# Designing 2D Ordinary Differential Equations To Obtain Abstract Paintings, Illustrations and Animations

Ergun Akleman
Departments of Visualization &
Computer Science and Engineering
Texas A&M University

Hüseyin Koçak
Department of Computer Science
University of Miami

## Abstract

In this work, we introduce a simple method for designing ordinary differential equations that can provide desired motions in 2D. Our method provides a simple and intuitive way to construct desired vectors as a mixture of gradient and tangent fields. Both of these fields are defined using 2D implicit functions which can easily be built using existing methods for designing implicit curves. These differential equations can be used to obtain paintings from photographs, abstract illustrations and animations. For abstract animations, we can simply apply a designed differential equation to a set of particles. The motion of the particles directly provides an abstract 2D animation. The trajectories of particles can further be used as abstract illustrations. The particle motion can also be viewed as the motion of the hands of painters and the trajectories of the particles as long unbroken brush strokes over a photograph. These brush strokes can be used to turn photographs into paintings in a controlled way.

## 1 Introduction

In this paper, we present a method for designing ordinary differential equations that can be used for simulating painting, illustration and animation effects. Ordinary differential equations with two variables are particularly useful for such simulations, since they describe a velocity vector at each position on a 2D space. These velocity vectors define a flow which can be used to move particles to obtain simulations for artistic applications. Figure 1 shows some examples of trajectories that are obtained using the differential equations that are designed with our method. As shown in the Figure, we can create orbits in any shape as far as they can be defined by an implicit representation.

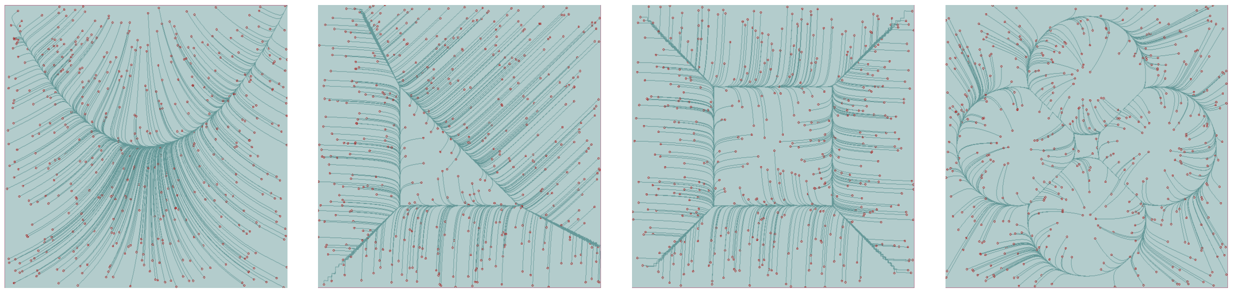
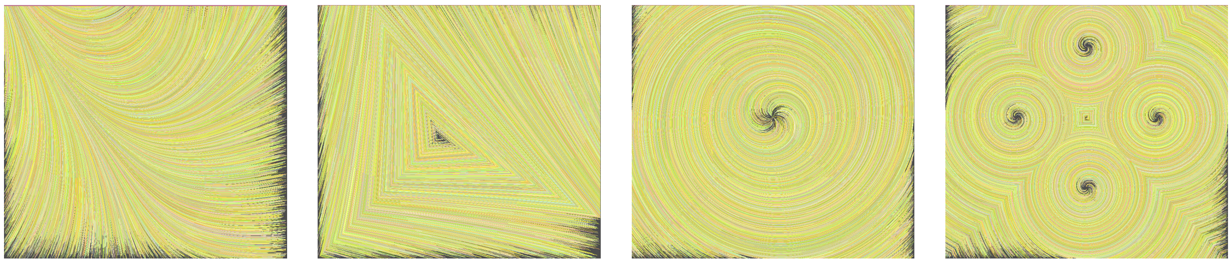

(a) Parabola as an attractor. (b) Triangle as an attractor. (c) Square as an attractor. (d) Union of four circles as an attractor.

**Figure 1** : *Examples of trajectories of differential equations designed using our method.*

Computing solutions of an ordinary differential equation is relatively fast since we only have to compute the solutions for the positions of the existing particles. The number of the particles can be at most equal to the number of the pixels in image. Practically, we can choose the number of particles an order smaller than the number of the pixels.



(a) Parabola as an attractor. (b) Triangle as an attractor. (c) Circle as an attractor. (d) Union of four circles as an attractor.

**Figure 2** : *Paintings obtained by randomly colored trajectories of differential equations designed using our method.*

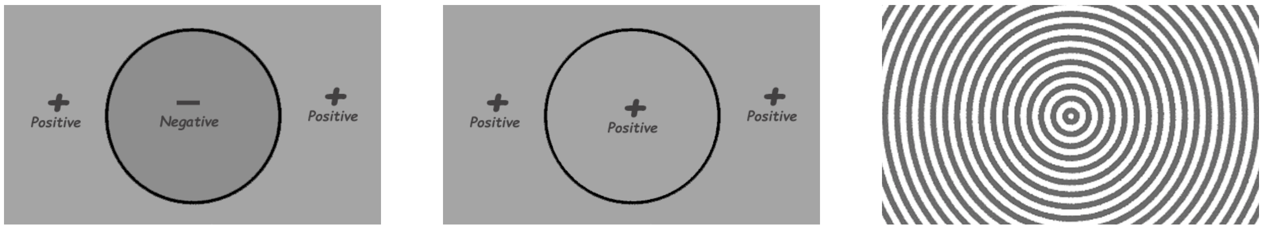
Determining the behavior of solutions of ordinary differential equations can be a daunting task. Moreover, the dynamics of equations can change significantly for small changes in coefficients [Koç89, HK91]. In this paper, we present a method to construct differential equation families that is robust and easy to manipulate. Using these differential equations users can control the flow to be used for painting, illustration, or animation.

These families of differential equations are constructed using two types of vector fields, gradient and tangent. Tangent fields consist of vectors that are perpendicular to gradient field vectors. The differential equations are defined as a weighted average of these two types of fields. The gradient part moves the solutions to points on an implicit curve and the tangent part facilitates rotational movement on the curve. For instance, we can make an implicitly defined curve into an attracting orbit of a differential equation. It is also possible to control how fast particles approach the implicitly represented curves and the speed of the particles on the curve. Moreover, by choosing the initial positions and the colors of particles, one can further control the resulting artworks.

In computer graphics literature, there have been similar approaches for 3D applications. Bloomenthal suggested a method to make an implicitly represented curve or surface stable equilibrium points of a differential equation [BW90] . Since when we apply this differential equation to a set of particles, the particles eventually reach the curve, this approach can be used to sample implicitly represented curves. However, the problem with this approach is that once particles reach the curve, they stop moving and at the end we can have a set of points which may not cover most parts of the curve or surface. In order to get better sampling of an implicitly represented surface Witkin and Heckbert proposed to use local repulsion among *floater* particles that roam freely over the surface [WH94]. They also allow particles to be born and die to spread them evenly over the surface. Their method is not strictly a differential equation based method; however, it is conceptually similar to ours since they let the particles move over the surface.

We want to create a flow under which particles should never stop moving even after reaching the curve and should periodically redraw it. To facilitate the movement of particles on the curve, tangent fields are the key in our approach. Since we work on 2D, the tangent fields are uniquely defined, which is not the case in higher dimensions.

This paper is organized as follows. In Section 2, we present the preliminary information on implicit curves and their usage. In Sections 3, we introduce a methods for designing simple differential equations, with examples in Section 4. We update this method in Section 5 for more complicated differential equations to provide continuous flows anywhere in the plane. Finally, we summarize our conclusions and discuss future work in Section 6.



(a) An implicit shape and its boundary curve. The function is negative in the darker region, called inside, and positive in the lighter region, called outside.

(b) An implicit shape whose boundary is itself. The function is positive everywhere except on the curve, which is zero.

(c) Iso-curves that can be obtained from the same function. Each gray circle is an iso-curve that is defined by $\mathbf{C}(f-d)$ where $d$ is constant.

**Figure 3** : *A simple example of implicitly represented shape: given by* $f(x,y) = \sqrt{x^2+y^2} - 1$ .

## 2 Preliminaries

Implicit representations are important in shape modeling for designing complicated shapes [BWW+97]. They can be constructed using set operations [AC99a, AC99b], convolution operations [BS91], using field functions [Bli82, WW88]. There also exist a variety of methods to design them interactively such as [BW90, WH94, Akl96, Akl98c].

Characteristics of implicit representations make them extremely suitable for designing differential equations to move particles or brushes. Let $f(x,y) : \mathbb{R}^2 \to \mathbb{R}$ be a real-valued function. The set $\mathbf{S}(f)$ defined as

$$\mathbf{S}(f) = \{ \ (x,y) \in \mathbb{R}^2 \ | \ f(x,y) \leq 0 \ \}$$

is called a 2D implicit shape. Figure 3(a) shows an implicit shape, its boundary curve, its inside and outside as the regions that make the function negative and positive respectively. The boundary of a 2D implicit shape $\mathbf{C}(f)$ is called an implicit curve and simply defined as

$$\mathbf{C}(f) = \{ \ (x,y) \in \mathbb{R}^2 \ | \ f(x,y) = 0 \ \}.$$

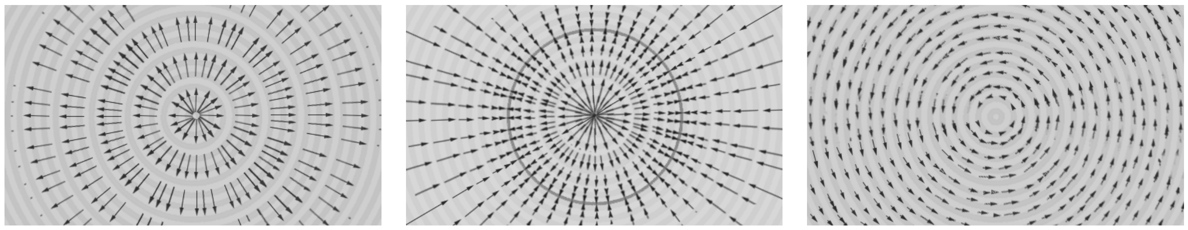
We can also define an iso-curve as $\mathbf{C}(f-d)$ where $d$ is any real number. Figure 3(b) shows iso-curves for the function that creates the implicit shape in Figure 3(a). A particularly useful type of function is $f^2$ since

$$\mathbf{C}(f) = \mathbf{S}(f^2).$$

In order words, to represent boundary curve of $\mathbf{S}(f)$, we can simply use $\mathbf{S}(f^2)$. This observation will particularly be useful for designing vector fields that consist of normals to boundary curves.

**Gradient Fields:** Another useful function is the gradient vector function that is defined as $\nabla f = (f_x, f_y)$. The vector function $\nabla f : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ gives a vector field that consists of vectors that are perpendicular to every iso-curve $\mathbf{C}(f-d)$ for all $d \in \mathbb{R}$ as shown in Figure 4(a). These particular types of fields are called *gradient fields* [MT03, HK91]. Gradient fields cannot directly be used in our formulation because of two limitations: (1) they do not let us reach any particular curve. (2) gradient fields cannot make particles move in a cyclical orbit because gradient fields are irrotational. As it is well-known, a vector field is irrotational if its curl is always zero [MT03]. In the gradient fields this is always true, since $\nabla x \nabla f = f_{xy} - f_{yx} = 0$ [MT03]. In this paper we derive two types of vector fields to solve these two issues of "traditional" gradient fields. We observe that starting from an implicit representation, we can construct these two types of vector fields. Desired differential equations are constructed by mixing these two types of fields.

**Negative Curve Normal Fields:** The gradient vector $-\nabla f^2 = (-ff_x, -ff_y)$ gives a vector field that consists of vectors that perpendicularly point to boundary curve $\mathbf{C}(f)$ as shown in Figure 4(b). We can use



Gradient Field: $\nabla f = (f_x, f_y)$. Negative of Curve Normal Field: $-\nabla f^2 = (-ff_x, -ff_y)$. Tangent Field: $(f_y, -f_x)$.

**Figure 4** : *Three types of vector fields associated with the function* $f(x,y) = x^2 + y^2 - 1$.

this vector field to move in 2D space to reach any particular iso-curve $\mathbf{C}(f - d_0)$ where $d_0$ is any constant that can be given by a user. This idea is used by Bloomenthal [BW90] to sample implicit surfaces, $\mathbf{S}(f)$. A shortcoming of these vector fields is that when particles reach the curve, they stop moving since $f$ becomes zero.

**Tangent Fields:** Any vector perpendicular to $\nabla f$ will be tangent to iso-curves. In 2D, there exists only two perpendicular vector fields: either right-handed, $(-f_y, f_x)$, or left-handed, $(f_y, -f_x)$. Without loss of generality, we will consider only the left-handed tangent fields as shown in Figure 4(c). The curl of a tangent field is $f_{xx} + f_{yy}$, which is very unlikely to be zero. Therefore, tangent fields are most likely rotational. By using these vector fields, it is possible to make a particle move on iso-curves. Witkin and Heckbert [WH94] used this fact in 3D to achieve a uniform sample of a complex surface using local repulsion between *floating* particles and letting particles be born and die. We also used them to create 3D paintings [Akl98b, Akl98a].

## 3 Designing Differential Equations

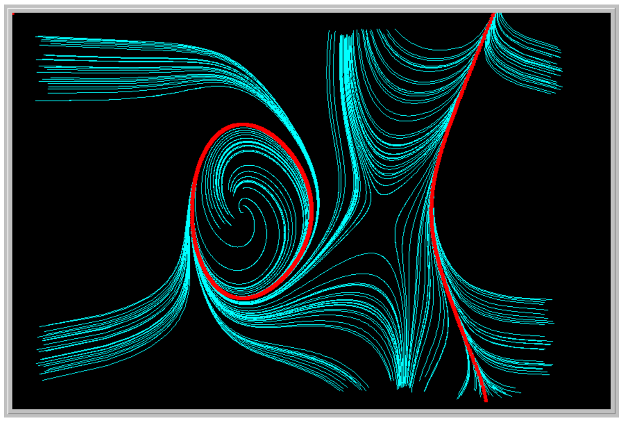
To develop 2D ordinary differential equations that can give control to the painter, we simply take a linear combination of negative curve normal fields and tangent fields. Now, let the curve be given by an implicit equation as $\mathbf{C}(f) = \{(x,y) \mid f(x,y) = 0\}$. As discussed earlier, the same curve $\mathbf{C}$ can also be represented by another implicit equation $\mathbf{C}(-f^2) = \{(x,y) \mid -f^2(x,y) = 0\}$. Note that in the second representation, $-f^2(x,y)$ is strictly negative everywhere except $\mathbf{C}(f)$. Therefore, $\mathbf{C}$ is the global maximum of the function $-f^2(x,y)$. In other words, the negative curve-normal field, which is the gradient of the function $-f^2$, will always give a vector towards $\mathbf{C}$ near $\mathbf{C}$.

**Curves as a set of attracting Equilibrium Points:** If we view the negative curve-normal field as the velocity of a particle at the point $(x,y)$, we can simply obtain a differential equation for which all points on the curve $\mathbf{C}$ are attracting equilibrium points :

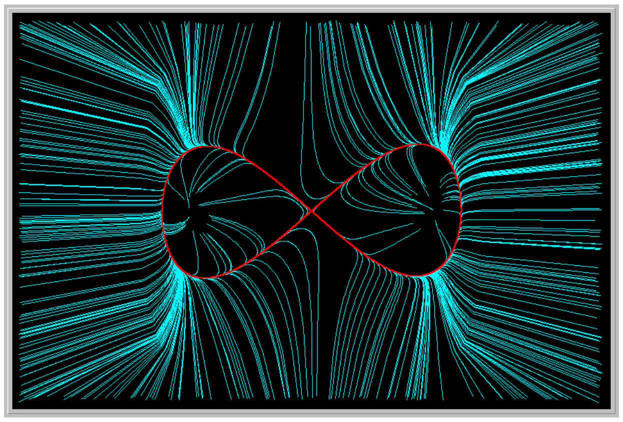
$$\dot{x} = -aff_x\,, \qquad \dot{y} = -aff_y$$

where $a$ is a positive number which represents the velocity at the direction of the gradient vector of $-f^2$. If we apply this differential equation, the particles near the curve $\mathbf{C}$ eventually reach the curve. Once they reach the curve, however, they will stop moving since $f = 0$ on the curve. Therefore, by using this differential equation each particle will determine just one point on the curve. Even if we send a huge number of particles, we may still not see parts of the curve. Since our actual goal is to see a flow along the curve, this form of the differential equation is not adequate. We next discuss how to move the particles once they reach $\mathbf{C}$ using the tangent field.

**Curves as attracting Orbits:** Particles should move in the direction perpendicular to the gradient vector. However, the gradient vector will be zero on $\mathbf{C}$ since $f(x,y) = 0$ by definition. Since $f(x,y)$ represent



(a) Elliptic curve as attractor.



(b) Figure 8 as attractor.

**Figure 5**: *(a) Two-piece attractor of $f(x,y) = y^2 - x^3 + 4x$ and (b) attractor with a singular point of $f(x,y) = x^4 - 4(x^2 - y^2)$.*

only the length of the gradient vector, we can simply disregard it. Then the vector perpendicular to gradient vector is simply the tangent field given as $(f_y, -f_x)$. If we add this new velocity term to our differential equation, we obtain the following equation:

$$\dot{x} = -aff_x + bf_y\,, \qquad \dot{y} = -aff_y - bf_x$$

where $b$ is a real number representing the velocity in the direction perpendicular to the gradient vector.

## 4 Curves Defined by Polynomials

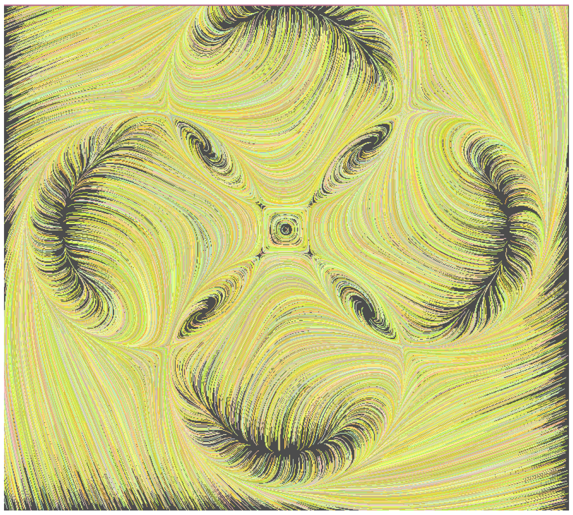
In this section, we demonstrate in several examples how particles can be attracted to implicit curves that are defined by polynomials. An implicit equation of the unit circle is $f(x,y) = x^2 + y^2 - 1 = 0$ and the gradient of $f(x,y)$ is $(2x, 2y)$. With this information, we arrive at the following differential equations:

$$\dot{x} = -ax(x^2 + y^2 - 1) + by,\,, \qquad \dot{y} = -ay(x^2 + y^2 - 1) - bx\,.$$

Note that the origin is an unstable equilibrium point and under the flow all particles are attracted to the unit circle and rotate about, as shown in Figure 2(c). Similarly, from the implicit equation $f(x,y) = y - x^2 = 0$ of a parabola, we derive the following differential equations:

$$\dot{x} = -2ax(y - x^2) + b\,, \qquad \dot{y} = -a(y - x^2) + 2bx.$$

Illustrations for the parabola example are shown in Figures 1(a) and 2(a). Higher degree polynomials can be used to construct more complicated attracting curves. However, such curves can have multiple components or self intersections, thus creating potential stagnation points in the flow. If the additional equilibrium points are isolated and unstable, the rotational part of the vector field can carry the particles without getting stuck at such points. For example, the cubic polynomial $f(x,y) = y^2 - x^3 + 4x$ is an elliptic curve consisting of two components. Notice the additional unstable equilibrium point between the two components as plotted in Figure 5(a). The quartic polynomial $f(x,y) = x^4 - k^2(x^2 - y^2)$ defines a "figure 8" of varying sizes for a parameter $k$. As seen Figure 5(b), the flow of the corresponding differential equations has three unstable equilibria and the particles rotate about the lobes of the figure 8.


(a) Random Colors.

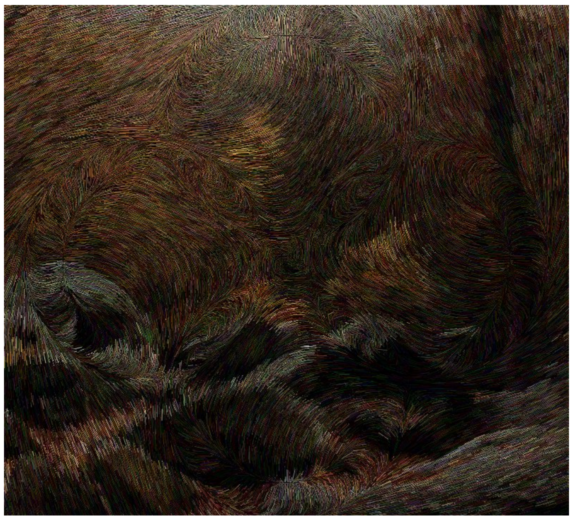

(b) Color from a photograph.

**Figure 6** : *Two paintings obtained by smudging (a) random color and (b) color from a photograph along the trajectory produced by the differential equation that comes from approximate union of four circles as an attractor.*

## 5 Curves Constructed with Set Operations

Complicated curves can readily be constructed by using the functional operators of union and intersection of simpler implicitly represented shapes. The well-known maximum and minimum functional operations provide regularized set operations. If $f_1$ and $f_2$ denote two functions and $\mathbf{S}(f_1)$ and $\mathbf{S}(f_2)$ denote two implicit shapes that are defined by these functions, then one can obtain regularized set operations over these two shapes using maximum as an operator as follows [BWW+97]:

$$\begin{aligned} \mathbf{S}(f_1)\cap\mathbf{S}(f_2) &= \mathbf{S}(max(f_1, f_2)) \\ \mathbf{S}(f_1)\cup\mathbf{S}(f_2) &= \mathbf{S}(-max(-f_1, -f_2)) \\ \mathbf{S}(f_1)-\mathbf{S}(f_2) &= \mathbf{S}(max(f_1, -f_2)) \end{aligned} \tag{1}$$

In other words, using the maximum operator, one can construct complicated shapes with union and intersection of the shapes that are defined by low degree polynomials using constructive geometry [Ric73, WGG99]. The boundaries of these shapes can also be obtained simply by changing inequality with an equality sign. When we use maximum operator, the differential equation can simply be computed using the following procedure:

if $f_i(x, y)$ is the smallest (or biggest) then

$$\dot{x} = -af_if_{ix} + bf_{iy}$$

$$\dot{y} = -af_if_{iy} - bf_{ix}\,.$$

Let $f_i(x, y)$'s be affine functions in the form of $a_ix + b_iy + c_i$; then by using the intersection operator, we can get all convex polygons. For instance, the "triangular shaped" curve in Figures 1(b) and 2(b) is constructed using the intersection of three half-spaces defined by three affine functions, and the "square shaped" curve in Figures 1(c) is constructed using the intersection of four half-spaces defined by four affine functions. The curve in Figures 1(d) and 2(d) is constructed using the union of four circles.

Maximum is not the only functional operator that provides exact intersection of shapes. There exists others such as Rvachev operators [Sha94, PASS95] or our own distance based operators [AC99a, AC99b].

One problem with exact set operators is that the resulting curves can have sharp corners. This is acceptable if we want to obtain polygonal shapes such as triangles or squares. However, if we want to have smooth curves, there is a need to replace maximum and minimum with operators that can provide smooth curves. Fortunately, there also exist such functional operations such as Ricci [Ric73], Wyvill [WMW86, WGG99] or our own approximate set operator [AC99a, AC99b], which is given below:

$$w_a(f_1, f_2) = \frac{1}{a} \log\left(e^{af_1} + e^{af_2}\right).$$

When the maximum operator is replaced by this $w_a$ operator in Equation 1, the resulting shapes smoothly approximate shapes that are produced by exact set operations. Also note that this operator approaches the maximum operator as $a$ gets large: $lim_{a\to\infty} w_a(f_1, f_2) \longrightarrow max(f_1, f_2)$. Another advantage of this operator for differential equations is that it does not change gradient vector length [AC99b]. In other words, particle speeds do not change significantly. Figure 6(a) shows an example of a differential equation that is obtained by replacing maximum with our $w_a$ operator in the union of four circles. As it can be seen in the example, trajectories become more curvy. Figure 6(b) shows the same differential equation applied to a photograph. In this case, a particle is assigned to each pixel and each particle takes the color of the pixel and moves based on underlying differential equation. It is evident in this example that using such a simple set-up we can obtain a painterly effect that resembles line integral convolution (LIC) methods [CL93, SH95].

## 6 Conclusion and Future Work

In this work, we have introduced a method for designing ordinary differential equations using implicitly defined curves as their attracting periodic orbits. We have developed intuitive ways to construct gradient and tangent fields and obtained differential equations as a linear combination of these two vector fields. These differential equations can be used in creation of 2D artworks of paintings from photographs, abstract illustrations and animations. Abstract animations are obtained by applying designed differential equations to a set of particles. The trajectories of particles can further provide abstract illustrations. By viewing particle motion as the motion of the hands of painters and the trajectories of the particles as long unbroken brush strokes over a photograph, these trajectories can be used to turn photographs into paintings in a controlled way.

In our current systems, users need to enter their own equations. A visual interface that allows equation design by direct manipulation of shapes of implicit curves will especially be useful for artists who are not interested in the underlying mathematics. We hope to develop such an interface using set operations over simple shapes.

## References

[AC99a] Ergun Akleman and Jianer Chen. Constant time updateable operations for implicit shape modeling. In *Proceedings of Implicit Surfaces*, pages 73–80, 1999.

[AC99b] Ergun Akleman and Jianer Chen. Coupled modeling of solid textures and implicit shapes. In *Proceedings of Implicit Surfaces*, pages 89–95, 1999.

[Akl96] Ergun Akleman. Interactive construction of smoothly blended star solids. In *Proceedings of Graphics Interface*, pages 159–167. Canadian Information Processing Society, 1996.

[Akl98a] Ergun Akleman. Implicit painting of csg solids. In *Proceedings of CSG*, volume 98, pages 99–113, 1998.

[Akl98b] Ergun Akleman. Implicit surface painting. In *Proceedings of Implicit Surfaces'1998*, volume 2, 1998.

[Akl98c] Ergun Akleman. Interactive computation of ray-quadric surfaces. In *Proceedings of Implicit Surfaces' 1998*, pages 105–114, 1998.

[Bli82] James F Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)*, 1(3):235–256, 1982.

[BS91] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. In *ACM SIGGRAPH Computer Graphics*, volume 25, pages 251–256. ACM, 1991.

[BW90] Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. In *ACM SIGGRAPH Computer Graphics*, volume 24, pages 109–116. ACM, 1990.

[BWW$^+$97] Jules Bloomenthal, Brian Wyvill, Geoff Wyvill, Marie-Paul Cani, Alexander Pasko, John Hart, Chandrajit Bajaj, Jim Blinn, and Alyn Rockwood. *Introduction to implicit surfaces*. Morgan Kaufmann, San Fransisco, 1997.

[CL93] Brian Cabral and Leith Casey Leedom. Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 263–270. ACM, 1993.

[HK91] Jack K Hale and Hüseyin Koçak. *Dynamics and bifurcations*. Springer, 1991.

[Koç89] Hüseyin Koçak. *Differential and difference equations through computer experiments*. Springer, 1989.

[MT03] Jerrold E Marsden and Anthony Tromba. *Vector calculus*. Macmillan, 2003.

[PASS95] Alexander Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.

[Ric73] A Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, 1973.

[SH95] Detlev Stalling and Hans-Christian Hege. Fast and resolution independent line integral convolution. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 249–256. ACM, 1995.

[Sha94] Vadim Shapiro. Real functions for representation of rigid solids. *Computer Aided Geometric Design*, 11(2):153–175, 1994.

[WGG99] Brian Wyvill, Andrew Guy, and Eric Galin. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. In *Computer Graphics Forum*, volume 18, pages 149–158. Wiley Online Library, 1999.

[WH94] Andrew P Witkin and Paul S Heckbert. Using particles to sample and control implicit surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 269–277. ACM, 1994.

[WMW86] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The visual computer*, 2(4):227–234, 1986.

[WW88] Brian Wyvill and Geoff Wyvill. Field functions for implicit surfaces. In *New Trends in Computer Graphics*, pages 328–338. Springer, 1988.