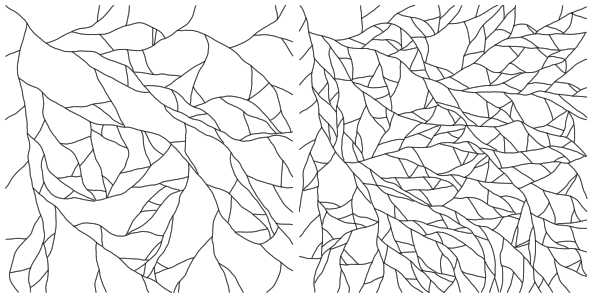


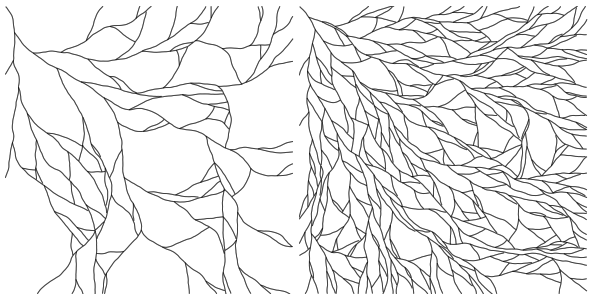
Figure 13: *The process for our splitting under length control.*



(a) 90°, *sparse* **(b)** 90°, *dense*



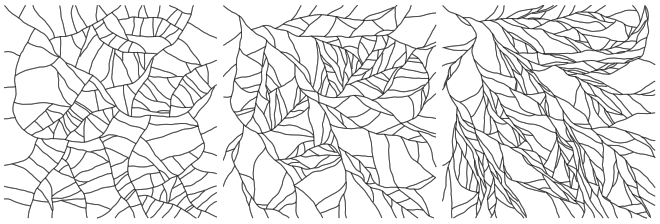
(c) 45°, *sparse* **(d)** 45°, *dense*



(e) 30°, *sparse* **(f)** 30°, *dense*

Figure 14: *Splitting I from length control and angle control (curve type II). Left: $\Delta l = 100$; Right $\Delta l = 50$.*

Figure 14 demonstrates the splitting results under the length control. Sparse versions (larger Δl) are shown on the left side, while dense tessellations (smaller Δl) are shown on the right side. Figure 15 demonstrates the splitting results under stage control (n = 7). This process introduces a lot of irregularities and randomness into the final effects. Sometimes the even spatial distribution is lost (e.g., Figure 14 (e)) and big empty areas appear. Even then, because each child curve has similar starting angles along its parent curve, and the same angle occurs frequently in this region, humans can easily detect the similarities and group them into a texture. We quickly discriminate each group of tiles based on the angles. Visually, the length control provides better spatial control but the stage control might be suitable for some branching phenomena.



(a) 90° **(b)** 45° **(c)** 30°

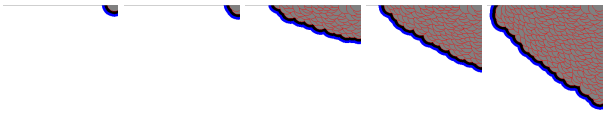
Figure 15: *Splitting II from stage control.*

We can use this process for creating shapes reminiscent of cracks or leaves. As Federl mentioned, cracking in dried mud is commonly seen splitting at 90 degree [Federl 2003]. Figure 14 (a) (b) and Figure 15 (a) resemble this cracking phenomenon, while Figures 14 (f) and 15 (c) are more like leaves. Figure 1 (b) shows a simple combination by putting our cracks on top of a painting generated by the image parsing method [Zeng et al. 2009]. It nicely simulates the craquelure seen in many old oil paintings. Figure 1 (a) shows another example by introducing the colors from the artist's work in Figure 2 (a) to present an abstraction for tree leaves.

5.2 Stacking Technique

5.2.1 Scales

Because we think the splitting technique is sometimes too random and irregular and we want more regular and organized patterns, we introduce a stacking technique to generate scale patterns for snakes or fish based on our basic sequential idea. Given a region and a growth line, we calculate a priority map for all pixels. The priority is calculated to maintain the stacking order. Each step the system pops out the location with the highest priority and grows a curve (here called a scale). If the arc length of the curve is in the proper range, we keep it; otherwise, we remove it and try the next placement. We store an identification map to quickly find the next position. After we have a valid curve, we mark the points inside the scale and on the curve as invalid points (shown in Figure 16 as grey and red respectively). The dark region indicates those points too close to previous scales, where new curves cannot begin. The blue marks the points with the highest priority to be chosen at the next step. The process grows the patterns step by step consistent with the assigned stacking direction. Figure 16 shows the process to generate the scales along the curved path shown in Figure 17 (d). Figure 17 (b) and (c) show the scales grow straightly from the top to the bottom with circular and wavy curves respectively; Figure 17 (e) and (f) maintain the curved path with two different sizes of scales.



(a) *1* **(b)** *2* **(c)** *70* **(d)** *140* **(e)** *250*

Figure 16: *The growth for scales.*

5.2.2 Rivers

There are also numerous elongated textures in nature, such as rivers, hair or feathers, which would be very difficult to simulate by previous region-based methods. We here demonstrate our creation

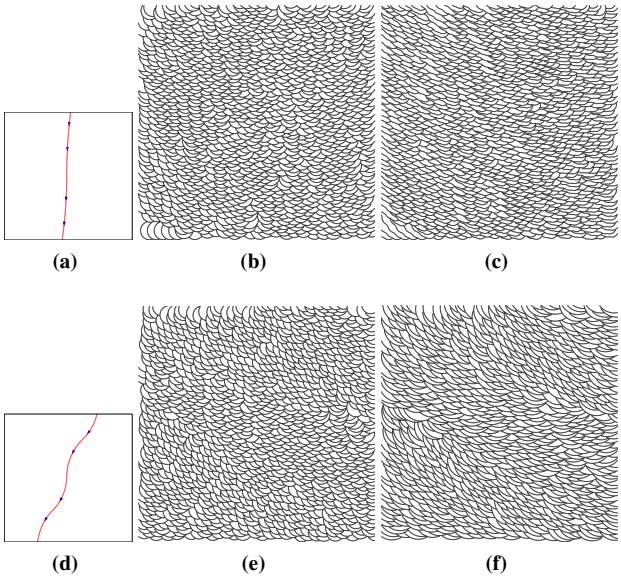


Figure 17: *Four scale examples. (a) The stacking direction for the top row; (b) Circular-like scales; (c) Wavy scales; (d) The stacking direction for the bottom row; (e) Small scales; (f) Large scales.*

for the rivers. Our process starts from a random distribution. At each point, we try to grow a wavy curve along the horizontal direction until it stops when it meets with previous curves or the region boundary. The growth path of stacking is incrementally from bottom to top. If the particles are too close to the previous trails, they die. The short trails are not preserved. The curves appear as the river surface. The top row of Figure 18 shows two river illustrations. Instead of growing a single curve at a point, we can also emit a group of particles at the same position and carefully grow a group of similar curves to illustrate artistic waves. Figure 18 (bottom) illustrates some curly waves by drawing the curly curves first and combining with previous process for the wavy curves later. They provide artistic illustrations for the river.

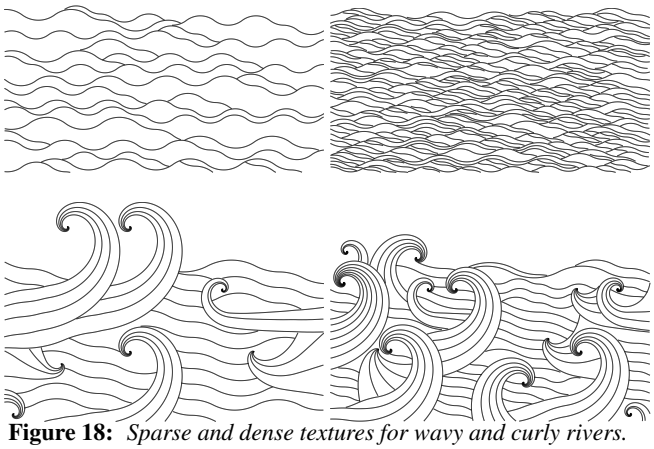


Figure 18: *Sparse and dense textures for wavy and curly rivers.*

5.2.3 3D Indication

In this subsection, we are going to show a powerful aspect of our idea. The key element of our basic idea is to encompass regions with curves. Rather than being purely 2D regions, a region in an image can also be thought of as a facet of an object's surface in 3D. In this way, a group of curves can imply the shape of a surface. Geometrically, if we slice a 3D surface with a plane along a chosen axis repeatedly, there should be a collection of curves generated at the slicing intersections. Actually, it is probably the case that when an artist tries to represent this 3D surface, he can sketch this surface with a group of curves aligning similar to the axis and draw them in the slicing order. We can simulate this process easily using our basic idea. Figure 19 (a) gives a slicing (or called stacking) path and an order. We construct a group of circular curves along this configuration line sequentially, creating the illusion of a 3D cylinder, shown in (b). Figure 19 (c) uses a group of larger circular curves to convey the impression of larger cylinder. An illusion of a small flat cylinder in (d) is obtained by a slightly flatter curve ($f(t) = 0.001 * (t - ((int)(t/500)) * 500)$, identified as curve type III). We can complicate our slice configuration, e.g, as shown in Figure 26; also, we can complicate the curve condition, e.g., as shown in Figure 27. This demonstrates that our method conveys a sense of even more complicated surfaces, and that this concept of 3D suggestion is a way to attain interesting abstracts as well.

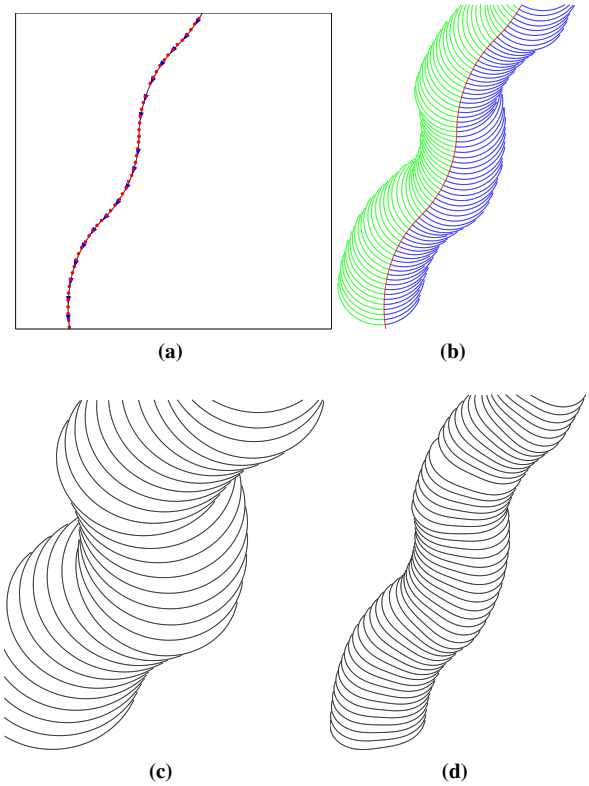


Figure 19: *3D indication I from stacking. (a) A stacking path; (b) Circular curves (curve type I, $s = 4$); (c) Circular curves (curve type I, $s = 1$); (d) Flattened circular curves (curve type III, $s = 4$).*

6 Discussion

We explored our basic tessellation method, applying it to produce mosaics for texture indication and to simulate some abstract and natural patterns automatically. The use of the particle system and the smoothed vector field provides texture indication for mosaics.

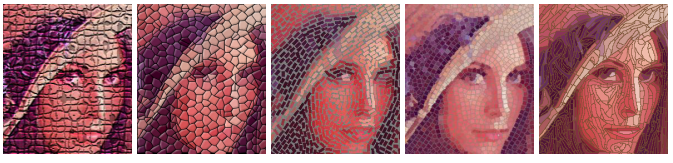


Figure 20: *Comparisons with previous methods. (a) Adobe Photoshop; (b) GIMP; (c) Hausner's method; (d) Artificial mosaics; (e) Our method ($A = 2$, smoothed twice, 993 tiles).*

In Figure 20, we show comparisons with commercial software (Adobe Photoshop and GIMP), and previous region-based meth-

ods such as traditional mosaics [Hausner 2001] and artificial mosaics [Di Blasi and Gallo 2005] in NPR. None of them can preserve highly textured areas, for example, Lena's hair or feather. Our result, despite using a smaller number of bigger tiles, maintains the hair texture nicely. The elongated and irregular tiles in our tessellation are very difficult for other methods to produce. However, our result is less attractive in smooth areas, such as Lena's face, and we still need to improve curve quality in such regions.

Our basic idea is very simple and provides tremendous flexibility to achieve different purposes. In this paper, we put more attention on the sequential method since it can more directly control every growth of the trails. However, it is more possible for the parallel method to produce regularity in the tessellation. P-Method can be used for our mosaic goal too. Figure 21 shows an example by using the same distribution as Figure 12 (b) and (c). The quality is highly dependent on the initial distribution. While the P-Method better respects the direction field, the S-Method is more likely to produce long curved tiles, which are instrumental in fostering a resemblance to Tiffany glass.

Figure 21: *Mould's stippling with P-Method.*

Our demonstrations for abstract and natural patterns should be further explored for textures such as hair, feathered wings, flowers, and other botanical elements. Additional applications can be invented by future users and researchers.

We show some examples of illusory three-dimensional structures in Section 5.2.3. Occlusion is one of the most powerful visual cues for 3D reconstruction, and the curve termination rules imitate occlusion. This synthesis approach is completely opposite to many existing non-photorealistic rendering techniques: instead of looking for silhouettes or creases from 3D models or images, it constructs apparent objects by a procedural organization of curves. We did not much explore the management of intersections in this paper, but we consider it a very intriguing future direction. Figure 22 (a) just randomly draws some groups of wavy curves in a region. Each group looks like a long leaf. If we do a simple intersection organization by controlling the order of each group to draw, we can show a striking depth illusion. If the multiple curves are partly hidden by previously drawn groups of curves, we are not drawing the excluded parts. This simple example shows near leaves obstruct far leaves in (b). This approach gives us a very strong feeling of depth. See also (c), which fills the entire region by this kind of intersection testing.

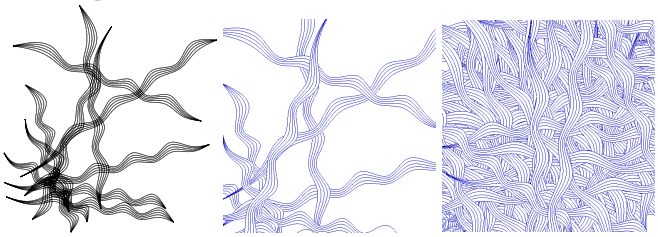
We have somewhat limited the curve variations in the images we showed. When the curves are too variable, the assignment for the path and the particle orientation become trivial and the curve properties will determine the final effects of the tessellations. We show a Jigsaw-like example in Figure 6 (c) and (d) which are using variable curves: $f(t) = sgn(AngleChange(f(t_0)) > \theta(t))) * (f(t_0) + Random(0.0001, 0.01))$ as curve type IV, in which sgn(P) returns 1 if P is true and -1 otherwise. It is difficult to detect any influence from the direction assignment (D1 and D2); instead, the small-scale properties of the curve dominate visually.

Images	resolution	points	curves	tiles	time (s)
Figure 12 (b)	400×468	9,803	3,164	3,579	5.5
Figure 21	400×468	9,803	12,608	8,750	13.9
Figure 23 (d)	400×468	17,997	3,470	3,554	7.2
Figure 1 (d)	500×326	17,832	2,335	2,523	8.6
Figure 24 (d)	660×560	17,300	5,385	6,997	23.7
Figure 28 (b)	622×800	16,791	6,046	7,298	29.5
Figure 28 (a)	800×935	19,571	13,228	15,613	54.8

Table 1: *CPU processing time for mosaics.*

Since we use a discrete map for the intersection checkup, we always let the particle run a few steps (say 5) at the beginning. Otherwise, collisions with the parent curve make it too hard for curves to grow. But if we are placing a lot of particles, it may disobey our definition; see Figure 12 (c), where leaking occurs in the very dark areas. Another limitation is that we should improve the quality at the intersections when two curves meet. These are outstanding implementation issues but do not demand a change in the underlying methods.

Using an Intel Core Duo CPU E8400@ 3.0GHz with 3GB RAM, most of our abstract or cracking results are produced in around $0.1 - 0.5$ seconds. But if we used a distance map for spatial adjustment, it takes longer, around 5-14 seconds. We show our CPU processing time for mosaic generations in Table 1. The performance is similar to previous mosaic methods.



(a) **(b)** **(c)**

Figure 22: *3D illusion from exclusion controls.*

7 Conclusion and Future Work

In this paper, we gave an idea to build a tessellation from colliding curves from a particle system. To further validate our idea, we introduced a texture-indication scheme. We gave numerous demonstrations of natural pattern creation, which do not have either regular tiles or exact shapes. This kind of tessellation should be very useful in applications for nonphotorealistic rendering, texture generation, and artistic abstracts. As for future work, we think the study of a group of particles might be very interesting and the management for the collision intersections might bring very convincing illusions of 3D shapes.

Acknowledgements

We'd like to thank Barbara Keith for inspiration from her amazing works, as well as the anonymous artist from Flickr.com. This work was supported by grants from Carleton University, NSERC, and GRAND.

References

BATTIATO, S., BLASI, G. D., FARINELLA, G. M., AND GALLO, G. 2006. A novel technique for opus vermiculatum mosaic rendering. In *14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'06)*, 133–140.

Figure 23: *Flow-preserving mosaics based on our S-Method by elongated tiles. $A = 4$, with twice smoothing.*

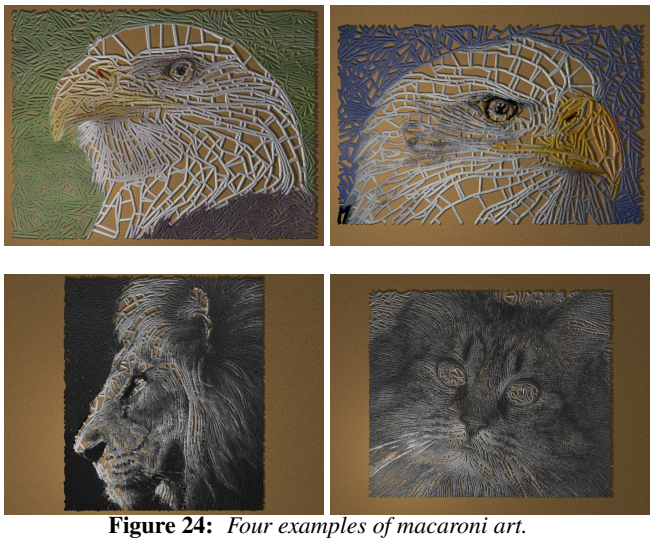
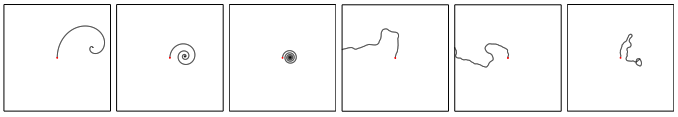
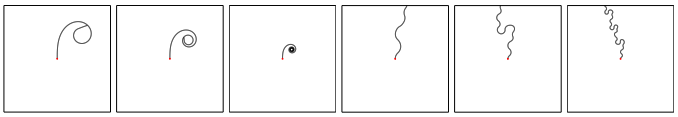


Figure 24: *Four examples of macaroni art.*

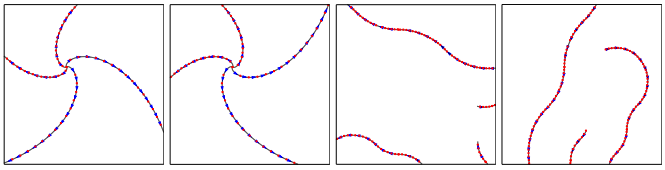


(a) *I:* $s = 0.8$ **(b)** *I:* $s = 2$ **(c)** *I:* $s = 4$ **(d)** *II:* $s = 2$ **(e)** *II:* $s = 3$ **(f)** *II:* $s = 4$



(g) *III:* $s = 0.6$ **(h)** *III* $s = 1$ **(i)** *III:* $s = 4$ **(j)** *IV:* $s = 1$ **(k)** *IV:* $s = 2$ **(l)** *IV:* $s = 4$

Figure 25: *Some examples from curve library. (a) (b) (c): Type I; (d) (e) (f): Type II; (g) (h) (i): Type III; (j) (k) (l): Type IV.*



(a) *C1* **(b)** *C2* **(c)** *C3* **(d)** *C4*

Figure 26: *Four stacking configurations for 3D indication II.*

BROOKS, S. 2006. Image-based stained glass. *IEEE Transactions on Visualization and Computer Graphics 12* (November), 1547–1558.

DI BLASI, G., AND GALLO, G. 2005. Artificial mosaics. *The Visual Computer 21*, 373–383. 10.1007/s00371-005-0292-4.

ELBER, G., AND WOLBERG, G. 2003. Rendering traditional mosaics. *The Visual Computer 19*, 67–78. 10.1007/s00371-002-0175-x.

FAUSTINO, G. M., AND DE FIGUEIREDO, L. H. 2005. Simple adaptive mosaic effects. In *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing*, IEEE Computer Society, Washington, DC, USA, 315–322.

FEDERL, P. 2003. *Modeling fracture formation on growing surfaces*. PhD thesis.

HAUSNER, A. 2001. Simulating decorative mosaics. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '01, 573–580.

HERTZMANN, A. 2003. A survey of stroke based rendering. *IEEE Computer Graphics and Applications 23*, 70–81.

KANG, H., LEE, S., AND CHUI, C. K. 2007. Coherent line drawing. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, ACM, New York, NY, USA, NPAR '07, 43–50.

KIM, J., AND PELLACINI, F. 2002. Jigsaw image mosaics. *ACM Trans. Graph. 21* (July), 657–664.

KYPRIANIDIS, J. E., AND KANG, H. 2011. Image and video abstraction by coherence-enhancing filtering. *Computer Graphics Forum 30*, 2. Proceedings Eurographics 2011.

LIU, Y., VEKSLER, O., AND JUAN, O. 2007. Simulating classic mosaics with graph cuts. In *Proceedings of the 6th international conference on Energy minimization methods in computer vision and pattern recognition*, Springer-Verlag, Berlin, Heidelberg, EMMCVPR'07, 55–70.

LIU, Y., VEKSLER, O., AND JUAN, O. 2010. Generating classic mosaics with graph cuts. *Computer Graphics Forum 29*, 8, 2387–2399.

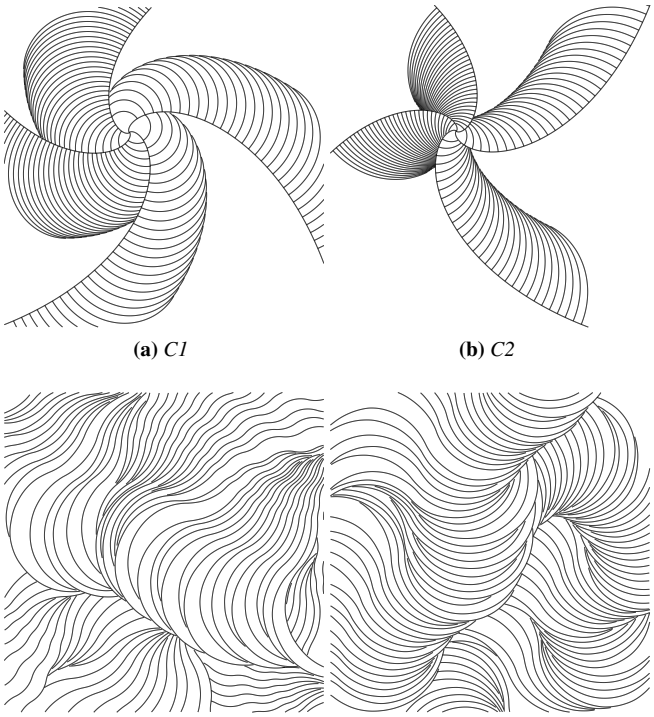
MIYATA, K., ITOH, T., AND SHIMADA, K. 2001. A method for generating pavement textures using the square packing technique. *The Visual Computer 17*, 475–490. 10.1007/s003710100123.

MOULD, D. 2003. A stained glass image filter. In *Proceedings of the 14th Eurographics workshop on Rendering*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, EGRW '03, 20–25.

MOULD, D. 2007. Stipple placement using distance in a weighted graph. In *Proceedings of Computational Aesthetics*, 45–52.

OKABE, A., BOOTS, B., AND SUGIHARA, K. 1992. *Spatial tessellations: concepts and applications of Voronoi diagrams*. John Wiley & Sons, Inc., New York, NY, USA.

ORCHARD, J., AND KAPLAN, C. S. 2008. Cut-out image mosaics. In *Proceedings of the 6th international symposium on*



(a) *C1* **(b)** *C2*

(c) *C3* **(d)** *C4*

Figure 27: *3D indication II. (a) Curve I ($s = 0.2$); (b) The same curve as (a); (c) (d) more complicated curves.*

Non-photorealistic animation and rendering, ACM, New York, NY, USA, NPAR '08, 79–87.

REEVES, W. T. 1983. Particle systems- technique for modeling a class of fuzzy objects. *ACM Trans. Graph. 2* (April), 91–108.

REGAN, D. 2000. *Human Percetpion of Objects: Early Visual Processing of Spatial Form Defined by Luminance, Color, Texture, Motion, and Binocular Disparity*. Sinauer Associates; 1st edition, Sutherland, Massachusetts, USA.

SMITH, K., LIU, Y., AND KLEIN, A. 2005. Animosaics. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, New York, NY, USA, SCA '05, 201–208.

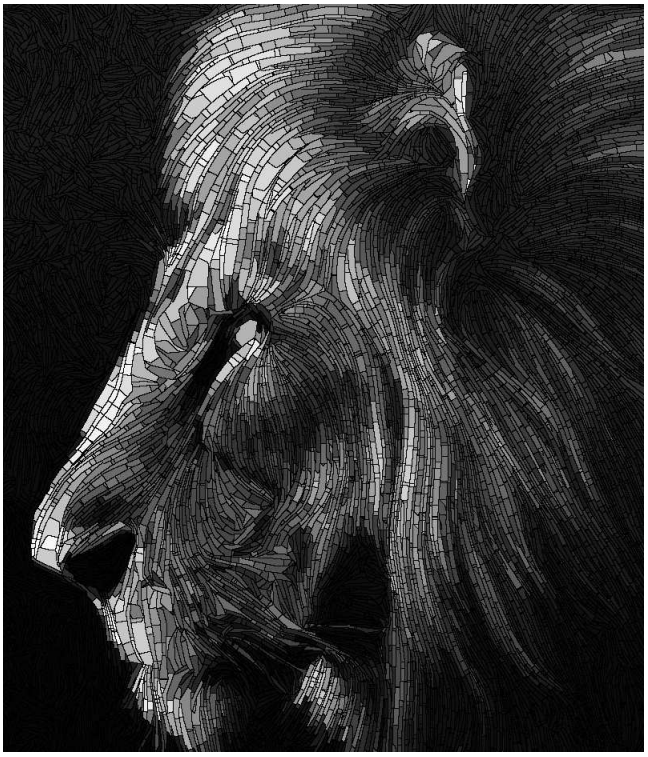
SON, M., KANG, H., LEE, Y., AND LEE, S. 2007. Abstract line drawings from 2d images. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, 333–342.

WONG, M. T., ZONGKER, D. E., AND SALESIN, D. H. 1998. Computer-generated floral ornament. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '98, 423–434.

XIE, N., LAGA, H., SAITO, S., AND NAKAJIMA, M. 2010. Ir2s: interactive real photo to sumi-e. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, ACM, New York, NY, USA, NPAR '10, 63–71.

XU, L., AND MOULD, D. 2009. Magnetic curves: Curvature-controlled aesthetic curves using magnetic fields. In *Proceedings of Computational Aesthetics*, 1–8.

ZENG, K., ZHAO, M., XIONG, C., AND ZHU, S.-C. 2009. From image parsing to painterly rendering. *ACM Trans. Graph. 29* (December), 2:1–2:11.



(a)

(b)

Figure 28: *Stained-glass mosaics. $A = 4$. Smoothing 5 times.*