Time-Varying Vector Field Design on Surfaces

Guoning Chen * Konstantin Mischaikow † Vivek Kwatra ‡ Eugene Zhang*

*Oregon State University †Rutgers University ‡ Google Inc.

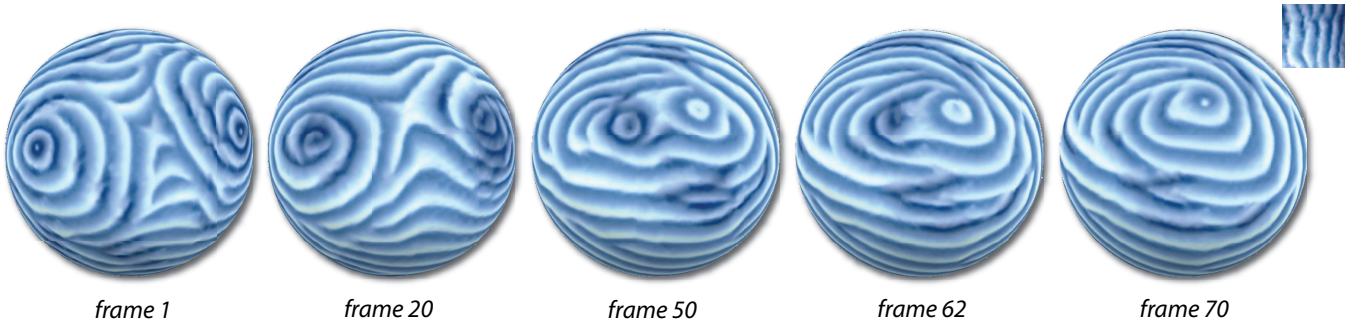


Figure 1: *This image shows a number of frames from a texture animation on sphere which simulates the collision of two storm systems. The animation is driven by an orientation field and an advection field, both are designed using the techniques introduced in this paper. Note that two vortex-like patterns (frame 1) are displaced by the advection field at the middle of the squence (frames 20 and 50). The two storms are then merged (frame 62) and become one system (frame 70).*

Abstract

Vector field design has a wide variety of applications in computer graphics, including texture synthesis, non-photorealistic rendering, fluid and crowd simulation, and shape deformation. This paper addresses the problem of the design of time-varying vector fields on surfaces. The additional time dimension poses a number of unique challenges to the design tasks such as the introduction of more complex structural changes. To address these challenges, we present a number of novel techniques to enable efficient design over important characteristics in the vector field such as *singularity paths*, *pathlines*, and *bifurcations*. These vector field features are used to generate a vector field by either blending basis vector fields or performing a constrained optimization process. Unwanted singularities and bifurcations can lead to visual artifacts, and we address them through singularity and bifurcation editing. We demonstrate the capabilities of our system by applying it to the design of two types of vector fields: orientation field and advection field for the application of texture synthesis and animation.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism I.6.3 [Simulation and Modeling]: Applications J.6 [Computer-Aided Engineering]: Computer-Aided Design (CAD)

Keywords: time-varying, vector fields, bifurcations, vector field design, texture synthesis and animation

1 Introduction

A wide variety of computer graphics applications require vector fields as the input, such as texture synthesis [Praun et al. 2000; Turk 2001; Wei and Levoy 2001; Kwatra et al. 2005; Lefebvre and Hoppe 2006; Zhang et al. 2006; Fisher et al. 2007], non-photorealistic rendering [Hertzmann 1998; Hertzmann and Zorin 2000; Zhang et al. 2006], fluid simulation [Stam 2003], hair modeling [Fu et al. 2007], crowd animation [Chenney 2004], and shape deformation [von Funck et al. 2006]. The design and control of steady (time-independent) vector fields on two dimensional manifolds has been well explored in the recent years ([Zhang et al. 2006; Fisher et al. 2007; Chen et al. 2007]). In contrast, there has been relatively little work in designing time-varying vector fields despite the potential benefits in applications such as controlled fluid and crowd simulation, shape deformation design, hair animation, artistic rendering of videos, and texture synthesis and animation. We address the problem of the design of time-varying vector fields on surfaces in this paper.

To be more precise, it has been observed that in time-varying graphics applications solving for a series of time-independent vector fields to produce each frame can lead to visual discontinuity artifacts. It has also been demonstrated that these types of artifacts are less prevalent when graphic applications are generated by solving a time-varying vector field. This suggests the desirability of designing time-varying vector field $V(\mathbf{x},t)$ with specific properties. This poses a number of challenges to the design task. For example, a number of fundamental concepts, such as a singularity of a vector field, used in steady field design are no longer mathematically well-defined in the time-varying setting. More importantly, the theory for the analysis and control of time-varying dynamics is much more primitive than that for time-independent vector fields. On the other hand it can be shown that solutions to the time-varying vector field converge to families of solutions of the instantaneous vector fields as the rate of change goes to zero. This observation motivates the approach we take in this paper. We apply previously developed tools of steady vector field design to a *one-parameter family of vector fields* $V(\mathbf{x},\lambda)$ (referred to as a parameterized vector field) with the desired instantaneous dynamics but then for the graphics applications treat it as a time-varying vector field.

System Overview: Based on this philosophy, we conduct the design of the parameterized vector fields using a three-stage pipeline. Figure 2 provides an illustrative diagram for this pipeline. In the first stage, our system supports the creation of a parameterized vector field through a number of types of user specifications, such as *singularities*, *streamlines*, *singularity paths*, *pathlines*, and *bifurcations*. These specifications are either converted into basis fields and summed or treated as constraints of a relaxation process (Section 4). In the second step, the system analyzes the initial field and

*{chengu|zhange}@eecs.oregonstate.edu
†mischaik@math.rutgers.edu
‡kwatra@google.com

provides necessary feedback to the user for further editing. To enable control over unwanted flow behaviors such as singularities and bifurcations, we provide topological editing functionalities to remove them or move them to more desirable locations in spacetime in the third stage (Section 5). The techniques that we use in vector field analysis (stage 2) is based on previous research. Thus, we will only describe the details of the initialization (stage 1) and editing (stage 3) of our system.

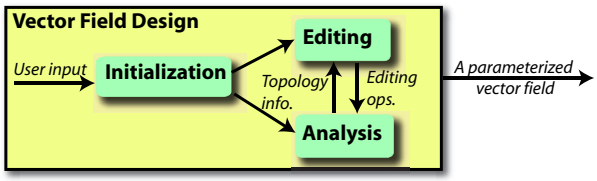


Figure 2: *The design pipeline.*

A vector field can be used to describe many physical phenomena such as displacement, velocity, acceleration, orientation, and momentum. The definitions of features in a vector field is application-dependent. For example, in texture synthesis and animation, one vector field is used to guide the placement of texture patches, while another is used to movet the patches. Therefore, understanding the differences between vector fields involved and separating their design tasks will not only ease the discussion of different design interfaces and primitives, but also enable us to achieve various effects. In this paper, we demonstrate the utility of this by separating orientation and advection field design in texture synthesis and animation.

Contributions: Our work yields the following contributions.

- We have identified time-varying vector field design as an important problem in computer graphics. We present an approach to this problem by designing parameterized vector fields to approximate time-varying vector field design. This enables the techniques for steady vector field design to be extended to assist time-varying vector field design.
- We describe the disparate usage of various vector field characteristics in diverse computer graphics applications. This distinction shows the need of design of different types of time-varying vector fields to serve diverse graphical purposes. Particularly, we show the design of orientation field and advection field for the orientation and movement of the texture structures in texture synthesis and animation.
- We present a number of techniques to enable different vector field features such as singularity paths, pathlines, and bifurcations, to be input as user specifications. The extended radial basis field approach and an extended constrained optimization technique are introduced to produce a parameterized vector field with coherent transition.
- We provide operations to support topological control in a parameterized vector field, including bifurcation removal and movement, and singularity movement.

Paper Structure: We will review the most related work in Section 2. Section 3 will review important concepts and theories on time-varying vector fields and parameterized vector fields. The detailed description of the design system and techniques will be presented in Sections 4 and 5, respectively. Finally, the results and discussion of the application of texture synthesis and animation will be addressed in Section 6, followed by the conclusion and a discussion on future work in Section 7.

2 Related Work

Vector field design refers to creating a continuous vector field on a 2-manifold based on user specifications or application-dependent requirements. 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 do not address vector field topology explicitly. The vector field design tools in the early applications of texture synthesis [Turk 2001; Wei and Levoy 2001], fluid simulation [Stam 2003], and visualization [van Wijk 2003] are the examples of this category. In the mean time, other applications, such as non-photorealistic rendering [Zhang et al. 2006], remeshing [Alliez et al. 2003], and parameterization [Ray et al. 2006] also employ field design, respectively. Topology-based approaches provide the user the ability to control the number and positions of singularities [van Wijk 2002; Zhang et al. 2006; Fisher et al. 2007] or the topological graph [Theisel 2002]. Recently, more general N-way rotational symmetry field design has been studied by Palacios and Zhang [2007] and Ray et al. [2008]. Most of this work concerns time-independent (i.e. steady) vector fields only. Some applications generate time-varying vector fields automatically during execution without providing the user an intermediate interface, such as fluid simulation [Stam 1999], crowd animation [Treuille et al. 2006], shape deformation [von Funck et al. 2006], and hair modeling [Fu et al. 2007] and video editing [Xu et al. 2008]. However, we are not aware of any work on the design of time-varying vector fields for the general purpose of graphics applications.

3 Background

To better understand the parameterized vector field and provide necessary background for our later discussion, we review a number of important concepts briefly in this section.

Definition of a 2D Parameterized Vector Field: Consider a 2-manifold M. A parameterized vector field is defined as a series of instantaneous vector fields with respect to a parameter λ, denoted by $V(\mathbf{x};\lambda)$, where $\mathbf{x} \in M$ and $\lambda \in \mathbf{R}$. An instantaneous field at λ_c is denoted by $V(\mathbf{x};\lambda_c)$. There is considerable freedom to move from the parameterized family of vector fields $V(\mathbf{x},\lambda)$ to a time-varying system. For example, let $g : \mathbf{R} \to \mathbf{R}$ be any smooth positive function. Then

$$\frac{d\mathbf{x}}{dt} = V(\mathbf{x},\lambda), \quad \frac{d\lambda}{dt} = g(\lambda)$$

is a time-varying system. As a first approximation the reader can assume that we are using the time-varying system $\frac{d\mathbf{x}}{dt} = V(\mathbf{x},\lambda)$, $\frac{d\lambda}{dt} = s$ where s is a positive constant. In other words, parameter λ and physical time t possess a linear relation such that $d\lambda = s \cdot dt$. Therefore, if $s = 1$, λ is equivalent to t. Further, it shows when $s \to 0$, the variation of the vector field is sufficiently small which guarantees a smooth transition. This is particularly useful for graphics applications where an input field with smooth transition over time is expected to achieve visually coherent results. It also reveals the relation between a time-varying vector field and a corresponding parameterized vector field which enables the discussion of certain concepts under the parameterized vector field framework, such as *pathlines*.

Instantaneous Topology: A point $\mathbf{p}$ is called a *singularity* at λ_j if $V(\mathbf{p};\lambda_j) = 0$. The linearization of $V(\mathbf{x};\lambda_j)$ about $\mathbf{p} \in M$ results in a 2×2 matrix $DV(\mathbf{p})$ (known as *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*. 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* $0 < \sigma_1, \sigma_2$. A *streamline* passing through $\mathbf{p} \in M$ at λ_j is defined as the integral curve computed under the instantaneous vector field $V(\mathbf{x};\lambda_j)$: $\mathbf{x}(b) = \mathbf{p} + \int_0^b V(\mathbf{x}(\eta);\lambda_j)d\eta$. The instantaneous topology of $V(\mathbf{x};\lambda)$ at λ_j is then defined as the topological graph of the instantaneous field $V(\mathbf{x};\lambda_j)$. It consists of singularities, *periodic orbits*, and their connectivity [Chen et al. 2007] and conveys the qualitative structure of the vector field at λ_j. This information has been applied to guide the creation and control

of steady vector fields [van Wijk 2002; Zhang et al. 2006; Fisher et al. 2007; Chen et al. 2007]. We have not identified graphics applications in which design and control of periodic orbits is important. Consequently, we will focus on singularities and singularity-related bifurcations in this paper.

Bifurcations: In many graphics applications structural changes of certain graphical properties are often observed, such as the splitting and merging of texture structures in texture synthesis and animation. In some cases, these structural changes may cause visual artifacts. Figure 3 shows an example where the break of texture structures could cause visual discontinuity in the animation. These structural variations are typically associated with the topological changes of the underlying fields, i.e. *bifurcations*, which we review later.

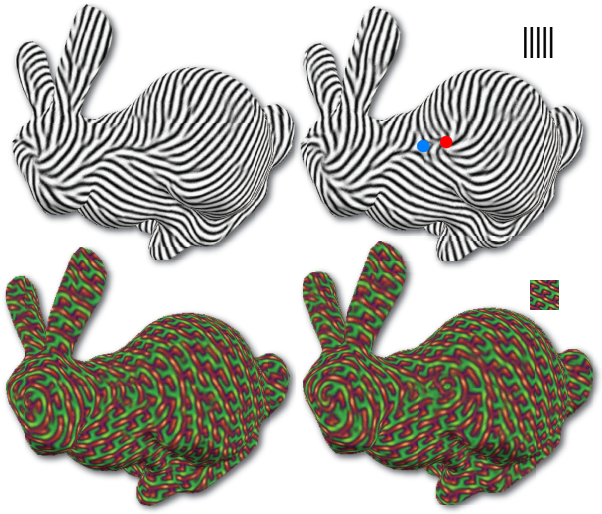


Figure 3: *This figure shows an example of saddle-node bifurcation in an orientation vector field, the creation of a pair of saddle and sink, which causes the break of texture structure on the back of the bunny. Note that we sample the two frames before (left column) and after (right) bifurcation happens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

Once we consider a parameterized vector field, we are able to keep track of the evolution of its instantaneous topology of $V(\mathbf{x};\lambda)$ with respect to the parameter λ by extracting instantaneous topological graph at a discrete step λ_j and matching singularities and other features. This technique has been adapted to analyze time-varying flow datasets [Tricoche et al. 2001]. The movement of a singularity along the λ axis gives rise to a curve which we refer to as a *singularity path*. Singularity characteristics, such as Jacobian property, can also be advected along the path. Therefore, the Jacobian can be computed at any λ given a continuous path and can be used to determine the local field structures near the singularity at λ. Two singularity paths can intersect in domain $M \times \mathbf{R}$ only if they have opposite *poincaré indices*, for instance, the paths of a source and a saddle, or a sink and a saddle, respectively. Note that we have excluded higher-order singularities in our discussion. At the intersections, no hyperbolic singularities exist. Thus, these intersections indicate certain qualitative structural changes (i.e. the number and types of singularities vary, which results in the changes of the topological graph). We refer to these qualitative structural changes as *bifurcations*, and the positions $(\mathbf{x};\lambda)$ where they happen as *bifurcation points*. Figure 4 provides an illustration of such process in a saddle-source bifurcation. More comprehensive introduction of the bifurcation theory can be found in [Hale and Kocak 1991]. Bifurcation points for saddle-node bifurcations can be extracted [Tricoche et al. 2001].

Orientation Field and Advection Field: We have reviewed a number of important concepts that help understand the instantaneous fields and their evolution with respect to the parameter λ. In graphics applications such as texture synthesis and animation, two types of vector fields are required as the input. One is used to orient texture patches, which we refer to as an *orientation field*, while the other describes the advection of texture patches over time, which we call an *advection field*. Figure 5 shows the effect of both types of fields in texture synthesis and animation. The directions of strip-like pattern depict the orientation field, and the cyan arrows in the left image illustrate the advection field (difficult to see with still images). We have specified the orientation field only for the first frame, which is then advected by the advection field using the technique proposed by Kwatra et al. [2007]. Hence, the orientation field over time need not be stationary. This is reflected through the texture movement in the three frames from the animation (Figure 5). Note that the orientation field in this example does not convey any interesting effects on the surface. It is preferrable to control this appearance to achieve meaningful effects, for example highlighting certain features of the shapes. Further, for the movement of the graphical properties (e.g. moving a texture patch from one place to the other), it can be valuable to allow the user to design their animation paths. Therefore, it is necessary to address the design of these two types of fields. The singularity and streamline features are naturally adaptable to the design of time-varying orientation fields in the fashion of parameterized vector field which are concerned with the instantaneous appearance at each discrete parameter value. In addition, they are also capable of producing advection fields. However, we cannot use concepts limited to the spatial domain to describe paths of moving particles. Hence we need *pathlines*.

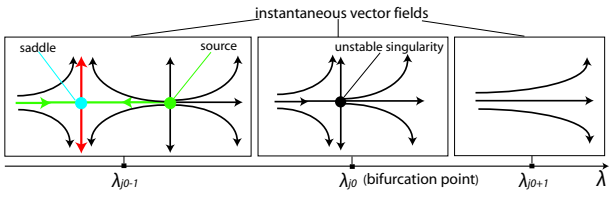


Figure 4: *This example demonstrates a saddle-node bifurcation, i.e. a source-saddle cancellation. The directional curves illustrates the flow behaviors. Two singularities are shown in the left at λ_{j_0-1}. They move towards each other when λ evolves and collide at λ_{j_0} (middle). The two singularities are cancelled after they meet, which results in a singularity-free vector field at λ_{j_0+1} (right).*

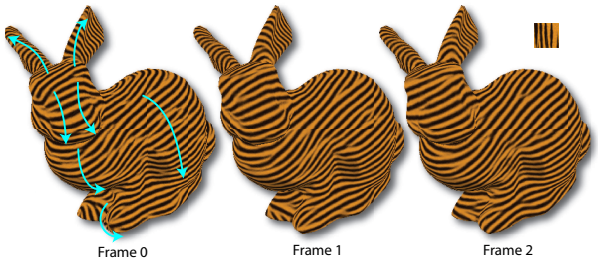


Figure 5: *This example demonstrates the different utility of orientation field and advection field.*

Pathline: A *pathline* passing through $(\mathbf{p};\lambda_j) \in M \times R$ is defined as $\mathbf{x}(b) = \mathbf{p} + \int_0^b V(\mathbf{x}(\eta);\lambda_j+\eta)d\eta$. Note that to define a pathline, we have assumed $\lambda = t$. In contrast to the streamlines passing through $\mathbf{p}$ at any λ, the pathline of $\mathbf{p}$ reflect the true path of the particle starting from $\mathbf{p}$ moving forward along λ. [Theisel et al. 2005] provides an example illustrating the difference between the streamlines and pathlines of the same time-varying vector field. Comparison and definitions of these two concepts shows that streamlines depict the instantaneous appearance of a time-varying vector field, while pathlines convey the real paths of the particles. With the aid of pathlines, we can discuss the design of an advection field much more easily

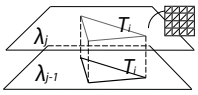
because we can control the moving path of specific particles. Note that singularity paths and pathlines are different.

4 Initialization

In this section, we discuss two approaches that we employ to generate a parameterized vector field. First, instantaneous vector fields are created in key frames and propagated to the rest of the field. The advantage of this approach is that we can reuse past techniques in designing instantaneous (steady) vector fields [Zhang et al. 2006; Fisher et al. 2007; Chen et al. 2007]. However, this approach does not address features unique to time-varying vector fields such as bifurcations and pathlines. In our second approach, we allow such features to be generated through the extended basis vector fields or constrained optimization in a *spatio-parameterized* domain.

4.1 Setting

Computation Domain: Given the definition of a parameterized vector field on a 2-manifold, we define the spatio-parameterized domain as $\mathbf{D} = M \times \mathbf{R}$. In our implementation, we are concerned with a sub-domain $\mathbf{X} \subset \mathbf{D}$ such that $\mathbf{X} = (X; \lambda)$ where X is a triangulation of a 3D surface, and $\lambda \in [0,1]$ is a parameter that we use to approximate time. For representing and storing the field, we discretize λ evenly. We denote these discretely sampled λ values as $\{\lambda_j\}$. A typical number for the discretization is 100 for the examples in this paper. We then compute and store the instantaneous fields at these discrete $\{\lambda_j\}$ in order. The figure to the right shows such a configuration.



Interpolation scheme: We resort to the interpolation scheme that has been successfully applied by Zhang et al. [2006], Palacios and Zhang [2007], and Chen et al. [2007] for continuous surface flow construction in the spatial subspace. Over $\mathbf{X}$, we employ a similar interpolation technique proposed by Tricoche et al. [2001] to guarantee a linear field along the parameter λ dimension. Particularly, in planar case this configuration can be formulated as follows.

$$V(\mathbf{x};\lambda) = \mathbf{a}(\lambda)x + \mathbf{b}(\lambda)y + \mathbf{c}(\lambda) \quad (\mathbf{x};\ \lambda) \in X \subset \mathbf{D}$$

where $\mathbf{a} = (a_x, a_y)$ and $\mathbf{a}(\lambda) = \frac{\lambda_{j+1}-\lambda}{\lambda_{j+1}-\lambda_j}\mathbf{a}_{\lambda_j} + \frac{\lambda-\lambda_j}{\lambda_{j+1}-\lambda_j}\mathbf{a}_{\lambda_{j+1}}$, where $\mathbf{a}_{\lambda_j}$ and $\mathbf{a}_{\lambda_{j+1}}$ are the coefficient of the linearaztion of the vector field at λ_j and λ_{j+1}, respectively. $\mathbf{b}(\lambda)$ and $\mathbf{c}(\lambda)$ are similarly defined.

4.2 Designing Instantaneous Fields

Basis Fields: To design an instantaneous field, the user can either specify the singular or regular design elements (locally) at desired locations [van Wijk 2002; Zhang et al. 2006] or provide a set of streamlines indicating the flow directions along the streamline and in nearby regions. The provided streamlines are eventually sampled and converted into polylines which are used to construct the regular elements. A regular element is an arrow pointing from a basis point to a certain direction. This idea has been employed by Chen et al. [2008] to design street networks that follow the boundaries of natural features such as rivers. Each design element is associated with a Jacobian J_i, and gives rise to a basis field which we use to compute a weighted sum to obtain the global field. Equation 1 defines such a weighted sum.

$$V(\mathbf{x}) = \sum_i e^{-d\|\mathbf{x}-\mathbf{p}_i\|^2} J_i \tag{1}$$

where d is a decay constant, $\mathbf{x}$ is a point in space, J_i is the Jacobian corresponding to a design element, and $\mathbf{p_i}$ is the position of the design element in space.

Further, a brush streamline interface inspired by the brush interface by Chen et al. [2008] for tensor field design is used for instantaneous vector field design. More specifically, the user sketches a curve. A local region with the curve as the skeleton is found [Sethian 1996]. The vector field inside the region is computed according to the derived regular elements from the curve. As pointed out by Chen et al., the brush interface can easily introduce large variations along boundaries of the brush region which may be interesting to segmented texture synthesis. Figure 6 shows three examples using brush streamlines.

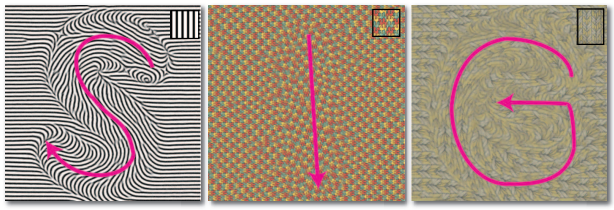


Figure 6: *This figure provides some results of brush stroke design. They are the sampled frames from the accompany animations.*

The radial basis field approach (equation 1) cannot be applied to the design on surfaces without a global parameterization of the surface. Consequently, we resort to the *constrained optimization*, or *relaxation* to design a surface field.

Constrained Optimization: Given a region N of a triangular mesh where the vector values at the boundary vertices of N are the constraints, the constrained optimization possesses the following form:

$$\overline{V}(v_i) = \sum_{j \in J} \omega_{ij} \overline{V}(v_j) \tag{2}$$

where v_i is an interior vertex, v_j's are the adjacent vertices that are either in the interior or on the boundary of N. $\overline{V}(v)$ is the average vector value at vertex v. The weights ω_{ij}'s are determined using Floater's mean-value coordinates [Floater 2003]. Equation 2 is a sparse linear system, which can be solved by using a bi-conjugate gradient method [Press et al. 1992]. Note that constrained optimization can be used to generate a local field for brush streamline design as well by considering the center curve as the boundary constraint.

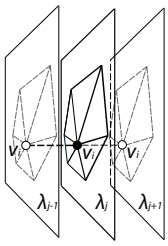
4.3 Designing Parameterized Vector Fields

We now describe the techniques that are used to produce a parameterized vector field. A natural idea is to set the designed instantaneous fields as *key frames* and derive a parameterized vector field from them.

4.3.1 Key Frame Design

In field design using key frames, the user first designs a sparse set of instantaneous fields that indicate the desired effects at specific frames (λ 's). The system then generates a parameterized vector field achieving these effects using an *extended constrained optimization*.

Extended Constrained Optimization: By taking into account the additional parameter λ, we introduce an extended constrained optimization technique to create parameterized vector fields on surfaces. Given a vertex $(v_i;\lambda_j)$ in the underlying mesh in domain $\mathbf{X}$, we consider a stencil of it shown in the figure to the right. In this stencil configuration, we assume there are (virtual) edges between $(v_i;\lambda_j)$ and $(v_i;\lambda_{j-1})$, and $(v_i;\lambda_j)$ and $(v_i;\lambda_{j+1})$, respectively. Therefore, the computation of discrete Laplacian



under this setting needs to consider $(v_i;\lambda_{j-1})$ and $(v_i;\lambda_{j+1})$ as the direct neighbors of $(v_i;\lambda_j)$. We then extend the spatial discrete Laplace as follows:

$$\omega\overline{V}(v_i;\lambda_j) = \sum_{l\in N(i)}\omega_{j,l}\overline{V}(v_l;\lambda_j) + \omega_{j,j-1}\overline{V}(v_i;\lambda_{j-1}) + \omega_{j,j+1}\overline{V}(v_i;\lambda_{j+1}) \quad (3)$$

where $N(i)$ denotes the one-ring neighbors of $(v_i;\lambda_j)$, $\overline{V}(v_i;\lambda_j)$ represents the average vector value at position $(v_i;\lambda_j)$. $\omega_{j,j-1}$ and $\omega_{j,j+1}$ are positive weights determining how fast the relaxation process is. In our implementation $\omega_{j,j-1}$=$\omega_{j,j+1}=10$. $\omega = \sum_{l\in N(i)}\omega_{j,l}+\omega_{j,j-1}+\omega_{j,j-1}$ is the normalization coefficient. We point out that this formula can be further extended by taking into account more sampled steps along λ axis to achieve smoother results as bi-Laplace smoothing does in time-independent case [Fisher et al. 2007].

Figure 7 shows a planar field generated using key frame design and the extended constrained optimization. Many surface fields (Figures 1, 3, 11, and 12) shown in this paper are also generated using this method.

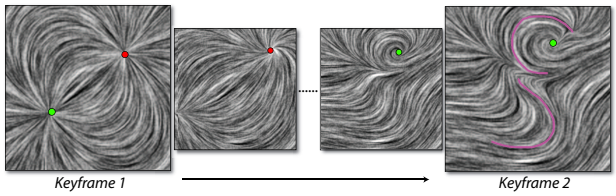


Figure 7: *An key frame design example. The purple curves are the user specified streamlines. The flow-like textures shown in the paper are generated using IBFV(S) techniques of van Wijk [2002; 2003].*

We point out that other interpolation scheme can be employed to obtain a parameterized vector field from a set of instantaneous field as well, such as vector linear interpolation. This typically does not produce smooth results due to the potential degenerate vectors and discontinuity.

For planar field design, an *extended basis field approach* can also be applied to generate a parameterized vector field from the user specifications (singularities and streamlines) directly.

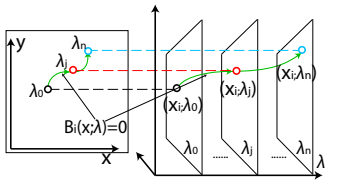
Extended Basis Field Approach: Similar to the instantaneous field design, we have the concepts of design elements with the additional parameter λ being considered. That is, the position of a singular or regular element in the computation domain $\mathbf{D}$ has the form of $(\mathbf{p};\lambda)$. For instance, if the user inserts a singularity in the spatial position $\mathbf{p_i}$ at λ_i, its position in $\mathbf{D}$ is $(\mathbf{p_i};\lambda_i)$. Similarly, all the regular elements stemming from a streamline defined at λ_i will be assigned this parameter value λ_i. Accordingly, the generated basis field will affect not only a single instantaneous field, but also the parameterized vector fields with the instantaneous field at λ_i as center. We update our basis field summation equation as follows:

$$V_I(\mathbf{x};\lambda) = \sum_i e^{-b\|\lambda-\lambda_i\|^2} e^{-d\|\mathbf{x}-\mathbf{x}_i\|^2} J_i \quad (4)$$

where b is a decay constant along λ axis, the ratio of b/d reflects the ratio of the propagation speeds over the spatial and parameter spaces.

4.3.2 Singularity Path Design

In many cases, we want to animate the moving of certain singular patterns over surfaces, for instance, the moving of a storm system in environment modeling. Our system supports this by allowing the user to specify the paths of the singular elements along positive λ direction. To do so, the user first specifies the path of the singular element on the surface in the spatial domain. The parameter information associated with the path is then provided by the user, including the parameter value λ_s at the start point and the value λ_e at the end point, or the parameter values corresponding to the discrete sample points along the paths if provided (see the hollow circles in the figure to the right). We denote the singularity path of i^{th} singular element as $B_i(\mathbf{p}_i;\ \lambda) = 0$ $(\lambda \in [\lambda_s,\lambda_e])$. That is, given $\lambda_j \in [\lambda_s,\lambda_e])$, the position of this singular element $(\mathbf{p}_{ij};\lambda_j)$ satisfies $B_i(\mathbf{p}_{ij};\ \lambda_j) = 0$. The system then induces a parameterized vector field by computing the positions of these singular elements on their paths at each sampled parameter value and summing up all the basis fields as follows.



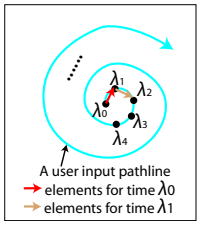
$$V_{S_p}(\mathbf{x};\ \lambda) = \sum_i V_i(\mathbf{p}_i;\ \lambda)|_{B_i(\mathbf{p}_i;\ \lambda)=0} \quad (5)$$

where V_{S_p} denotes the basis fields generated by the singularities that are currently at λ. $V_i(\mathbf{p}_i;\lambda) = e^{-d\|\mathbf{x}-\mathbf{p_i}\|^2} J_i$ is the basis field stemming from the i^{th} singularity at λ and $\lambda \in [\lambda_s,\lambda_e]$). The figure above provides an illustrative example of singularity path design.

Singularity path design on surfaces is handled differently compared to the design on plane. Due to the lack of a global parameterization, we resort to the constrained optimization to generate the individual instantaneous fields at the desired sampled λ_j.

4.3.3 Pathline Design

It is often necessary to specify trajectories of a particle according to a time-varying vector field. The trajectory, a pathline, can be designed using our system as follows. Note that a pathline is different from a singularity path despite the similar design mechanisms in our system for both. To create a field from a pathline, the user first specifies a curve to infer the desired pathline. The system then induces a parameterized vector field based on the sampled positions along the pathline. Our technique is based on the following observation. Given a pathline P and a point $\mathbf{p}_j$ on it at time λ_j, it was advected from previous position on P, $\mathbf{p}_{j-1}$. If $\mathbf{p}_{j-1}$ and $\mathbf{p}_j$ are sufficiently close, the vector pointing from $\mathbf{p}_{j-1}$ to $\mathbf{p}_j$ approximates the true vector value at $(\mathbf{p}_{j-1};\lambda_{j-1})$. We then can set this vector as a regular element at λ_{j-1}. In our implementation, we offer the user a similar interface to the streamline design for sketching the desired pathline. But during the discretization process, each sampled point will be assigned a unique λ value according to the parameter information associated with the corresponding pathlines. Assume the start value λ_s and end value λ_e are known. Then, for the i^{th} sampled point (out of n samples), its assigned λ value is computed as $\lambda_s + i*(\lambda_e - \lambda_s)/(n-1)$. Figure to the right provides an illustrative example of pathline based design. Note that the i^{th} regular element stemmed from line segment $(i,i+1)$ is located at λ_i. The extended basis field approach can be adapted to induce a parameterized vector field for the planar case (equation 4), while a constrained optimization process is needed for the generation of the individual instantaneous fields on surfaces.



Matrix-based Design is also provided by our system for the user to determine how the vector field transforms (rotates, scales, stretches) over λ. The matrix (field) has the form of $M(\lambda) = \begin{pmatrix} M_{11}(\lambda) & M_{12}(\lambda) \\ M_{21}(\lambda) & M_{22}(\lambda) \end{pmatrix}$, where

$M_{ij}(\lambda)$ are functions of λ. In our implementation, we require the user to provide an affine transformation matrix (i.e. the combination of 2D rotation and scaling) to act on the initial instantaneous field (at λ_s) to obtain the last field (at λ_e). The system then interpolates the rotation (rotation angles) and scaling (scaling factors). The vector field V at λ_j is computed as $V(\lambda_j) = M(\lambda_j)V(\lambda_{j-1})$, where $M(\lambda_j)$ is the combination of the rotation and scaling matrices at time λ_j. Figure to the right provides the texture synthesis results guided by an orientation field generated using matrix-based approach where the field is rotated *w.r.t.* λ.

4.3.4 Bifurcation Design

As a significant and novel contribution, our system allows the user to insert a bifurcation at specific location in the spatio-parameterized domain. This is particually useful for the applications where the split or merge of the graphical primitives are desired. Figures 1 and 11 provide examples of desirable bifurcations in the texture synthesis and animation. Recall that we are only concerned with saddle-node bifurcations in this paper. Equation 6 provides a formula we are using to create a saddle-node bifurcation (saddle and node pair creation) at position $(x,y;0)$ in X (see Figure 8). Similarly, we can enforce a saddle and source pair cancellation at position $(x,y;0)$ in X using equation 7. In addition, we can scale the range of the bifurcation in both space and time, and re-orient the axis (a straight line in this case) to control where and how the bifurcation happens.

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \lambda - x^2 \\ -y \end{pmatrix} \tag{6}$$

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \lambda + x^2 \\ y \end{pmatrix} \tag{7}$$

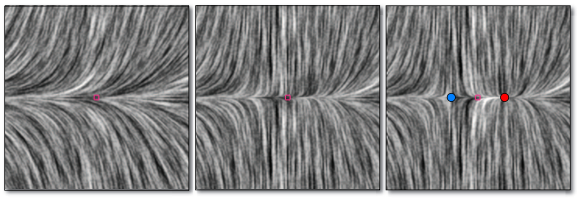


Figure 8: *A saddle sink creation bifurcation happens at* $(0.5,0.5;0.5)$ *in the spatio-parameterized domain* X.

Other types of bifurcations [1991] can be created in the similar manner. The global field induced from a set of bifurcations can be computed as the weighted sum of individual bifurcations (equation 8). In other words, each bifurcation is a design element.

$$V_B(\mathbf{x};\,\lambda) = \sum_i e^{-d\|\mathbf{x}-\mathbf{x_i}\|^2} V_i(\mathbf{x};\,\lambda-\lambda_i) \tag{8}$$

where $(\mathbf{x_i};\lambda_i)$ is the position at which the i^{th} bifurcation happens. Note that the decay along λ is considered by the formula $V_i(\mathbf{x};\,\lambda-\lambda_i)$ (see equations 6 and 7). Accordingly, we define the global field as the weighted sum of the basis fields generated using singular and regular elements V_I (equation 4), and bifurcations V_B (equation 8).

$$V(\mathbf{x};\,\lambda) = \omega_B V_B(\mathbf{x};\,\lambda) + \omega_I V_I(\mathbf{x};\,\lambda) \tag{9}$$

where ω_I and ω_b are positive values which we are using 0.5 in our implementation.

There are two alternative approaches of inserting bifurcations into the designed field. First, the user can generate a bifurcation through key frame design. Basically, the user sets two instantaneous fields as key frames before and after the λ value where the bifurcation point is desired. One of these fields is trivial (similar to the left field shown in Figure 8), while the other contains the saddle and source (or sink) singularity pair (the right field shown in Figure 8). Then, a bifurcation is enforced to occur by the extended constrained optimization. The second approach allows the user to design the split of a singularity path to indicate a saddle-node creation bifurcation or intersect the end points of two singularity paths to induce a saddle-node cancellation bifurcation. Both approaches were applied to insert bifurcations into the designed fields (Figures 1and 11).

5 Editing

Editing functionality is required for a design system because of the appearance of undesired features such as singularities and bifurcations in the initialization phase. Our system provides the user with a number of options to edit a given parameterized vector field. First, the conventional editing operations for instantaneous vector field modification are provided. Second, the novel bifurcation removal and movement are introduced along with a general smoothing scheme in the spatio-parameterized domain.

5.1 Instantaneous Field Editing

First, the system extracts the instantaneous vector field topology. Then, the user can cancel two unwanted singularities at a particular λ value using the simplification techniques proposed by Chen et al. [2007]. 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 parameter axis. We relieve this by adding the constraint of maximum propagation distance along λ.

5.2 Bifurcation Editing

We have demonstrated the relations between saddle-node bifurcations and the structural changes in texture animations. We now develop techniques to control them. To do so, we need to first know where the bifurcations occur. In our implementation, we keep track of singularities and extract bifurcations from the designed fields using the techniques proposed by Tricoche et al. [2001].

Bifurcation Removal: We allow the user to remove a bifurcation if the singularities involved do not participate in other bifurcations. We refer to these bifurcations as *isolated* bifurcations. If a bifurcation is not isolated, we can not cancel it without affecting other features. To remove an isolated bifurcation, we keep track of the involving singularities along the λ axis until their birth (saddle-node cancellation) or their death (saddle-node creation). We assume the λ value of their birth (or death) is λ_c. Then, cancelling these singularities at λ_c will induce the removal of the corresponding bifurcation. Under our setting of $\mathbf{X}$, only boundary singularities will satisfy this requirement. Figure 9 shows an example of saddle-node bifurcation removal. More complex local control of connected bifurcations is possible which is beyond the scope of this paper. Note that this operation is not valid in a real time-dependent vector field where the range of the physical time is infinite. In that setting, the more meaningful operation is *bifurcation movement*.

Bifurcation Movement: Similar to the singularity movement functionality, a bifurcation can be moved. Moving bifurcation can be achieved by moving the involving singularities over space at particular λ value. The edited instantaneous field is then set as a key frame. The extended constrained optimization will smooth the

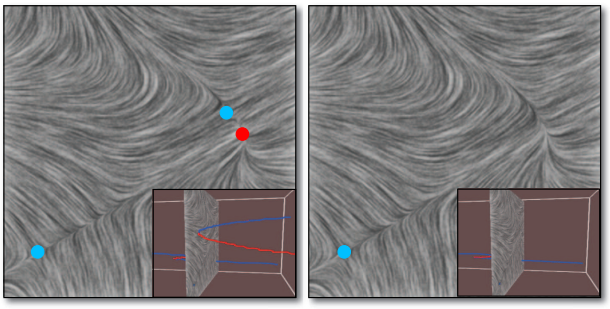


Figure 9: *Example of bifurcation editing. Left column shows the effect before editing; right column shows the results after bifurcation removal.*

rest of the field. Note that the movement of these two singularities should obey the topological constraints proposed by Zhang et al. [2006] in their steady vector field design tool. This guarantees no other topological features will be affected during the movement.

General Smoothing: The two aforementioned bifurcation control techniques are typically too constrained for the design fields. We then introduce a more relaxed editing functionality to allow the user to modify the designed fields without concerning with the topological constraints. We refer to this technqiue as *general smoothing*.

General Smoothing is a spatio-parameterized smoothing in which the user defines a box in $\mathbf{X}$. The vector values at the inner vertices of the box are replaced with a hopefully smoother version computed using the extended constrained optimization (equation 3), with the boundary vertices as the constraints.

6 Application: Texture Synthesis and Animation

We have applied the designed parameterized orientation fields and advection fields generated using our techniques to create a number of synthetic texture animations (Figures 1, 3, 5, 6, 12, and 11). Flow-guided texture synthesis and advection has been introduced to visualization community for dense flow visualization by van Wijk [2002; 2003], Laramee et al. [2003], and Neyret [2003]. Kwatra et al. [2005] present an optimization-based plane texture synthesis which can be used for flow-guided texture animation. Lefebvre and Hoppe's [2006] introduce an appearance-space texture synthesis technique that can handle texture advection over static surfaces. Later, Kwatra et al. [2007] and Bargteil et al. [2006] extend the advected texturing techniques onto the problem of fluid texturing on surfaces, respectively. In addition, Wiebe and Houston [2004] and Rasmussen et al. [2004] perform fluid texturing by advecting texture coordinates along the flow field using level sets and particles. In this paper, we employ Kwatra et al.'s [2007] texturing fluid techniques for our texture synthesis and animation examples.

Many applications may want the animated texture on surfaces, such as special effects, games, and digital arts. In addition, with the proper choice of texture exemplars and careful field design, other graphical effects can be resembled through advected textures, such as the ripple-like advection, the time-varying caustic reflection and the lava effect (Figure 10).

Performance: The intialization of a planar field with 100 frames defined on a 65×65 regular grid typically takes less than 5 seconds on a 3 GHz PC with 1GB RAM. For the design on surface (up to $20,000$ vertices), it can take up to 4 minutes to generate the field with 100 frames without optimization. The times spent on synthesis vary from 8 hours to 20 hours on a 3.6 GHz PC with 2GB RAM depending on the number of sample points being put on the surfaces, the volume sizes of the models, and the sizes of the input texture exemplars.

7 Conclusion and Future Work

This paper addresses the problem of the design of time-varying vector field on surfaces. We propose the use of the parameterized vector fields to approximate the solution. Two different types of vector fields are discussed for different purposes in texture synthesis and animation. Various design techniques are introduced to address the design of these two types of fields efficiently. The initial fields can be further edited to eliminate undesired effects. To our knowledge, the presented design framework is the first in its kind for general time-varying vector field design with bifurcation control.

We do not currently provide an explicit solution to control the in between field generation. Therefore, unexpected behaviors may arise which requires either post processing or re-configurating the initial setting and re-produce it again. This regeneration process is expensive compared to steady field design and does not guarantee more satisfiable solutions can be found. In the future, we expect a more robust technique that can inform the user what could be possibly obtained given the specified constraints to address this problem. Second, we have only focused on a small set of vector field features for the creation of a parameterized vector field. There are other design primitives that may be important such as streaklines and timelines. Third, the bifurcation control techniques proposed in this work are still limited due to the lack of the support of a systematic time-varying dynamics theory. Fourth, it will be interesting to experiment other vector field generation methods such as the one based on discrete calculus (DEC) [Fisher et al. 2007].

This work opens a new range of the field design topic which can be extended to the higher dimensional field design, such as time-varying tensor field design.

Acknowledgments

This work was supported by NSF CCF-0546881 and CCF-0830808.

References

ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LÉVY, B., AND DESBRUN, M. 2003. Anisotropic polygonal remeshing. *ACM Trans. Graph. 22*, 3, 485–493.

BARGTEIL, A. W., SIN, F., MICHAELS, J. E., GOKTEKIN, T. G., AND O'BRIEN, J. F. 2006. A texture synthesis method for liquid animations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

CHEN, G., MISCHAIKOW, K., LARAMEE, R. S., PILARCZYK, P., AND ZHANG, E. 2007. Vector field editing and periodic orbit extraction using morse decomposition. *IEEE Transactions on Visualization and Computer Graphics 13*, 4, 769–785.

CHEN, G., ESCH, G., WONKA, P., MUELLER, P., AND ZHANG, E. 2008. Interactive procedural street modeling. *ACM Trans. Graph. 27*, 3, 103:1–103:10.

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

FISHER, M., SCHRÖDER, P., DESBRUN, M., AND HOPPE, H. 2007. Design of tangent vector fields. *ACM Trans. Graph.*, 56:1–56:8.

FLOATER, M. S. 2003. Mean value coordinates. *Comput. Aided Geom. Des. 20*, 1, 19–27.

FU, H., WEI, Y., TAI, C.-L., AND QUAN, L. 2007. Sketching hairstyles. In *SBIM '07: Proceedings of the 4th Eurographics*

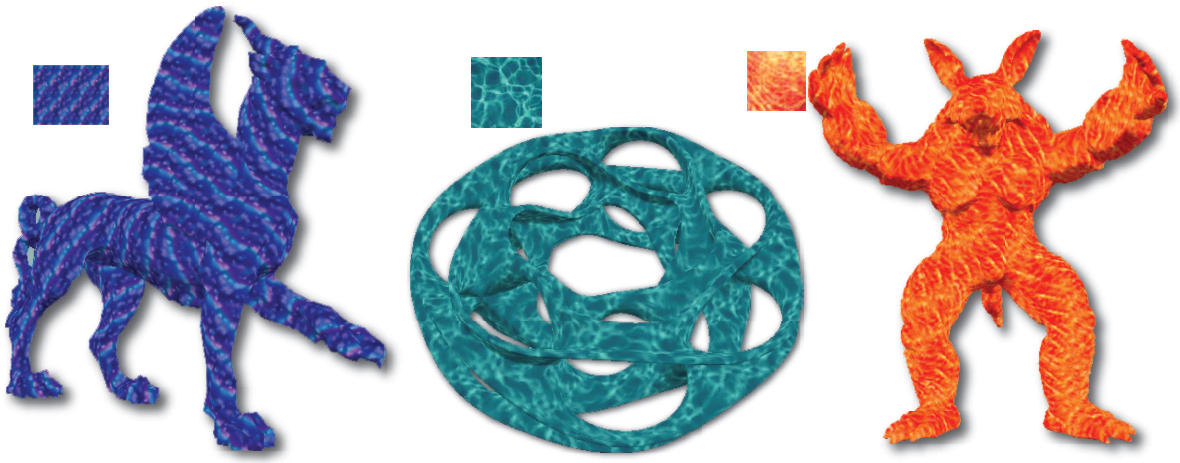


Figure 10: *Different effects obtained using texture synthesis and animations: ripple advection (left), caustic reflection (middle), and the lava effect (right). All the texture synthesis and animations are driven by the time-varying vector fields created using our system.*

workshop on Sketch-based interfaces and modeling, ACM, New York, NY, USA, 31–36.

HALE, J., AND KOCAK, H. 1991. *Dynamics and Bifurcations*. New York: Springer-Verlag.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 517–526.

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

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. *ACM Transactions on Graphics, SIGGRAPH 2005* (August).

KWATRA, V., ADALSTEINSSON, D., KIM, T., KWATRA, N., CARLSON, M., AND LIN, M. 2007. Texturing fluids. *IEEE Transactions on Visualization and Computer Graphics 13*, 5, 939–952.

LARAMEE, R. S., JOBARD, B., AND HAUSER, H. 2003. Image space based visualization of unsteady flow on surfaces. In *Proceedings IEEE Visualization '03*, IEEE Computer Society, 131–138.

LEFEBVRE, S., AND HOPPE, H. 2006. Appearance-space texture synthesis. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, ACM, New York, NY, USA, 541–548.

NEYRET, F. 2003. Advected textures. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 147–153.

PALACIOS, J., AND ZHANG, E. 2007. Rotational symmetry field design on surfaces. *ACM Trans. Graph. 26*, 3, 55:1–55:10.

PRAUN, E., ADAM, F., AND HUGUES, H. 2000. Lapped textures. In *Proceedings of ACM SIGGRAPH 2000*, 465–470.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press.

RASMUSSEN, N., ENRIGHT, D., NGUYEN, D., MARINO, S., SUMNER, N., GEIGER, W., HOON, S., AND FEDKIW, R. 2004. Directable photorealistic liquids. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 193–202.

RAY, N., LI, W. C., LVY, B., AND AN D PIERRE ALLIEZ, A. S. 2006. Periodic global parameterization. *ACM Transactions on Graphics*.

RAY, N., VALLET, B., LI, W.-C., AND LEVY, B. 2008. N-symmetry direction field design. *ACM Trans. Graph. 27*, 2, 10:1–10:13.

SETHIAN, J. 1996. A fast marching level set method for monotonically advancing fronts. In *Proc. Nat. Acad. Sci.*, vol. 93, 1591–1595.

STAM, J. 1999. Stable fluids. In *Proceedings of ACM SIGGRAPH 1999*, Addison Wesley Longman, Los Angeles, A. Rockwood, Ed., 121–128.

STAM, J. 2003. Flows on surfaces of arbitrary topology. In *ACM Transactions on Graphics (SIGGRAPH 03)*, vol. 22, 724–731.

THEISEL, H., WEINKAUF, T., HEGE, H.-C., AND SEIDEL, H.-P. 2005. 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.

THEISEL, H. 2002. Designing 2d vector fields of arbitrary topology. *Computer Graphics Forum (Proceedings Eurographics 2002) 21* (July), 595–604.

TREUILLE, A., COOPER, S., AND POPOVIĆ, Z. 2006. Continuum crowds. *ACM Trans. Graph. 25*, 3, 1160–1168.

TRICOCHE, X., SCHEUERMANN, G., AND HAGEN, H. 2001. Topology-based visualization of time-dependent 2d vector fields. In *Data Visualization 2001 (Joint Eurographics-IEEE TCVG Symposium on Visualization Proceedings)*, 117–126.

TURK, G. 2001. Texture synthesis on surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 2001)*, 347–354.

VAN WIJK, J. J. 2002. Image based flow visualization. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 745–754.

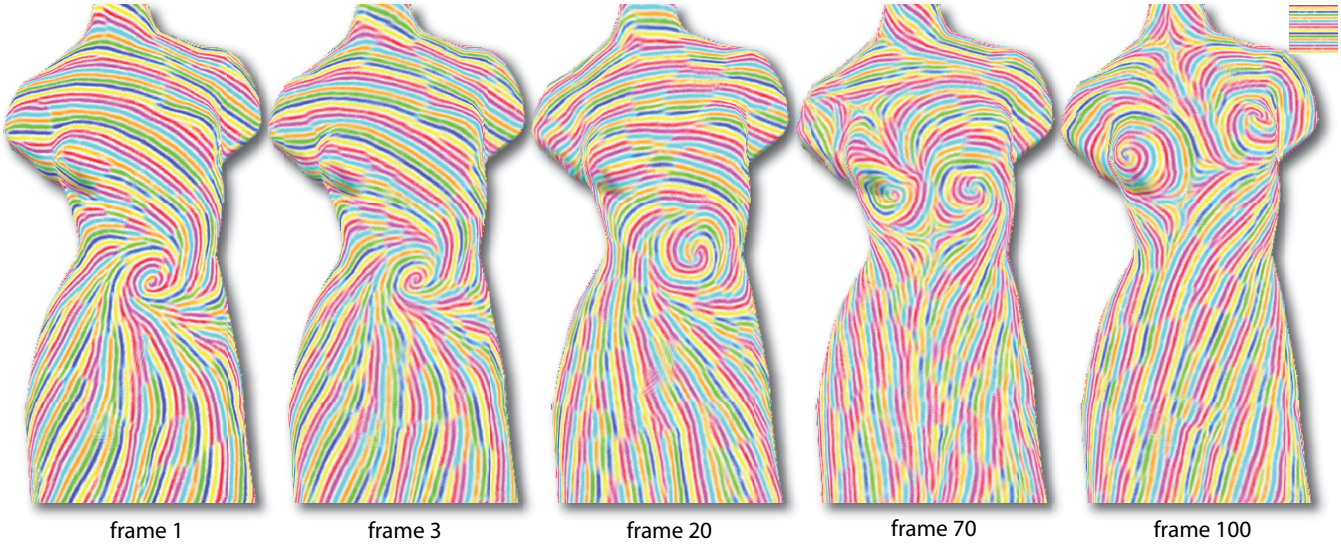


Figure 11: *This image shows a number of frames from a texture animation on venus. The animation is driven by an orientation field and an advection field, both are designed using the techniques introduced in this paper. Note that a vortex-like pattern (frame 1) is displaced by the advection field at the middle of the model (frames 3 and 20). The vortex is then splitted into two (frame 70), and both of them continue moving upwards to the upper middle along the model (frame 100).*

VAN WIJK, J. 2003. Image based flow visualization for curved surfaces. In *Proceedings IEEE Visualization '03*, IEEE Computer Society, 123–130.

VON FUNCK, W., THEISEL, H., AND SEIDEL, H.-P. 2006. Vector field based shape deformations. *ACM Trans. Graph. 25*, 3, 1118–1125.

WEI, L. Y., AND LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 2001)*, 355–360.

WIEBE, M., AND HOUSTON, B. 2004. The tar monster: creating a character with fluid simulation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches*, ACM, New York, NY, USA, 64.

XU, L., CHEN, J., AND JIA, J. 2008. A segmentation based variational model for accurate optical flow estimation. I: 671–684.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2006. Vector field design on surfaces. *ACM Transactions on Graphics 25*, 4, 1294–1326.

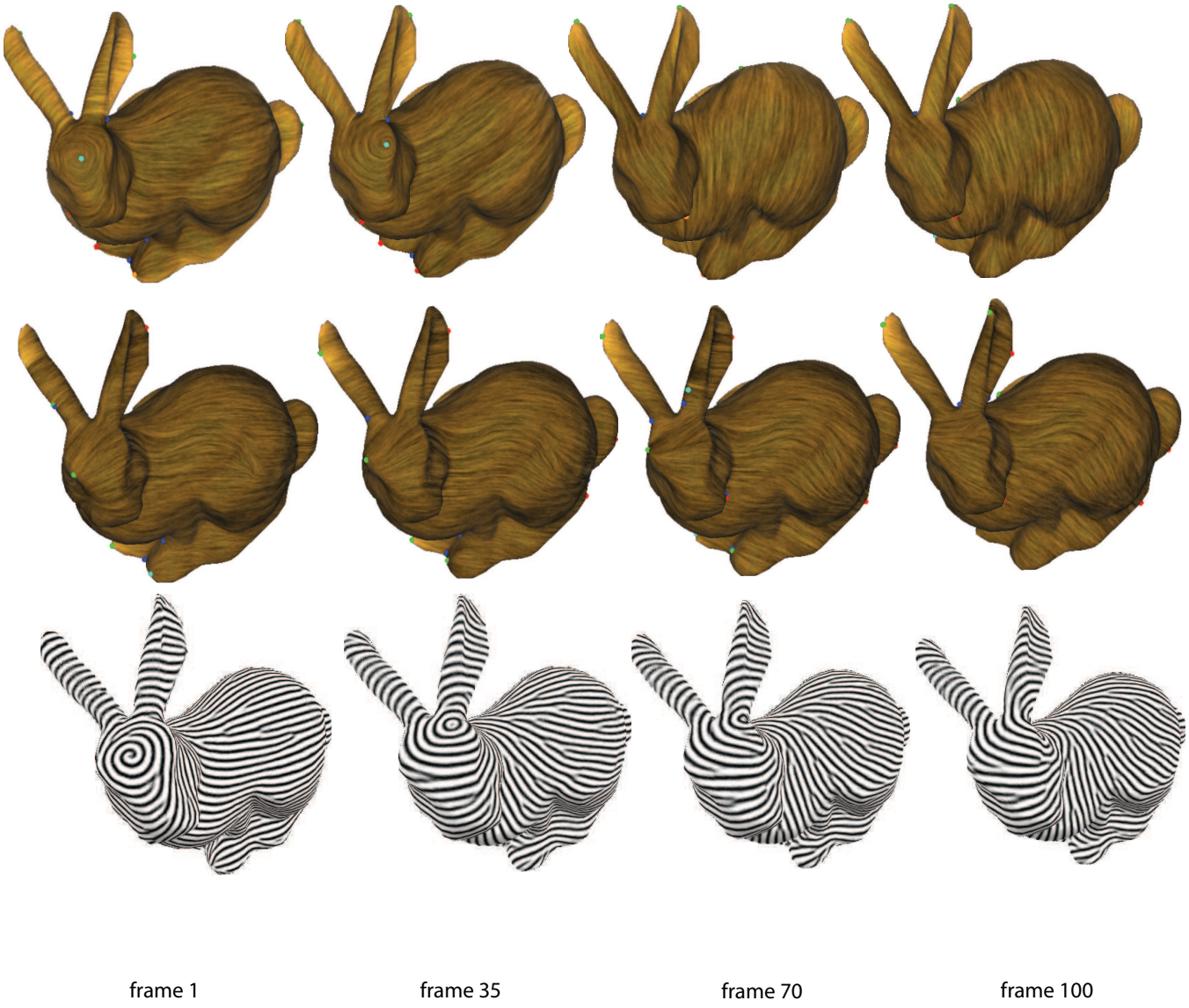


Figure 12: *The designed results of an orientation field (first row) and and advection field (second row) on bunny. The sampled frames from the corresponding texture synthesis and animation results are provided below the fields. Particularly, the third row shows the advection of the orientation field of the first frame.*