Design of 2D Time-Varying Vector Fields

Guoning Chen, *Member, IEEE,* Vivek Kwatra, Li-Yi Wei, Charles D. Hansen, *Senior Member, IEEE,* and Eugene Zhang, *Member, IEEE,*

Abstract—Design of time-varying vector fields, i.e., vector fields that can change over time, has a wide variety of important applications in computer graphics. Existing vector field design techniques do not address time-varying vector fields. In this paper, we present a framework for the design of time-varying vector fields, both for planar domains as well as manifold surfaces. Our system supports the creation and modification of various time-varying vector fields with desired spatial and temporal characteristics through several design metaphors including streamlines, pathlines, singularity paths, and bifurcations. These design metaphors are integrated into an element-based design to generate the time-varying vector fields via a sequence of basis field summations or spatial constrained optimizations at the sampled times. The key frame design and field deformation are also introduced to support other user design scenarios. Accordingly, a spatial-temporal constrained optimization and the time-varying transformation are employed to generate the desired fields for these two design scenarios, respectively. We apply the time-varying vector fields generated using our design system to a number of important computer graphics applications that require controllable dynamic effects such as evolving surface appearance, dynamic scene design, steerable crowd movement, and painterly animation, many of which are difficult or impossible to achieve via prior simulation-based methods. In these applications, the time-varying vector fields have been applied as either orientation fields or advection fields to control the instantaneous appearance or evolving trajectories of the dynamic effects.

Index Terms—time-varying vector fields, 2D vector fields, vector field design, dynamic effects for surfaces

1 Introduction

Vector field design is a fundamental component for a variety of graphics applications such as remeshing [1], [33], texturing [20], [13], [23], [31], [41], [46], and non-photorealistic rendering [16], [17]. The paramount importance of vector fields in these applications has invoked a line of comprehensive study on the techniques of vector field design on surfaces [6], [8], [34], [51]. Nonetheless, prior research has paid little attention to the more natural and general applications of vector field design to modeling dynamic effects, such as fluid animation [35], [36], crowds [4], [29], shape deformation [45], and video editing [18]. This is in part due to the fact that such dynamic systems are usually time-varying (or time-dependent), with the additional time dimension significantly increasing the complexity of the possible dynamics in the vector fields. In addition, there is no existing theory for the characterization of time-varying vector fields compared to the well-defined feature characterization of static vector fields upon which the design techniques are built. This paper, for the first time, systematically studies the design of time-varying vector fields on two-dimensional manifolds, such as the applications, the taxonomy of the vector fields (orientation, advection), the requirements, and appropriate techniques.

- *Guoning Chen and Charles Hansen are with Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112. E-mail: {chengu,hansen}@sci.utah.edu*
- *Vivek Kwatra is with Google Inc., Mountain View, CA 94043. E-mail: kwatra@gmail.com*
- *Li-Yi Wei is with Microsoft Research, Redmond, WA 98052-6399, and The University of Hong Kong, Pokfulam, Hong Kong.*
- *Eugene Zhang is with Oregon State University, Corvallis, OR 97331. E-mail: zhange@eecs.oregonstate.edu*

1.1 Requirements

For most of graphics applications involving dynamic effects, there are a number of requirements on the underlying time-varying vector fields and how they are modeled.

First, the designed time-varying vector fields should preserve temporal coherence to guarantee the smooth transition of dynamic effects that they are driving. This is a fundamental requirement for achieving a visually pleasing animation.

Second, the obtained time-varying vector fields can be physically plausible or implausible, incompressible or compressible, in order to satisfy the different requirements of specific applications. For instance, practitioners in fluid dynamics often require incompressible flows, while animators may seek for more flexible vector fields for the dynamic effects with volume change such as crowd simulation. Any vector field system needs to be able to handle general time-varying vector fields with such diverse properties.

Third, the time-varying vector fields are designed to either control the evolution of the instantaneous appearance of certain graphical primitives (e.g. the sizes and orientations of the texture and brush strokes) or advect certain objects (e.g. flow parcels) over time, in order to control different aspects of the dynamic effects. A vector field design system should facilitate both types of vector fields.

Fourth, the design system for the time-varying vector fields should provide the user an intuitive and flexible interface to support the modeling of various flow behaviors. In addition, a number of different modeling approaches should be supported. Specifically, there are a few possible situations during the modeling of a time-varying vector field that a user may encounter. 1) The user wishes to design the detailed local behavior of the flow over time. 2) The user cares about the

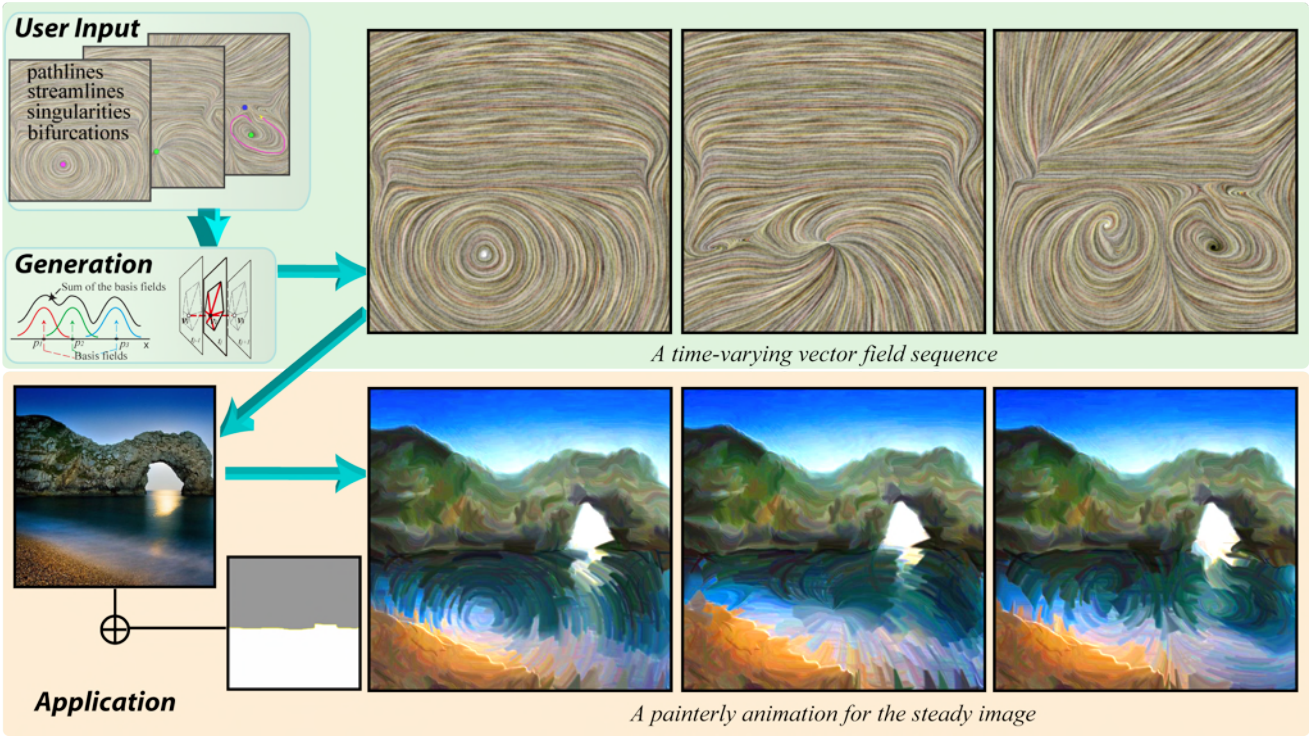


Fig. 1: This figure shows the pipeline of the presented design system for 2D time-varying vector fields. First, the user specifies the desired flow behaviors in the forms of spatial-temporal constraints. The system then produces a time-varying vector field that matches the constraints. The obtained field is then applied to the computer graphics applications to create various dynamic effects. Here, we apply the obtained fields to produce painterly animation from a single image. Note that we use the created time-varying vector field to orient and move the brush strokes in the lower part of the image to achieve an artistic water wave effect: the vortex rotates, moves and changes its characteristics, then splits into two vortices at the end. Please see the accompany video for this animation. The inset plot shows the changes of the consecutive instantaneous fields in terms of the total variance of the vector values in the space.

exact states of the flow at only certain times and would like the system to generate the rest of the field. 3) The user is given a static vector field, and tries to deform it to make up a sequence of time-varying vector field as people do for mesh deformation. A properly devised design system should be able to accommodate these scenarios.

1.2 Our Method

In order to develop a design system for time-varying vector fields to satisfy the aforementioned requirements, we propose a design framework which is based on the discretization of the time-varying vector fields in the time dimension and consider them as the sequences of static vector fields with slow changes over time. This philosophy is based on an observation that solutions to the time-varying vector fields converge to families of solutions of the instantaneous vector fields as the rate of temporal change in the vector field goes to zero, which preserves temporal coherence and helps achieve smooth transition of the dynamic effects. This observation is also a fundamental assumption when developing bifurcation theory for time-varying vector fields [11]. With this temporal discretization, we are able to adapt the previously developed tools for static vector field design to time-varying vector fields with the desired instantaneous dynamics.

To enable the creation of various flows, we provides the user with the ability of modeling the following flow properties: 1) a snapshot of the flow at a given time, 2) the path of a particle in the domain, 3) the path of a singular feature, and 4) the interaction of the features of interest. These features in turn reflect important flow characteristics, such as the solution of the dynamical system at a given time, the trajectories of the flow parcels, and how the flow parcels interact over time. These flow characteristics can be described by *streamlines*, *pathlines*, *singularity paths*, and *bifurcations*, respectively. They sufficiently describe the local flow behavior in space and time, and are capable of creating time-varying vector fields that are used to align or advect the graphical primitives as required. The former type of vector field is referred to as an *orientation field* and the latter an *advection field*. We provide the design metaphors for the user to model these flow characteristics. Specifically, we present the first technique that allows the user to prescribe bifurcations, a unique type of phenomena not present in static fields.

To support the required design scenarios, we introduce three distinct field design approaches. Specifically, the modeling of the local flow behaviors is supported by the *time-varying design elements* extracted from the user specified flow characteristics. A basis field summation or a constrained optimization is performed to generate the instantaneous vector field at a given time based on the instantaneous characteristics of the elements. *Key frame design* is employed to support the case when a user only provides the instantaneous fields at

the desired times. A spatial-temporal Laplacian relaxation is proposed to generate the rest of the sequence. *Time-varying transformation* is used when an initial static field is *deformed* over time to generate a time-varying vector field.

The combination of the proposed design metaphors and generation techniques has led to a design system which takes the user input and generates a time-varying vector field using one of the generation approaches according to the selected design approach. The system also enables the user to further modify the obtained vector field through local topological editing. The generated time-varying vector fields can be applied to a number of important computer graphics applications to achieve various dynamic effects including producing artistic fluid effects over static images, steering 2D crowds, controlling field animations, and time-varying effects of surface appearance.

2 Related Work

Vector field design refers to the creation of a continuous vector field on a manifold that respects user specified or application-dependent constraints. Most existing work focuses on a static vector field. Depending on the goals, there are two different classes of vector field design techniques: One is non-topology based; the other is topology based.

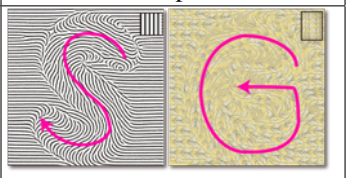
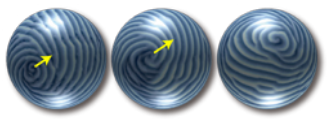
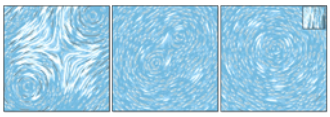
Non-topology based methods: Non-topology based methods do not address *vector field topology* [15] explicitly. The vector field design tools in the early graphics applications, such as texture synthesis [41], [46], fluid simulation [35], [36], and visualization [43], are the examples of this category. Other applications, such as non-photorealistic rendering [16], [17], remeshing [1], and parameterization [33], also employ vector field design respectively. Most of these applications require only the direction information of the input vector fields, and hence a simple design functionality for the vector fields. However, the user has little control of unwanted singularities in the field that often leads to visual artifacts.

Topology-based methods: Topology-based approaches allow the user to control the number and positions of singularities [44], [51], [8] or the topological graph explicitly [37]. More general N-way rotational symmetry field design has also been studied by Palacios and Zhang [27], Ray et al. [34], and Lai et al. [21]. Recently, Crane et al. [6] present a technique which allows arbitrary prescription of singularities and constraints on the fields.

Time-varying methods: Most of the above work concerns only time-independent (i.e. static) vector fields. On the other hand, many applications produce time-varying vector fields during execution without providing the user an effective intervening interface, such as fluid simulation [35], crowd animation [39], [29], shape deformation [45], hair modeling [10], and video editing [50]. This has restricted the achievable effects. Wejchert and Haumann [47] introduce the idea of flow design to create controllable aerodynamics animation. The modeled field is steady and needs to be combined with physically-based simulation to generate aerodynamics animation. To achieve time-dependent control, the user exerts external force to the system as the work by Stam [35], [36]. However, simulation is expensive and hard to control. In addition, simulation is incapable of generating physically impossible artistic fluid flow effect. Pighin et al. [30] has accomplished a closely related work. They introduce an advected radial basis function to model and edit flows as well as an interactive pathline editing interface. Compared to their work, our techniques enable the user to create more general 2D vector fields than those incompressible flows. Recently, Kagaya et al. [18] present a simple design interface to control the time-varying tensor fields for the temporarily coherent painterly rendering of videos. Xu et al. [49] describe a technique for fast dynamic design of vector fields. These fields need not preserve temporal coherence. Ma et al. [24] propose a motion field synthesis technique which enables the user to generate artistic flow effects. However, the method only generates detailed motion vectors and relies on a predetermined low resolution dynamic vector field for synthesis. To that end, we are not aware of any work on the design of time-varying vector fields for the general purpose of graphics applications.

3 Overview

In this section, we provide a brief description of how our framework assists the design of a time-varying vector field. First, the user provides a number of specifications on the desirable characteristics of the field. These include:

flow descriptors	examples
Streamline, for the control of the flow geometry at a certain time frame and most useful for the design of orientation fields	
Pathline, for the description of movements of specific particles across space and time (appropriate for advection fields)	
Singularity path, for the representation of the trajectory of the singular features over space and time (useful for orientation fields)	
Bifurcation, for the description of the collisions or splits of different singular features over space and time (useful for orientation fields)	

These characteristics depict diverse flow behaviors which can be observed in many dynamic applications. For instance, in texture synthesis and painterly rendering, the user often wishes the texture patches and brush strokes to be oriented in a certain way. An orientation field can be created to achieve that with the desired instantaneous flow patterns prescribed by the user specified streamlines. In crowd simulation, the user would like to steer a group of pedestrians to follow certain route. This is similar to specifying a path for these pedestrians. An advection field generated from the specified

pathline can be applied to accomplish that (see Figure 16). In meteorological animation design, the user may create the effect of two storm systems moving toward each other and eventually colliding (see Figure 9). This can be done by controlling the movement (singularity paths) and interaction (bifurcation) of the two vortices in a time-varying vector field.

Note that for most graphics applications shown in this paper, instantaneous appearance is often a bigger goal than the exact path of a particle. For the rest of the paper, we will assume the designed fields serve as orientation fields except for the application of crowd simulation where the pathline design is used to generate an advection field. Nonetheless, for most examples the orientation fields are also used to advect the graphical primitives over time to achieve the *moving* effect.

The overall pipeline of our system is as follows (shown in Figure 1). First, according to the selected design scenario, the user specifies a number of constraints. For key frame design and field deformation, the focus is on the creation of some instantaneous (static) fields. As such, specifying streamlines and the singularities is sufficient. A streamline can be specified using the drawing tool of our system. Our system will compute the tangent vectors at the sample positions as the constraints. For element-based design, pathlines, singularity paths, and bifurcations can be used. In particular, for a pathline, besides computing the tangent vectors at the sampled positions, the temporal value for each sample point is required from the user (Section 5.1). The user is also responsible to provide the type of a singularity path (source, sink, or saddle), represented as a time-varying Jacobian. To specify a bifurcation, the user can describe a template function (Section 5.1) that will lead to the desired bifurcation. Note that in our paper we only handle saddle-node bifurcation where a node is either a source or sink. Figure 2 provides some examples on how the users can specify these flow descriptors with our system.

Once the constraints have been specified, our system generates a time-varying vector field by using the basis field summation (Section 5.2), constrained optimization (Sections 5.3 and 6.1), or time-varying transformation (Section 7) according to the selected design method. The resulting field is analyzed with singularities and bifurcations extracted. The user then has the ability to specify additional constraints or perform local topological editing in the form of singularity and bifurcation movement or cancellation. This process continues until the user is satisfied (Section 8).

In the next section, we will provide the mathematical definitions for the aforementioned flow characteristics.

4 Time-Varying Vector Fields

In this section, we briefly review the mathematical definitions of the important concepts of time-varying vector fields, which will facilitate our later design tasks.

Streamlines and Pathlines: We consider a 2-manifold $\mathbb{M}$. A time-varying vector field V is a map $V : \mathbb{M} \times \mathbb{R} \to \mathbb{M}$, which can be expressed as a differential equation $\frac{d\mathbf{x}}{dt} = V(\mathbf{x};t)$. The solution of it given an initial state $\mathbf{p}_0 = (\mathbf{x_0};t_0)$ is $\mathbf{x}(b) = \mathbf{p}_0 + \int_0^b V(\mathbf{x}(\eta);t+\eta)d\eta$, which is referred to as a *pathline*. It is the trajectory of the particle under V. The vector field $V(\mathbf{x};t_c)$ is an *instantaneous vector field* of V at time t_c, which is static. The solution from $\mathbf{p}_c = (\mathbf{x_c};t_c)$ constrained in $V(\mathbf{x};t_c)$ is a *streamline*, and $\mathbf{x}(b) = \mathbf{p}_c + \int_0^b V(\mathbf{x}(\eta);t_c)d\eta$.

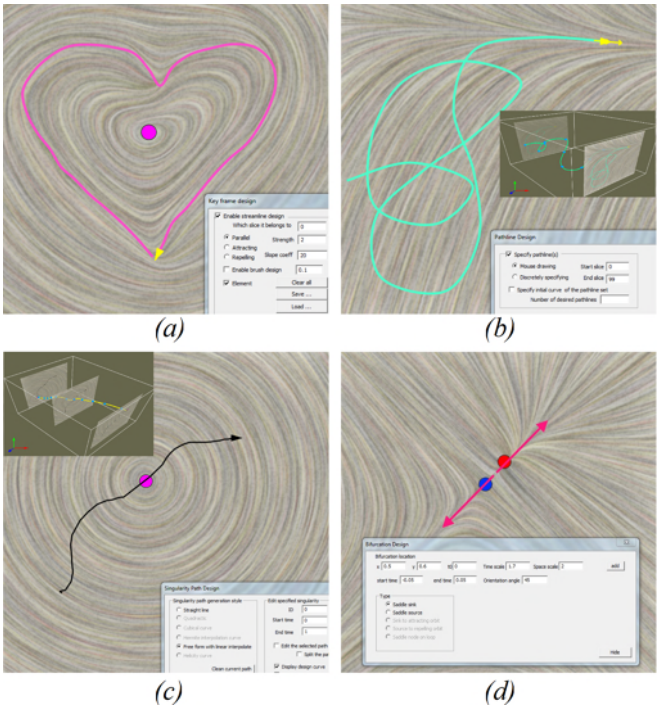


Fig. 2: Our user interface showing different design metaphors: (a) streamline, (b) pathline, (c) singularity path, and (d) bifurcation. A streamline is specified at a particular time as a 2D curve. A pathline can be provided either in the 2D domain with the starting and ending time information or directly in the spatial-temporal domain (see the inset of b). Similarly, a singularity path can be designed in either 2D domain with the birth and death times or in the spatial-temporal domain (see the inset of d). A bifurcation is prescribed as a point in the spatial-temporal domain with the coordinate, scaling, and orientation information.

Instantaneous Topology: The topology of $V(\mathbf{x};t_c)$ is referred to as the *instantaneous topology* of V at t_c. It consists of *singularities*, *periodic orbits*, and their connectivity [3] and conveys the qualitative information of $V(\mathbf{x};t_c)$. This information has been applied to guide the creation and control of static vector fields [3], [8], [44], [51]. It has been shown that analyzing and tracking instantaneous features can provide more information for graphics applications than the space-time topology defined based on pathlines which is typically featureless [38]. Therefore in the rest of the paper, we will make use of the notion of instantaneous topology to discuss the structural evolution of a time-varying vector field. Also, we focus on singularities only as they are more relevant to the present graphics applications.

Singularities and Singularity Paths: A point $\mathbf{p}$ is called a singularity of $V(\mathbf{x};t_c)$ if $V(\mathbf{p};t_c) = 0$. We are interested in the isolated singularities in the field, each of which can be enclosed by a compact neighborhood containing no other singularities. The type of each singularity is determined by the flow characteristics within this neighborhood. The linearization of $V(\mathbf{x};t_c)$ about $\mathbf{p}$ results in a 2×2 matrix

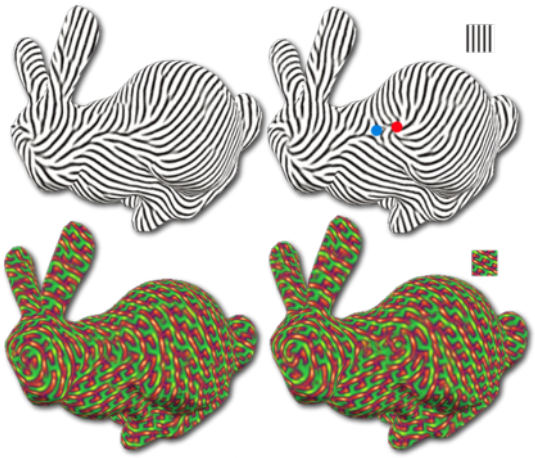


Fig. 3: This figure shows an example of saddle-node bifurcation in an orientation vector field. The creation of a pair of saddle and sink causes the break of texture structure on the back of the bunny. Note that we sample the two frames before (left column) and after (right) the bifurcation point to reveal the discontinuity.

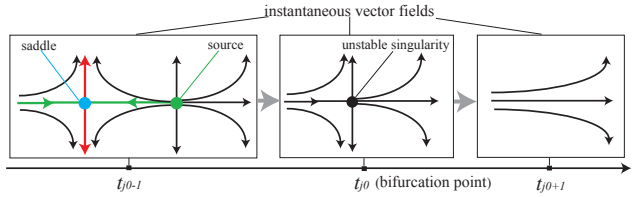


Fig. 4: This example demonstrates a saddle-source cancellation bifurcation. The directional curves illustrate the flow behavior. Two singularities are shown in the left at t_{j_0-1}. They move towards each other when t increases and collide at t_{j_0} (middle). The two singularities are canceled after they meet, which results in a singularity-free vector field at t_{j_0+1} (right).

$DV(\mathbf{p}) = \begin{pmatrix} \partial v_x/\partial x & \partial v_x/\partial y \\ \partial v_y/\partial x & \partial v_y/\partial y \end{pmatrix}$ (called *Jacobian*) which has two (potentially complex) eigenvalues $\sigma_1 + i\mu_1$ and $\sigma_2 + i\mu_2$. If $\sigma_1 \neq 0 \neq \sigma_2$,then $\mathbf{p}$ is called a *hyperbolic singularity*. They are the stable structures *w.r.t* perturbation compared to those unstable ones, such as centers. Observe that on a surface there are three types of hyperbolic singularities: *sinks* $\sigma_1, \sigma_2 < 0$, *saddles* $\sigma_1 < 0 < \sigma_2$, and *sources* $0 < \sigma_1, \sigma_2$. Each singularity has a life span $[t_s, t_e]$ $(t_s, t_e \in \mathbb{R})$ where t_s represents the time of its birth and t_e is the time of its annihilation. The curve connecting each position of the singularity during its life span is called a *singularity path*. We assume the type of a singularity does not change during its life span.

Bifurcations: The birth and annihilation of singularities imply the change of the topological structural of the vector field. We refer to this qualitative change as the *bifurcation* and the places where these changes occur as the *bifurcation points*. Bifurcation is an important event in time-varying vector fields. In many graphics applications involving time-varying vector fields, bifurcations can lead to the structural changes of certain geometry or properties, such as the splitting and merging of vortices in fluid animation. In some cases, these structural changes may cause visual artifacts. Figure 3 shows an example where the break of texture structures induced by the bifurcations of the underlying vector field causes visual discontinuity in the animation. Therefore, studying bifurcations and developing effective techniques to control them is necessary from the application perspective. The rigorous definition of bifurcation is beyond the scope of this paper. Interested readers can find a more comprehensive introduction of the bifurcation theory in [11]. Consistent with our focus on singularities, in this paper we discuss only the local bifurcations, such as *saddle-node (fold) bifurcation* and its inverse bifurcation which refers to the annihilation of sink/source and saddle pairs. Figure 4 illustrates a saddle-source bifurcation where a source with Poincaré index 1 and a saddle with index -1 move towards and finally cancel each other over time. This bifurcation can be formulated as follows [11].

$$V((x,y);t) = \begin{pmatrix} t+x^2 \\ y \end{pmatrix} \quad (1)$$

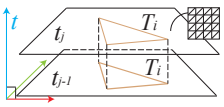
while a saddle and sink creation can be formulated as follows,

$$V((x,y);t) = \begin{pmatrix} t-x^2 \\ -y \end{pmatrix} \quad (2)$$

The change of the type of a singularity also corresponds to a *transcritical bifurcation*, for instance, sink→center→source, and vice versa. When this occurs, we consider a new singularity is born while the old one is eliminated.

With these concepts, we next describe how we support the three different design scenarios as introduced in Section 1. We first describe the setting of our computation domain.

Computation Domain: We consider a sub-domain $\mathbf{D_X} = (X;t)$ where $\mathbf{D_X} \subset \mathbb{M} \times \mathbb{R}$ is a spatial-temporal domain. X is a triangulation of a 2D curved surface embedded in 3D, and $t \in [0,1]$ is the time parameter. For representing and storing the field, we discretize t evenly. We denote these discretely sampled values as $\{t_j\}$. We then compute and store the instantaneous fields at these discrete times $\{t_j\}$ in order. In each instantaneous field, vector values are sampled at the vertices of the triangulation X. The inset figure shows such a configuration. For a planar domain, we use a free-form boundary. Along t, we assume the time-varying vector field in $\mathbf{D_x}$ is a portion of the time-varying field with $t \in (-\infty, \infty)$. Other boundary conditions of t can be employed, such as a periodic boundary conditions often used in fluid simulation [35].



Note that in order to enable more flexible speed control of the final animation sequences, the parameter t has a linear relation with the physical time, that is, $c \cdot \mathrm{d}t$ $(c \in \mathbb{R}^+)$ will be the actual time interval when applying the created field to the target applications.

Interpolation scheme: We assume that the vector field is defined on the vertices of the mesh domain. In space X, the vector values within a triangle is computed using the parallel transport technique of Zhang et al. [51]. Along the parameter t dimension, we employ a similar interpolation technique proposed by Tricoche et al. [40] to guarantee the linearity over t. In particular, the vector value of a sample point p in-between two frames can be computed using linear interpolation of the two values at p at the two frames.

5 Element Based Design

In this section, we describe how we support the design of local spatial and temporal behaviors of the flow through a number of design elements that can be extracted from the user specified flow characteristics. These design elements are later combined to generate a time-varying vector field.

5.1 Design Elements

Our system supports the following design elements.

Singular Elements:

Modeling singular elements is an essential functionality for vector field design. We extend the singular elements in the static field design [51] to our spatial-temporal setting. Specifically, we denote a singular element as $S(J,P(t),M(t))$, where J is the Jacobian that defines the selected type of the singular element, $P(t)$ represents the path of the singular element over time, and $M(t)$ is the affine transformation matrix (i.e. scaling and rotating) that is exerted on the element along $P(t)$. We assume J is fixed along $P(t)$. $P(t)$ is derived from a user specified path (Figure 5). In particular, after the user sketches the path of a singular element, a Hermite spline curve is fitted to it to form a smooth path $P(t)$. $M(t)$ is initialized as an identity matrix and can be modified along $P(t)$. Given a time t_c, $M(t_c) = \begin{pmatrix} s_x(t_c) & 0 \\ 0 & s_y(t_c) \end{pmatrix} R(\theta(t_c))$ where $s_x(t_c)$ is a x scaling, $s_y(t_c)$ a y scaling, and $R(\theta(t_c))$ a rotation centered at $P(t_c)$. The user specifies a number of $M(t_i)$ at the desired times t_i. $M(t)$ can be computed through linearly interpolating s_x, s_y and θ at t_i and t_{i+1} where $t_i < t < t_{i+1}$.

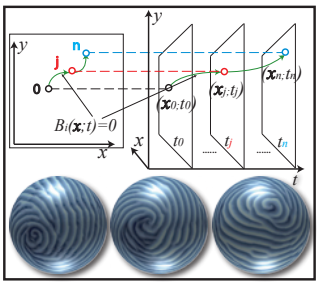


Fig. 5: Singularity path.

$P(t)$ starts and ends at $t = 0$ and 1 by default. If it starts or ends in between, a certain bifurcation is induced, which in turn involves another singularity with opposite Poincaré index. In design, this can be achieved by intersecting the two singularity paths (by definition in Section 4). At the bifurcation point where the two paths intersect, the local Jacobian is degenerate, while the Jacobian of the rest of the field is not. In addition, after (cancellation) bifurcation, the field is singularity free (see Figure 3 (right)). Therefore, local control of the Jacobians through the singularity paths to their bifurcation point is insufficient to guarantee a smooth and non-degenerate field. In the following, we introduce the bifurcation elements that can reduce the burden of the local and global control of the Jacobians from the user.

Bifurcation Elements: Recall that we are concerned with saddle-node bifurcations in this paper. Equations 1 and 2 are two normal forms that define a saddle-node bifurcation at position $(0,0;0)$ in domain X (i.e. a bifurcation element). Specifically, equation 1 induces a saddle-source cancellation and equation 2 defines a saddle-sink creation. During design, the user prescribes the position, $(x_0,y_0;t_0)$, of a bifurcation with the desired type in domain X. Thus, we replace $x = x - x_0$, $y = y - y_0$, and $t = t - t_0$ in equations 1 or 2 to place the bifurcation elements in the right position. Further, a user manipulated transformation can be exerted to scale the range of the bifurcation in both space and time, and re-orient an axis (a straight line in this case) to control where and how the bifurcation occurs along the axis. Figure 6 provides an example where the user inserts a number of bifurcations.

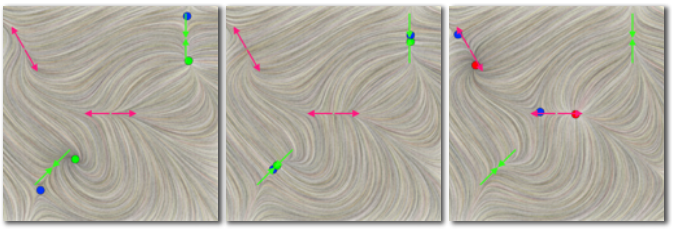


Fig. 6: A number of bifurcations are inserted using the Equations 2 (red), 1 (green).

These bifurcation elements enable the user to insert bifurcations through *templates* (i.e. the bifurcation normal forms) with guaranteed smooth transition in space and time. However, it does not allow the modification of the paths of the involving singularities. More intuitive and flexible design metaphor for bifurcations is much desired and should be included in the future work.

Regular Elements:

In static field design, a regular element is useful in providing the translation or advection direction for a particle located at a point and related to streamlines. In the design of time-varying vector fields, this element is tightly linked to pathlines.

We define a regular element as $R(V(t),P(t))$ where $P(t)$ is a prescribed pathline and $V(t)$ is the tangent direction at $P(t)$ in space at a time t.

Consider a user specified pathline curve which consists of the positions of a particle $\mathbf{p}$ from t_s to t_e ($t_e \geq t_s$), denoted by $\bigcup_{t_s}^{t}(\mathbf{p}(s))$. Assume m sample points, $\mathbf{p}_i$, along the curve are taken. A Catmull-Rom spline $P(t)$, is computed with $\{\mathbf{p}_i\}$ as the control points. The spline curve is densely sampled as the set of evenly spaced points $\{\mathbf{sp}_j\}$. Assume K is the number of sample points on the spline curve and N is the number of time samples. We set $K > 4N$ for a smooth representation such that $V(t_i) = (\mathbf{sp}_j - \mathbf{sp}_{j-1})$, a good approximation of the tangent direction, is placed at $P(t_i)$ where $t_i \in [0,1]$ is the i^{th} sampled time (see Figure 7). To reduce user input, a uniform sampling, $t_i = t_s + i \times (t_e - t_s)/(N-1)$ can be used. However, this is not required. $\mathbf{sp}_j$ and $\mathbf{sp}_{j-1}$ are the points that enclose $P(t_i)$ on $P(t)$.

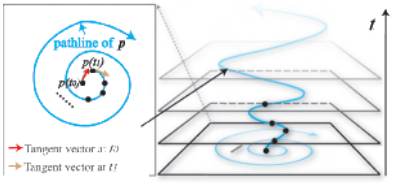


Fig. 7: Pathline example.

Because of the cumulative numerical integration error, the computed pathline starting from $P(0)$ following the generated flow is typically deviated from the prescribed $P(t)$. To resolve

that, instead of using conventional regular elements, we make use of the attachment elements introduced in [3], which guarantee the obtained pathline converging to the desired one. The following formula describes an attachment element that has a vector value of $(1,0)$ at (x_0,y_0).

$$V(x,y) = \begin{pmatrix} 1 \\ c(y-y_0) \end{pmatrix} \tag{3}$$

5.2 Basis Fields Summation

In order to generate a time-varying vector field from the user specified elements described above, a basis field summation can be used which has been applied to static field design [47], [44], [51]. We extend this basis field summation to take into account the design elements with time-varying characteristics introduced in the previous section. Specifically, the basis field generated by a singular element at time t has the form:

$$V_i(\mathbf{x};t) = e^{-d\|\mathbf{x}-\mathbf{p}_i(t)\|^2} M_i^T(t) J_i M_i(t) \begin{pmatrix} x - x_{\mathbf{p}_i}(t) \\ y - y_{\mathbf{p}_i}(t) \end{pmatrix} \tag{4}$$

where $\mathbf{p}_i(t) = (x_{\mathbf{p}_i}(t), y_{\mathbf{p}_i}(t))$ is the position of the singular element at time t along the path $P_i(t)$, and $M_i(t)$ is the transformation acting on J_i. The basis field for a regular element given time t has the form

$$V_i(\mathbf{x};t) = e^{-d\|\mathbf{x}-\mathbf{p}_i(t)\|^2} V_i(t) \tag{5}$$

The basis field for an attachment element at t is

$$V_i(\mathbf{x};t) = e^{-d\|\mathbf{x}-\mathbf{p}_i(t)\|^2} M_i^T(t) \begin{pmatrix} 1 \\ c(y - y_{\mathbf{p}_i}(t)) \end{pmatrix} \tag{6}$$

A bifurcation elements generate the following basis field.

$$V_i(\mathbf{x};\ t) = e^{-d\|\mathbf{x}-\mathbf{p}_i(t)\|^2} V_{bi}(M_i \begin{pmatrix} x - x_{\mathbf{p}_i}(t) \\ y - y_{\mathbf{p}_i}(t) \end{pmatrix};\ t - t_i) \tag{7}$$

where $(\mathbf{p}_i(t_i);t_i)$ is the position at which the i^{th} bifurcation occurs and M_i is a transformation matrix specified by user to orient the moving direction of the two singularities. V_{bi} is one of the bifurcation normal forms (e.g. equations 1 and 2).

Accordingly, the obtained global time-varying vector field is the sum of these individual basis fields.

$$V(\mathbf{x};t) = \sum_i V_i(\mathbf{x};t) \tag{8}$$

Figure 8 provides a time-varying vector field generated using the element-based design and the basis field summation. Note that we extended the design elements to the space time domain from their static counterparts. Each design element at a given time acts as a static one except for a bifurcation element that is defined by its normal form. As such, the basis field summation is largely the same as its static counterpart. Consequently, the issue of the cancellation of an element by the influence of its nearby elements can arise. To relieve that, we can use a sharper fall off function with a larger d or require the design elements to be placed sufficiently far apart to reduce their mutual influence. Another possible solution is to extend the work of Turk and O'Brien [42] for surface modeling (i.e. scalar function modeling) to basis field summation. Specifically, we determine

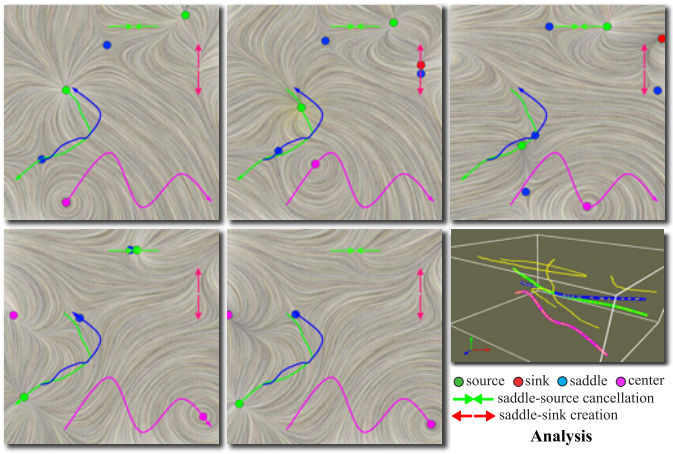
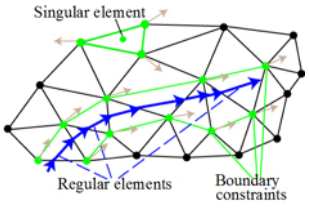


Fig. 8: A time-varying vector field generated using a number of design elements. The instantaneous fields are ordered from left to right and top to the bottom. The singularity paths of the singular elements are highlighted as the colored curves (green for source, blue for saddle, and magenta for center). Two saddle-node bifurcations are also inserted. The obtained field has smooth change over time as shown in the plot of the lower right. In addition to the desired singularities and bifurcations, there are also unexpected singularities and bifurcations as shown in the analysis, due to the nature of the basis field summation approach.

the weight for each basis field at a vertex and compute a weighted sum of the basis fields instead of a uniform sum. It is hoped this will preserve the prescribed features. However, it is unclear whether such an extension is easy to devise and how well it works for vector data. In this work, we resort to the *constrained optimization*, a popular vector field generation techniques for static field design [6], [8], [34], [49]. In particular, the constraints are set at the boundaries of a number of small and compact regions that enclose the design elements (see the inset).



5.3 Constrained Optimization

In static field design, constrained optimization solves for a harmonic vector field which satisfies the Laplacian system $\Delta\vec{\mathbf{V}} = \vec{0}$ where $\Delta = \nabla^2$ is the discrete Laplace operator and V is the unsolved vector field [51]. Particularly given a region N of a triangular mesh where the vector values at the boundary vertices of N are the constraints, the constrained optimization has the form of:

$$\overline{V}(v_i) = \sum_{j \in J} \omega_{ij} \overline{V}(v_j) \tag{9}$$

where v_i is an interior vertex, v_j's are its adjacent vertices in N. $\overline{V}(v)$ represents the average vector value at vertex v. ω_{ij} is the weight between vertex v_i and v_j. Note that we consider the boundary condition of Dirichlet type. Equation 9 is a sparse linear system in the form of $A\vec{\mathbf{x}} = \vec{\mathbf{b}}$. For fast solution, one can use uniform weighting scheme or mean curvature weighting [49] which guarantees A be a symmetric positive definite ($s.p.d.$) matrix such that the state-of-the-art Cholesky

factorization solver can be applied to solve it efficiently [48], [49]. In this case, we assume the vector values at vertices are expressed under the 3D global coordinate system. The solution of this setting will result in vector fields that do not always reside in the tangent space for a curved surface. Although we can project these vector fields to their tangent space, the projected field usually contains many unexpected singularities. In order to produce a tangential vector field with better quality (i.e. fewer unexpected singularities), we recommend the technique of parallel transport used in [27] to construct the Laplacian system in tangent space directly.

$$\overline{V}(v_i) = \sum_{j \in J} \omega_{ij} T_{ij} \overline{V}(v_j)$$

where T_{ij} is the transformation matrix for parallel transport along an edge (v_i, v_j). This will give rise to a non- $s.p.d.$ matrix. To solve it, we use a bi-conjugate gradient approach [32]. This also provides the foundation for our later extension to solve for the spatial-temporal problem.

Given the constrained optimization, the time-varying vector field can be generated by solving a sequence of Laplacian systems with the boundary constraints being set according to the instantaneous characteristics of the prescribed elements at the sampled times.

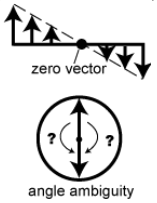
Although constrained optimization can better preserve the specified features, the basis field summation still has its value. In particular, a bifurcation can be specified easily in the similar fashion to a singular element with the provided normal forms as the templates. This is not the case using the constrained optimization by carefully specifying the vector values at the boundary of a region enclosing the bifurcation.

In the next two sections, we will describe two different design scenarios that complement the element-based design.

6 Key Frame Design

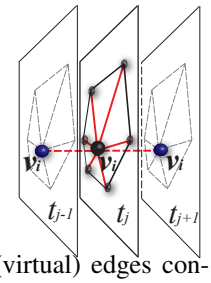
Given our discrete setting of time-varying vector fields, it is natural to prescribe the flow behavior in certain time steps and ask the system to create a time-varying vector field that smoothly transits from one state to the next. This leads to the *key-frame design*. This design scenario is useful when a number of critical time steps need to be designed to achieve the desired behaviors while the others are not so important. It is a widely used technique in computer animation community to efficiently generating reasonable animation sequence.

The key frame vector fields can be designed using any existing static vector field design techniques [6], [8], [34], [51]. In order to generate the rest of the time-varying vector field from the given key frame fields, some vector or angle based interpolation can be employed. However, using interpolation can either generate degenerate instantaneous fields (using vector-based linear interpolation) or discontinuities due to angle ambiguity (using angle-based interpolation). To address that, we introduce a spatial-temporal constrained optimization, an extension of the approach in Section 5.3.



6.1 Spatial-Temporal Constrained Optimization

Similar to the static case, we define an extended spatial-temporal Laplacian system by taking into account the additional parameter t. Note that discrete Laplacian system is constructed by considering the relation between spatially connected vertices. This can be extended to higher dimensions. Based on this observation, we treat t as the additional dimension in space and assume a direct connection between two neighboring vertices that are projected to the same vertex in one slice. The image to the right shows such a configuration. Given a vertex $(\mathbf{v}_i;t_j)$ in the underlying mesh in domain $\mathbf{X}$, consider a stencil shown in the figure to the right. We assume there are (virtual) edges connecting $(\mathbf{v}_i;t_j)$ and $(\mathbf{v}_i;t_{j-1})$, $(\mathbf{v}_i;t_j)$ and $(\mathbf{v}_i;t_{j+1})$, respectively. We solve a spatial-temporal Laplacian $\nabla \sum_j \omega_j V = \vec{\mathbf{0}}$ where $\sum_j \omega_j V$ represents the weighting sum of the time-varying vector field over t. This is equivalent to solve the following linear system:



$$\begin{aligned} \omega \overline{V}(\mathbf{v}_i;t_j) = \; & \textstyle\sum_{k \in N(i)} \omega_{ik} T_{ik} \overline{V}(\mathbf{v}_k;t_j) + \omega_{j,j-1} \overline{V}(\mathbf{v}_i;t_{j-1}) \\ & + \omega_{j,j+1} \overline{V}(\mathbf{v}_i;t_{j+1}) \end{aligned} \tag{10}$$

where $N(i)$ denotes the one-ring neighbors of $(\mathbf{v}_i;t_j)$, $\overline{V}(\mathbf{v}_i;t_j)$ represents the average vector value at position $(\mathbf{v}_i;t_j)$. $\omega_{j,j-1}$ and $\omega_{j,j+1}$ are positive weights determining how fast the relaxation is along t. In our implementation $\omega_{j,j-1}$=$\omega_{j,j+1} = b\sum_k \omega_{ik}$. $\omega = \sum_{k \in N(i)} \omega_{ik} + \omega_{j,j-1} + \omega_{j,j-1}$ is the normalization coefficient. b controls the speed of change along t. We use $b = 30$ for all examples. We point out that this formula can be further extended by considering more sampled steps along t axis to achieve smoother results as bi-Laplace smoothing does in time-independent case [8]. For better quality, we use parallel transport to solve for tangential vector fields with mean value weights [9].

With the spatial-temporal constrained optimization, a bifurcation can be implicitly created by specifying the instantaneous fields before and after bifurcations, e.g. the left and right fields in Figure 4. However, user is not able to specify the exact paths for both singularities, which can be achieved by the element-based design. Figure 9 shows a time-varying vector field generated using key frame design. It demonstrates that two vortices move toward each other, and finally collide and merge into one.

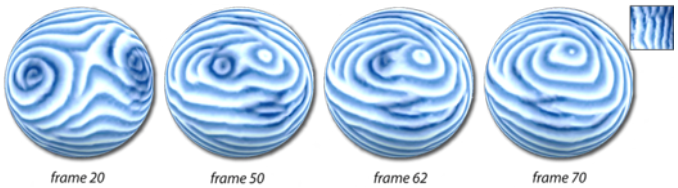


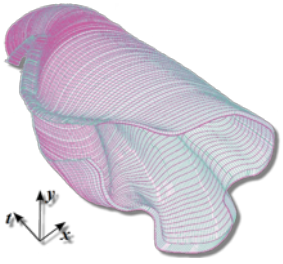
Fig. 9: This image shows a number of frames from a texture animation on sphere which simulates the collision of two storm systems. The animation is driven by an orientation field and an advection field, both are designed using the techniques introduced in this paper.

7 Time-Varying Field Deformation

In some design cases, the user is only given or has to start with an existing static field to generate a sequence of continuously changing fields over time. One way to achieve that is to *deform* the initial field gradually. This deformation process can be performed through physically-based simulation as one uses in fluid simulation. To incorporate the initial field in the simulator, it has to be considered as some external force field to start the simulation. The initial field has only indirect influence to the obtained sequence. This is not always desirable if the user prefers a time-varying vector field that is neither incompressible nor physically plausible. Therefore, other more intuitive and flexible approach needs to be explored. In this section, we propose a simple way to deform the initial field globally using a time-varying transformation, a matrix whose entries are functions of time. We refer to this approach as the *matrix-based design*.

Fig. 10: A transcritical bifurcation at the belly of the buddha using field deformation.

Our system assists such design by exerting a time-dependent transformation matrix on the initial field, $V(t) = M(t)V(t_0)$. $M(t)$ is an affine transformation of the form $\begin{pmatrix} M_{11}(t) & M_{12}(t) \\ M_{21}(t) & M_{22}(t) \end{pmatrix}$, where $M_{ij}(t)$ are some functions of t. $M(t)$ can be design through the graphics interface by specifying the x scaling, y scaling, and rotation similar to what is described in Section 5.1. In addition, our system also provides a text editor interface to allow the user to directly provide the numeric values for the four entries. The transformation matrix at t_i is then computed through linear interpolating the identity matrix and the user specified one. However, such random specification of transformation can easily lead to degenerate (e.g. zero or discontinuous) fields. Matrix-based design has its own value where the transcritical bifurcations can be achieved easily by rotating the Jacobians of the singularities over time. For instance, a transcritical bifurcation (i.e. source $\rightarrow$ center $\rightarrow$ sink) is induced at the belly of the buddha (Figure 10).



In addition to transforming the whole field, our system also allows the user to select one or more representative streamlines computed based on the initial field, and continuously transform them over time. The deformed representative streamlines are then used to generate new instantaneous fields at each sampled time using the static field generation techniques mentioned earlier. The inset figure above shows an example of a representative streamline (shown in magenta) deformed over time. A surface connecting the new position with the preceding one of this streamline is shown. The intersection of this surface with the $t = t_i$ plane is a smooth curve representing the streamline at t_i, which is used to generate an instantaneous field. However, more elegant and sophisticated technique should be devised to accomplish the deformation process to achieve more flexible and controllable results, which we will leave for the future work.

8 Local Topological Editing

Editing functionality is required for a design system because of the appearance of undesired features such as singularities and bifurcations in the generation phase. Our system provides the user with a number of options to edit a given time-varying vector field. First, the user can edit the instantaneous fields to modify the time-varying vector field. Second, the bifurcations can be removed or moved under certain conditions.

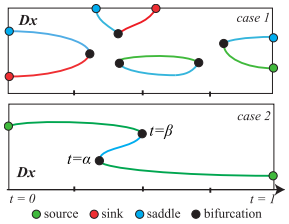
8.1 Instantaneous Field Topological Editing

Given the instantaneous field at a particular time t, the user can remove two unwanted singularities using the simplification techniques of Chen et al. [3]. This instantaneous field is then considered as a key frame for the regeneration of the field. Note that this editing process may potentially introduce more complex dynamics such as unintended bifurcations due to the weak constraint along the time axis (only one slice). We relieve this by smoothing the field within a box region center at t.

8.2 Bifurcation Editing

We have demonstrated the relations between saddle-node bifurcations and the structural changes in texture animations. We now describe techniques to control them. To do so, the system first extracts bifurcations from the designed fields using the techniques proposed by Tricoche et al. [40], and then provides two editing operations for the user.

Bifurcation Removal: Certain bifurcations can be removed. First, the user can remove the *isolated* bifurcations. According to our setting (Section 4) and Poincaré theorem, these isolated bifurcations can only involve singularities that start or end at the boundary of our computation domain $\mathbf{D_X}$, or the involving singularities form loops. The top row of the inset shows the removable isolated bifurcations. To remove them, we simply cancel the pairs of the involving singularities [3]. Second, if there are three singularities p_i $(i = 1,2,3)$ with interval of existence $(0,\beta)$, (α,β), $(\alpha,1)$, respectively. Assume a saddle-node bifurcation between p_1, p_2 at β and a saddle node bifurcation between p_2 and p_3 at α. We then can remove both bifurcations and retain only one singularity. The bottom row of the inset shows such an example. The two bifurcations that are connected by the blue curve (the path of a saddle) can be collapsed. Figure 11 shows an example of saddle-node bifurcation removal. More complex control of non-isolated (i.e. connected) bifurcations is possible, which is beyond the scope of this paper.



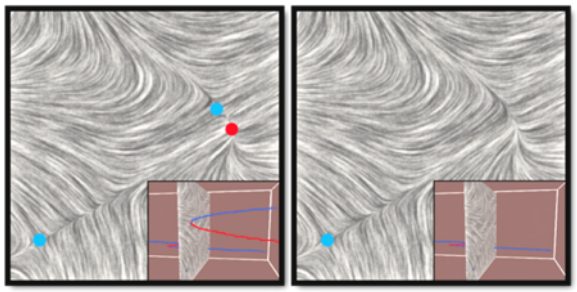


Fig. 11: Example of bifurcation editing.

Bifurcation Movement: Similar to singularities, a bifurcation can be moved using our system. This can be achieved by moving the involving singularities over space at particular time t. The edited instantaneous field is then set as a key frame. The spatial-temporal constrained optimization will smooth the rest of the field. Note that the movement of these two singularities should obey the topological constraints proposed by Zhang et al. [51]. This guarantees no other topological features be affected during the movement. This functionality is particularly useful when the bifurcation is not isolated and causes visual discontinuity (e.g. Figure 3). The bifurcation movement could move it to the invisible part of the object.

General global smoothing over spatial-temporal domain is also available similar to the smoothing scheme of [18] for fining the edge fields, i.e. some tensor fields, in the application of painterly rendering of videos.

9 Applications and Discussion

In this section, we present a number of graphics applications that can be beneficial from the time-varying vector fields generated using the proposed techniques.

Texture Synthesis and Animation

We have applied the designed time-varying fields to creating a number of synthetic texture animations (Figures 3 and 9). Flow-guided texture synthesis and advection has been introduced to visualization community for dense flow visualization by van Wijk [44], [43], Laramee et al. [22], and Neyret [26]. Kwatra et al. [20] present an optimization-based plane texture synthesis which can be used for flow-guided texture animation. Lefebvre and Hoppe [23] introduce an appearance-space texture synthesis technique that can handle texture advection over static surfaces. Han et al. [13] extend the work of [20] to 3D mesh surfaces. Later, Kwatra et al. [19] and Bargteil et al. [2] extend the advected texturing techniques to the problem of fluid texturing on surfaces, respectively. Recently, Ma et al. [24] introduce a texture synthesis technique for flow patterns to create a more detailed synthetic texture and animation. In this paper, we employ the technique of Kwatra et al. [19].

To apply the created time-varying vector fields, we make use of the instantaneous snapshots of the time-varying vector fields to orient and move the texture patches on surfaces. Note that using a height map as input to the texture synthesis, we can create similar effect of time-varying geometry by displacing the mesh vertices based on the synthetic height map. Figure 12 demonstrates an evolving terrain effect using a height map as the input to the texture synthesis and advection program. The underlying field in this example was generated using the field deformation where a streamline representing one mountain range was deformed over time.

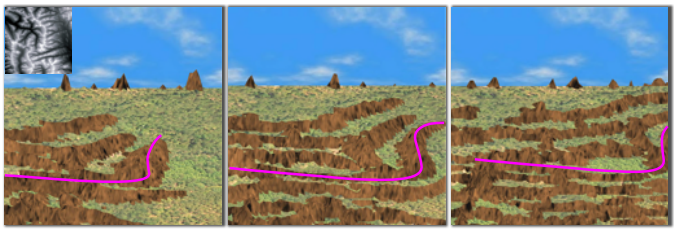


Fig. 12: A growing terrain driven by a designed time-varying vector field using field deformation. Note that how the reference streamline (magenta) evolves over time. The input texture exemplar is shown in the upper left corner of the first image.

Creation of Dynamic Scenes

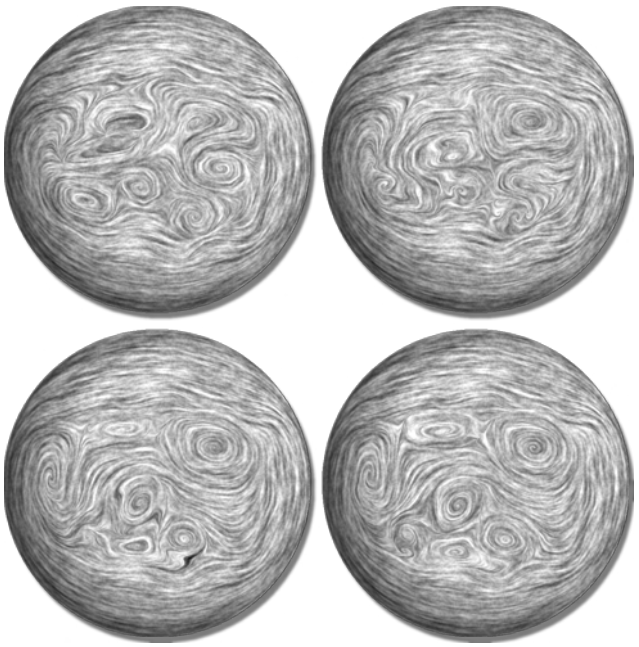


Fig. 13: A time-varying incompressible flow on the sphere.

Our system can be used to create fluid effects on surfaces through proper design and setting of the boundary conditions. Figure 13 shows an incompressible flow on the sphere generated using our system. In addition, the present system allows the creation of more complex dynamic effects such as wind writing on a meadow (Figure 14), the advection of leaves in the fluid flow with self-spinning effect (Figure 15). In Figure 14, instead of adding a physically realistic winds, we design a time-varying vector field which mimics writing on the grass. To achieve that, the user first specifies the flow to represent the writing of the letters. The system automatically records the vector fields as key frames during the sketching of these letters. The spatial-temporal constrained optimization is then used to solve for a time-varying vector field. Each strand of the grass is represented as a rigid body skeleton. The bottom

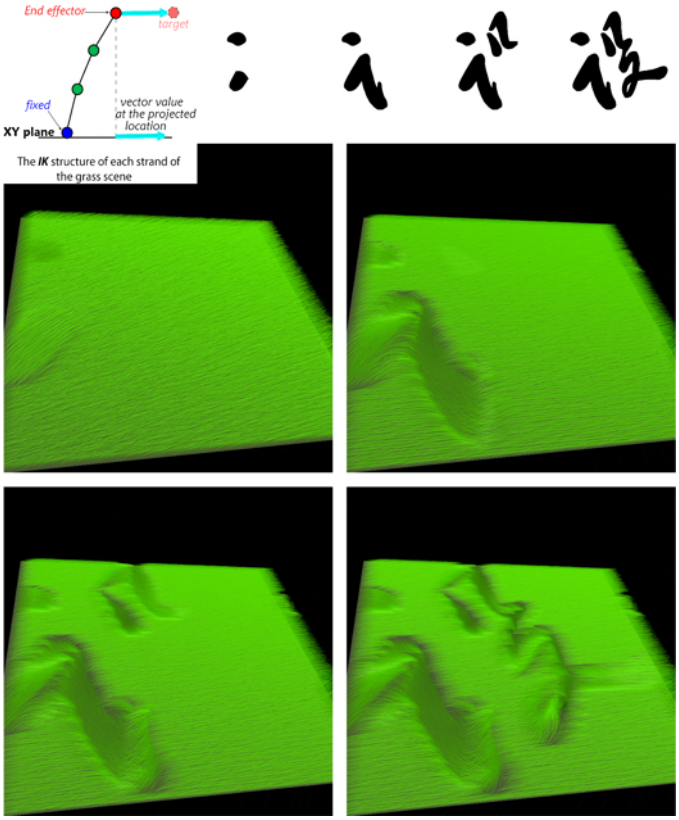


Fig. 14: An animation of writing on the grass. The dynamics of the grass is driven by a created time-varying vector field. The grass consists of over 32,000 strands, each of which has the structure shown in the top left corner.

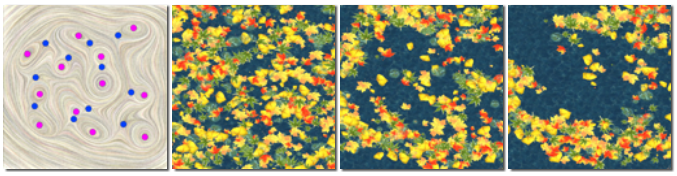


Fig. 15: An animation of the fallen leaves advected by a time-varying flow. The leaves are self-spinning according to the advection flow and their orientation direction. This scene contains 1000 particles. Please see the accompanying video for the spinning effect.

of the skeleton is fixed on the ground while the top node is manipulated by a force field which is the created time-varying vector field. The movement of this skeleton is computed by an inverse kinematic solver [28]. The grass is rendered using the technique of illuminated lines [25]. The field used to drive the movement of the leaves (Figure 15) was created through element-based design. The spinning effect is achieved by maintaining a constant angle between the *up* direction of a particle and the advection direction at the given position.

Steerable 2D Crowd Animation

Crowd simulation is an important technique in games, movies, and urban planning. There are two groups of crowd simulation techniques: agent-based and force-based. While agent-based method can provide more detailed and realistic simulation, it is still prohibitively expensive in the simulation

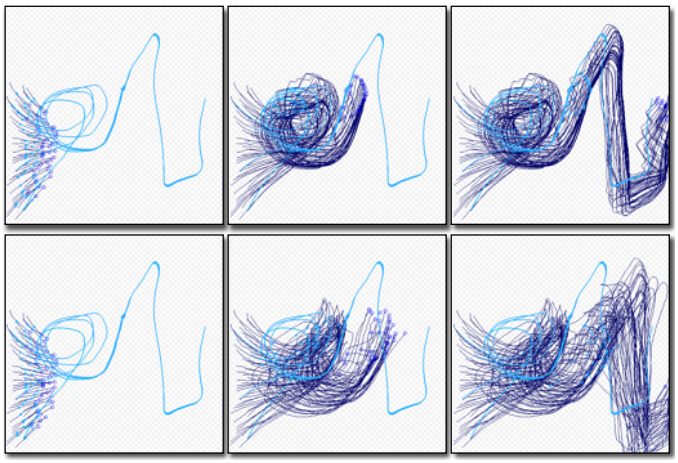


Fig. 16: This example demonstrates a crowd simulation driven by the combination of a social force G with a designed time-varying vector field F. The top shows the results of the combination $0.4G+0.6F$, and the bottom is $0.53G+0.47F$. The cyan curves are the reference pathlines based on the initial positions of the pedestrians and the underlying time-varying flow. The brown curves are the actual paths that the pedestrians have taken.

of large number of pedestrians under a complex environment. In contrast, the force-based technique considers the pedestrians in the crowd as particles. Their movement is determined by computing the gradient of certain cost field by taking into account the environment and neighboring people. This method is fast at the expense of losing the detailed behavior of the individual pedestrians. Both approaches support the control of initial state yet lack of the continuous steering of the crowds over time. Recently, Patil et al. [29] propose to make use of a navigation field (essentially a vector field) to control the traveling paths of groups of pedestrians, which has achieved better control of crowds. We further observe that the paths of the individual pedestrians can be considered as pathlines, thus can be designed and controlled using our system. In our steerable crowd simulation, the crowds are driven by both a gradient field G derived from the cost function introduced in the continuum crowd technique [39] and a designed time-varying field F. The final direction that each pedestrian will take is the weighted sum of these two fields $\omega_G G + \omega_F F$. Different combinations of weights will determine how closely the crowd follows the specified paths. In the example shown in Figure 16, we compare the results of different combinations: $\omega_G = 0.4, \omega_F = 0.6$ (top) and $\omega_G = 0.53, \omega_F = 0.47$ (bottom). The swirling pattern of the paths was created with purpose to show the difference between pathlines and streamlines. No streamlines can achieve such self-intersecting patterns.

Artistic Painterly Animation

In painterly rendering the brush stroke orientations are typically guided by a vector field [16], [14], [51]. A time-varying vector field can also be applied to a static image to achieve animating effect in certain regions, such as background, to make the static photo seem alive [5]. Figure 1 provides such an example. The effect of the evolution of one vortex is inserted to the lower part of the painting to provide artistic like water

animation. The input time-varying vector field is used to orient the brush strokes as well as advecting them along the flow directions. Figure 17 shows another example where several vortices are inserted to provide a burning effect to the original steady image. These vortices interact with each other and eventually collapse into a large vortex in the center. Both fields of these two examples were created using key frame design, although they could also be generated using element-based design.

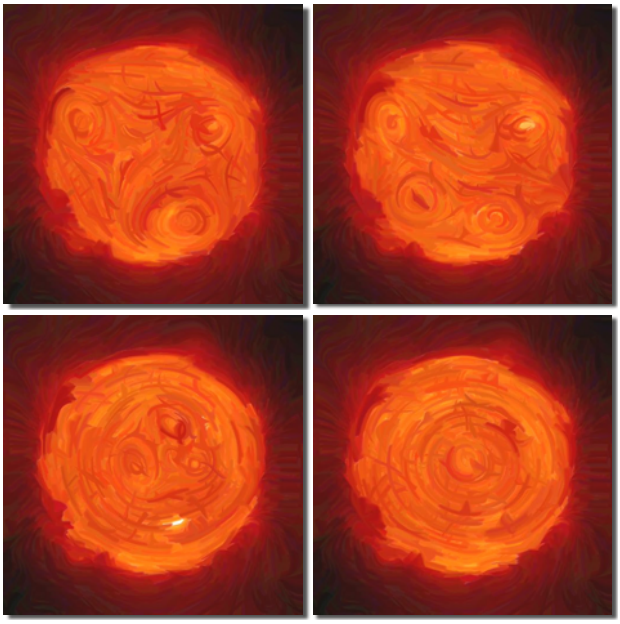


Fig. 17: The effect of a burning sun.

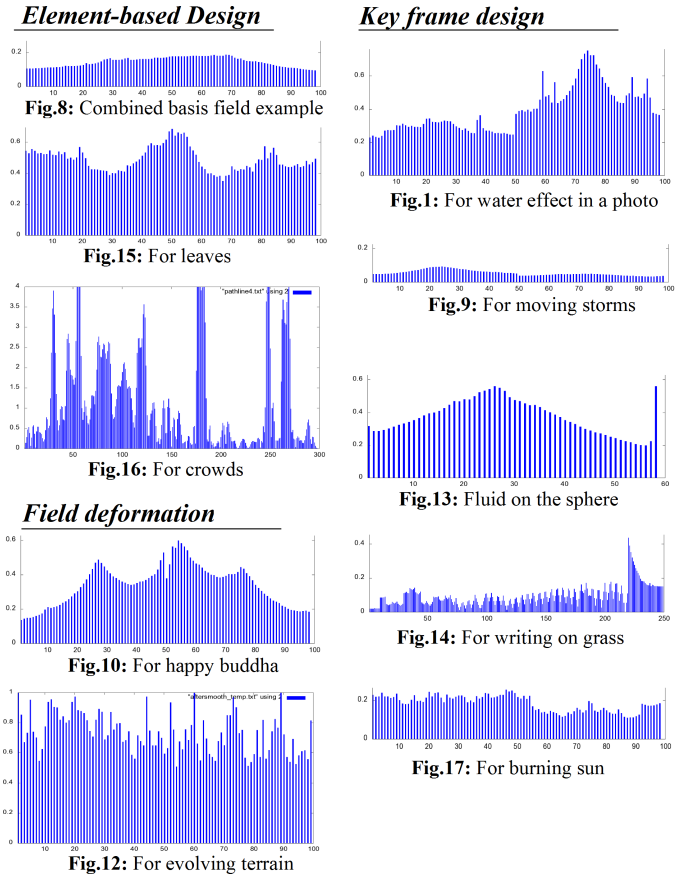


Fig. 18: This figure shows the coherence plots of the time-varying vector fields used in the papers.

Performance

For all the examples shown in this paper, the initialization of a planar time-varying field with 100 frames defined on a 65×65 regular grid typically takes less than 5 seconds on a 3 GHz PC with 4GB RAM. For the design on surface (up to $20,000$ vertices), it can take up to 3.5 minutes to generate the field with 100 frames without optimization and with error threshold $1.e-10$ and maximum iteration number 400 for the bi-Conjugate Gradient solver. Note that a direct solver could be applied, such as the Cholesky decomposition. However, when a long sequence of time-varying vector field is created on a large mesh, the memory usage may not be efficient for such direct decomposition method.

Evaluation and Discussion: To evaluate the generated time-varying vector fields, for each example field used in the paper, we display a plot showing the change of the instantaneous field over time (Figure 18). This plot allows us to visualize the temporal coherence of a time-varying vector field. The X axis of each plot is the frame index and the Y axis is the total change of the vector field computed as $y(x_i) = \sum_k ||V_k(t_{i+1}) - V_k(t_i)||$ where $V_k(t_i)$ is the vector value at vertex k at time t_i. According to our smoothness assumption in the introduction, the smaller the $y(x_i)$, the slower and smoother the change is. From the plots, we can see that the fields generated using the key frame design combined with the spatial-temporal Laplacian are typically smooth because of its energy minimization nature. The element-based method could generate fields with larger fluctuation due to the occurrence of unexpected features or insufficient sampling along a long integral curves (e.g. the pathline design shown in Figure 16). Field deformation also generates fields with large variations over time. This is because of the large change of the transformation matrix or the reference streamline in succeeding times (e.g. the streamline deformation used in Figure 12). However, this issue is not fundamental and can be resolved by simply increasing the time sampling to capture the smooth transition of the features. In terms of which design scenario should be used for a given situation, it is application dependent. It is determined by what and how the graphics properties need to be controlled in the specific applications, as demonstrated through various applications in this section. For instance, if the path of certain local graphics primitive (e.g. a vortex in Figure 5 and the path of a group of pedestrian in Figure 16) needs to be controlled exactly, the element-based design can be employed. If the exact states at some desired times have to be met (e.g. the writing on the grass in Figure 14), the key frame design is more suitable. In addition, element-based design and field deformation approach may provide full control of the local behaviors of the flow at and near the prescribed elements and the representative streamline, but could be labor intensive if the number of local patterns that need to be controlled is large. Key frame design is effective if the instantaneous appearance

is the main goal and only a few instantaneous fields at the desired times are required to meet, but is lack of the control of the rest of the field. An ideal solution will be the combination of these different approaches to devise a more flexible and thorough design framework. We plan to investigate this in the future work.

10 Conclusion and Future Work

This paper addresses the problem of the design of time-varying vector field on 2D domains. We have identified a set of design requirements as well as provided a system and a user interface for the design of 2D time-varying vector fields. A number of design primitives are then discussed based on the requirements of two different types of vector fields, i.e., orientation and advection fields, for different purposes in computer graphics. Efficient algorithms are introduced to generate time-varying vector fields from the user specified design elements. A number of editing operations with certain topological guarantees are introduced to enable the adjustment of the initial fields. To our knowledge, the presented design framework is the first of its kind for general time-varying vector field design with bifurcation control. This work opens a new range of the field design topics which can be extended to the higher order time-varying tensor field design [27]. For instance, the same framework can be easily modified to handle the design of time-varying tensor fields by extending the time-varying singular elements to the elements for degenerate points in the element-based design or solving a tensor-based spatial-temporal Laplacian in the key frame design.

There are a number of future research directions. First, the present generation techniques do not guarantee the desired topology over time, especially for key frame design. Only the topology at the key frame fields are defined. More comprehensive control of topology in between key frame fields is needed. Second, the bifurcation design is an important component in time-varying vector field design as shown in the paper. More flexible and sophisticated design technique for bifurcations are desired. We also wish to extend our system to handle more types of bifurcations that may involve more sophisticated objects such as periodic orbits and separation and attachment lines. Third, we plan to explore other flow descriptors including streaklines, timelines [7], and Lagrangian coherent structures [12]. Fourth, comprehensive combinations of the proposed design functionality, such as combined design with simulation, should be studied to support more complex design tasks in the future. Finally, extending the design techniques for 2D fields to 3D domain will be challenging yet more beneficial for computer graphics.

Acknowledgments

We would like to thank Dr. Konstantin Mischaikow for the valuable discussion on the topology and dynamics of vector fields, which initiated this work. We also thank Dr. Mark van Langeveld for the valuable discussion on potential applications. This work was supported by NSF IIS-0546881, CCF-0830808 awards. Guoning Chen was partially supported by King Abdullah University of Science and Technology (KAUST) Award No. KUS-C1-016-04 and DOE VACET.

References

[1] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics*, 22(3):485–493, 2003.

[2] A. W. Bargteil, F. Sin, J. E. Michaels, T. G. Goktekin, and J. F. O'Brien. A texture synthesis method for liquid animations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 345–351, Sept 2006.

[3] G. Chen, K. Mischaikow, R. S. Laramee, P. Pilarczyk, and E. Zhang. Vector field editing and periodic orbit extraction using morse decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):769–785, 2007.

[4] S. Chenney. Flow tiles. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 233–242. Eurographics Association, 2004.

[5] Y.-Y. Chuang, D. B. Goldman, K. C. Zheng, B. Curless, D. H. Salesin, and R. Szeliski. Animating pictures with stochastic motion textures. *ACM Transactions on Graphics*, 24(3):853–860, 2005.

[6] K. Crane, M. Desbrun, and P. Schröder. Trivial connections on discrete surfaces. *Computer Graphics Forum (SGP)*, 29(5):1525–1533, 2010.

[7] T. Faber. *Fluid Dynamics for Physicists*. Cambridge University Press, 1995.

[8] M. Fisher, P. Schröder, M. Desbrun, and H. Hoppe. Design of tangent vector fields. *ACM Transactions on Graphics*, 26(3):56:1–56:9, 2007.

[9] M. S. Floater. Mean value coordinates. *Comput. Aided Geom. Des.*, 20(1):19–27, 2003.

[10] H. Fu, Y. Wei, C.-L. Tai, and L. Quan. Sketching hairstyles. In *SBIM '07: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pages 31–36. ACM, 2007.

[11] J. Hale and H. Kocak. *Dynamics and Bifurcations*. New York: Springer-Verlag, 1991.

[12] G. Haller. Finding finite-time invariant manifolds in two-dimensional velocity fields. *Chaos*, 10(1):99–108, 2000.

[13] J. Han, K. Zhou, L.-Y. Wei, M. Gong, H. Bao, X. Zhang, and B. Guo. Fast example-based surface texture synthesis via discrete optimization. *Vis. Comput.*, 22(9):918–925, 2006.

[14] J. Hays and I. Essa. Image and video based painterly animation. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 113–120, New York, NY, USA, 2004. ACM.

[15] J. L. Helman and L. Hesselink. Representation and Display of Vector Field Topology in Fluid Flow Data Sets. *IEEE Computer*, 22(8):27–36, August 1989.

[16] A. Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460, New York, NY, USA, 1998. ACM.

[17] A. Hertzmann and K. Perlin. Painterly rendering for video and interaction. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 7–12, New York, NY, USA, 2000. ACM.

[18] M. Kagaya, W. Brendel, Q. Deng, T. Kesterson, S. Todorovic, P. J. Neill, and E. Zhang. Video painting with space-time-varying style parameters. *IEEE Transactions on Visualization and Computer Graphics*, 17(1):74–87, 2011.

[19] V. Kwatra, D. Adalsteinsson, T. Kim, N. Kwatra, M. Carlson, and M. Lin. Texturing fluids. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):939–952, 2007.

[20] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics, SIGGRAPH 2005*, pages 795–802, August 2005.

[21] Y.-K. Lai, M. Jin, X. Xie, Y. He, J. Palacios, E. Zhang, S.-M. Hu, and X. Gu. Metric-driven rosy field design and remeshing. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):95–108, 2010.

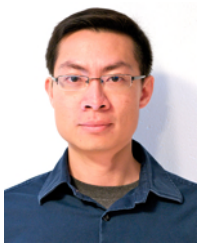
[22] R. S. Laramee, B. Jobard, and H. Hauser. Image space based visualization of unsteady flow on surfaces. In *Proceedings IEEE Visualization '03*, pages 131–138. IEEE Computer Society, October 2003.

[23] S. Lefebvre and H. Hoppe. Appearance-space texture synthesis. *ACM Trans. Graph.*, 25(3):541–548, 2006.

[24] C. Ma, L.-Y. Wei, B. Guo, and K. Zhou. Motion field texture synthesis. *ACM Transactions on Graphics, (Proceedings SIGGRAPH Asia 2009)*, 28(5):110:1–110:8, 2009.

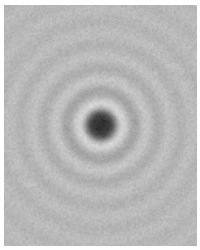
[25] O. Mallo, R. Peikert, C. Sigg, and F. Sadlo. Illuminated lines revisited. In *Proceeding of IEEE Visualization 2005*, pages 19–26, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[26] F. Neyret. Advected textures. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 147–153, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
[27] J. Palacios and E. Zhang. Rotational symmetry field design on surfaces. *ACM Transactions on Graphics*, 26(3):56:1–56:10, 2007.
[28] R. Parent. *Computer Animation, Second Edition: Algorithms and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.
[29] S. Patil, J. van den Berg, S. Curtis, M. C. Lin, and D. Manocha. Directing crowd simulations using navigation fields. *IEEE Transactions on Visualization and Computer Graphics*, 17:244–254, February 2011.
[30] F. Pighin, J. M. Cohen, and M. Shah. Modeling and editing flows using advected radial basis functions. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 223–232, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.
[31] E. Praun, F. Adam, and H. Hugues. Lapped textures. In *Proceedings of ACM SIGGRAPH 2000*, pages 465–470, July 2000.
[32] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 1992.
[33] N. Ray, W. C. Li, B. Lvy, and A. S. an d Pierre Alliez. Periodic global parameterization. *ACM Transactions on Graphics*, 25(4):1460–1485, 2006.
[34] N. Ray, B. Vallet, W.-C. Li, and B. Levy. N-symmetry direction field design. *ACM Transactions on Graphics*, 27(2):10:1–10:13, 2008.
[35] J. Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999.
[36] J. Stam. Flows on surfaces of arbitrary topology. volume 22, pages 724–731, New York, NY, USA, July 2003. ACM.
[37] H. Theisel. Designing 2d vector fields of arbitrary topology. *Computer Graphics Forum (Proceedings Eurographics 2002)*, 21(3):595–604, July 2002.
[38] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Topological methods for 2d time-dependent vector fields based on stream lines and path lines. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):383–394, 2005.
[39] A. Treuille, S. Cooper, and Z. Popović. Continuum crowds. *ACM Transactions on Graphics*, 25(3):1160–1168, 2006.
[40] X. Tricoche, G. Scheuermann, and H. Hagen. Topology-based visualization of time-dependent 2d vector fields. In *Data Visualization 2001 (Joint Eurographics-IEEE TCVG Symposium on Visualization Proceedings)*, pages 117–126, 2001.
[41] G. Turk. Texture synthesis on surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 2001)*, pages 347–354, 2001.
[42] G. Turk and J. F. O'brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, 21(4):855–873, October 2002.
[43] J. van Wijk. Image based flow visualization for curved surfaces. In *Proceedings IEEE Visualization '03*, pages 123–130. IEEE Computer Society, 2003.
[44] J. J. van Wijk. Image based flow visualization. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 745–754. ACM, 2002.
[45] W. von Funck, H. Theisel, and H.-P. Seidel. Vector field based shape deformations. *ACM Transactions on Graphics*, 25(3):1118–1125, 2006.
[46] L. Y. Wei and M. Levoy. Texture synthesis over arbitrary manifold surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 2001)*, pages 355–360, 2001.
[47] J. Wejchert and D. Haumann. Animation aerodynamics. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '91, pages 19–22. ACM, 1991.
[48] K. Xu, D. Cohne-Or, T. Ju, L. Liu, H. Zhang, S. Zhou, and Y. Xiong. Feature-aligned shape texturing. *ACM Transactions on Graphics (SIGGRAPH Asia 2009)*, 28(5):108:1–108:7, 2009.
[49] K. Xu, H. Zhang, D. Cohen-Or, and Y. Xiong. Dynamic harmonic fields for surface processing. *Computers and Graphics (Special Issue of Shape Modeling International)*, 33:391–398, 2009.
[50] L. Xu, J. Chen, and J. Jia. A segmentation based variational model for accurate optical flow estimation. pages 671–684, 2008.
[51] E. Zhang, K. Mischaikow, and G. Turk. Vector field design on surfaces. *ACM Transactions on Graphics*, 25(4):1294–1326, 2006.



Guoning Chen received a bachelors degree in 1999 from Xi'an Jiaotong University, China and a masters degree in 2002 from Guangxi University, China. In 2009, he received a PhD degree in computer science from Oregon State University. His research interests include scientific visualization, computational topology, and computer graphics. Currently, he is a post-doctoral research associate in Scientific Computing and Imaging (SCI) Institute at the University of Utah. He is a member of the IEEE.

Vivek Kwatra Vivek Kwatra received the BTech degree in computer science and engineering from the Indian Institute of Technology (IIT) Delhi, India, in 1999 and the MS and PhD degrees in computer science from the Georgia Institute of Technology in 2004 and 2005, respectively. He was a postdoctoral researcher in the Computer Science Department at the University of North Carolina, Chapel Hill, from 2005 to 2007. He is currently working at Google as a research scientist.



Li-Yi Wei is a researcher with Microsoft Research and an associate professor at The University of Hong Kong. Before that he has been architecting graphics chips with NVIDIA until 2005. He obtained Ph.D. from Stanford in 2001. He likes computer graphics because it covers a variety of subjects in art, science, and engineering. He is also interested in parallel computing, hardware architecture, human computer interaction, or whatever subjects that his collaborators force him to learn.

Charles D. Hansen received a BS in computer science from Memphis State University in 1981 and a PhD in computer science from the University of Utah in 1987. He is a professor of computer science at the University of Utah an associate director of the SCI Institute. From 1989 to 1997, he was a Technical Staff Member in the Advanced Computing Laboratory (ACL) located at Los Alamos National Laboratory, where he formed and directed the visualization efforts in the ACL. He was a Bourse de Chateaubriand PostDoc Fellow at INRIA, Rocquencourt France, in 1987 and 1988. His research interests include large-scale scientific visualization and computer graphics.

Eugene Zhang received the PhD degree in computer science in 2004 from Georgia Institute of Technology. He is currently an associate professor at Oregon State University, where he is a member of the School of Electrical Engineering and Computer Science. His research interests include computer graphics, scientific visualization, geometric modeling, and computational topology. He received an National Science Foundation (NSF) CAREER award in 2006. He is a member of the IEEE and a senior member of ACM.