Output-Coherent Image-Space LIC for Surface Flow Visualization

Jin Huang[†] Wenjie Pei[†] Chunfeng Wen[†] Guoning Chen[‡] Wei Chen[†*] Hujun Bao[†]

[†]State Key Lab of CAD&CG, Zhejiang University [‡] The University of Utah

ABSTRACT

Image-space line integral convolution (LIC) is a popular approach for visualizing surface vector fields due to its simplicity and high efficiency. To avoid inconsistencies or color blur during the user interactions in the image-space approach, some methods use surface parameterization or 3D volume texture for the effect of smooth transition, which often require expensive computational or memory cost. Furthermore, those methods cannot achieve consistent LIC results in both granularity and color distribution on different scales.

This paper introduces a novel image-space LIC for surface flows that preserves the texture coherence during user interactions. To make the noise textures under different viewpoints coherent, we propose a simple texture mapping technique that is local, robust and effective. Meanwhile, our approach pre-computes a sequence of mipmap noise textures in a coarse-to-fine manner, leading to consistent transition when the model is zoomed. Prior to perform LIC in the image space, the mipmap noise textures are mapped onto each triangle with randomly assigned texture coordinates. Then, a standard image-space LIC based on the projected vector fields is performed to generate the flow texture. The proposed approach is simple and very suitable for GPU acceleration. Our implementation demonstrates consistent and highly efficient LIC visualization on a variety of datasets.

Index Terms: I.3.3 [Computer Graphics]: Picture/Image Generation—Line and Curve Generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1 INTRODUCTION

Visualizing vector fields is of paramount importance in many applications like engineering design, computational fluid dynamics, and climate research. Many approaches for vector field visualization have been studied, among which Line Integral Convolution (LIC) [2] is a popular one. Reasons for that include its superior capability of space-filling, and its high efficiency when implemented on modern graphics hardware. More importantly, LIC can be applied to surfaces to study the vector fields in 3D space. This paper concentrates on the LIC visualization of surface vector fields.

Roughly speaking, existing texture-based visualization approaches for surface vector fields can be classified into two categories: parameterization-dependent methods and image-space methods. The first category generates the LIC textures in a parametric space, which can cover the whole surface and naturally preserve the texture continuity over the surface even under user interactions. However, this either requires a surface parameterization as a priori [5] or an expensive precomputation of the necessary surface parameterization [14]. Furthermore, surface parameterization is greatly affected by the quality of the surface mesh, spacial distortion of the texture may arise. An additional surface patching process is typically required to overcome this issue [14]. This is challenging for large-scale surfaces with complex geometry and topology. Alternatively, an image-space approach generates LIC image for only the visible portion of the surface given a view point. Specifically, it projects the vector fields and surface geometry into the viewing screen, and then applies texture advection [26, 12] or 2D LIC in the image space [15]. This provides higher performance than the parameterization-dependent approaches, and it is easy to implement with the aid of the model graphics hardware. However, this scheme may suffer from artifacts of inconsistency (Figure 1) when the viewer is rotating or zooming the model. In other words, the LIC result is not consistent among consecutive frames during user interactions when the noise texture is generated independently on the image space for each frame, or say, the noise texture on the surface is changing. To address this issue, Weiskopf and Ertl [27] integrate some aspects of object-space methods to achieve the effect of smooth transition during user interactions, and keep the granularity consistent. However, the method requires a 3D solid texture which is memory costly, and cannot get consistent color distribution on different scales (see the comparison in Figure 2 concerning the inconsistency).

In this paper, we present a novel image-space technique for the visualization of surface vector fields which takes into account the texture inconsistency during user interactions. Our approach takes a similar pipeline as the conventional image-space surface LIC visualization, while modifies it with a simple texture mapping technique and an additional pre-process stage. Concerning the first one, we propose to fix the texture coordinates of each vertex with a simple triangle-texture matching technique, making the noise textures under different viewpoints coherent, while the conventional image-space approach cannot warrant this through mapping the whole model to the texture space. This scheme is feasible because convoluting a white noise with a vector field yields a result that is not sensitive to the texture coordinates. In this way, no surface parameterization is required, and the underlying surface model can be arbitrarily complex and large.

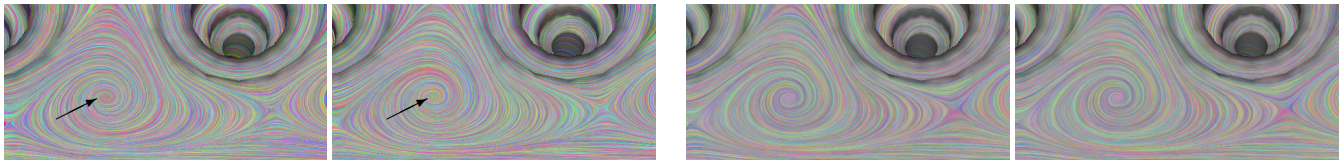
Even with a consistent noise texture, however, the result can still exhibit popping artifacts caused by the texture aliasing. Meanwhile, the LIC streamlines may greatly vary with the zooming operations, which makes the result unclear or flickering. Our solution is to precompute a customized noise texture pyramid that simultaneously characterizes the consistency and variance of the noise texture, and choose an appropriate mipmap level on-the-fly. This scheme ensures consistent LIC results because all noise textures have similar appearances and differ only in the granularity.

We additionally design several enhancement techniques to allow for easy controls on effects like the contrast, the density and the length of streamlines. These techniques are compatible with standard image-space surface LIC algorithm, and can be seamlessly incorporated into the GPU implementation.

In summary, this paper presents an efficient surface LIC visualization approach with the following contributions:

1. A parameterization-free image-space surface LIC generation scheme that works for arbitrarily complex (manifold or non-manifold) and large mesh models with $O(N^2)$ memory complexity (assuming the image is of size N by N);
2. A novel mipmap-based noise texture generation technique

*Corresponding Author:chenwei@cad.zju.edu.cn



(a) (b)

Figure 1: Two consecutive frames during the zooming in operation. Randomly generating the noise in the image space [15] causes popping of color as shown in (a), especially around singularities. Our method generates consistent streamline rendering result (b). Please see the accompany video for a better illustration.

that leads to output coherent, which not only leads to smooth transition of LIC streamlines during interaction, but also consistency in different scales; To further improve the consistency around the silhouette, i.e., the surface boundary, we extended the streamlines to the back face of the surface, which is different to the conventional image-space approach which uses only the visible portion of the surface.

3. A suite of LIC visualization enhancement techniques that is suitable for GPU implementation.

The remainder of this paper is organized as follows. We first review related work in Section 2 and then present our approach in Section 3. Next, we show the results in Section 4. Finally, we summarize this paper and highlight the future work in Section 5.

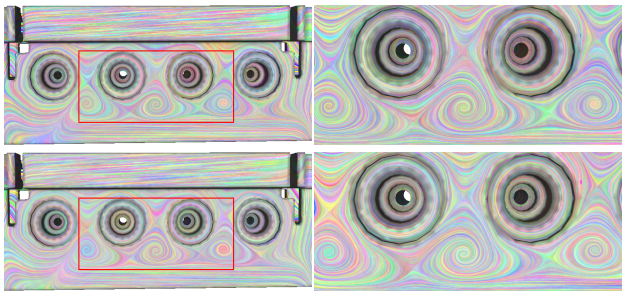


Figure 2: The image on the right column shows the region in the red rectangle on the left image by moving the camera closer to the surface. Our method utilizes a set of correlated noise textures in a mipmap texture pyramid, and makes the results share similar color distribution in different scales (upper row). Using uncorrelated multi-resolution noise textures (lower row) cannot achieve such consistency.

2 Related Work

Reviewing all work on vector field visualization is beyond the scope of this paper. We will cover the most relevant work here (LIC-based methods) and refer readers to [8, 11] for a comprehensive survey.

The first texture-based vector field visualization method [24] distributes a large number of spots on the spatial domain, and generates an illustrative texture, called noise-based texture. Later, Cabral and Leedom [2] proposed to locally smooth an input noise texture by convoluting the texture with a filter kernel derived from the vector fields. The so-called LIC technique, leads to a high correlation along streamlines, and generates a dense texture representation for vector fields. Simply speaking, both the noise-based and the LIC based schemes employ a space-filling scheme, and are amenable for parallelization.

Subsequently, the LIC-based vector field visualization approach has been improved in various aspects [16], [6], [21], and been extended to surface [5], unsteady [22], and 3D flows [4], respectively. The efficiency of LIC is also improved by a fast LIC approach [23], of which the speed acceleration is gained from the minimization of the total number of streamlines.

In particular, Forssell and Cohen [5] first applied the LIC-based visualization approaches to parametric surfaces by transforming the vector fields into the parameter space, and generating LIC in this parameter space. In the last step, the LIC result is mapped back to the surface. The main problem of this scheme lies in that it is often difficult to get a global parameterization with low distortion, which is almost impossible for non-manifold or even curved (i.e. non-flat) surface models. Alternatively, the technique proposed by Battke et al. [1] tessellates a surface with triangles and packs triangles into texture memory. A local LIC texture is computed for each triangle based on its local Euclidean coordinates, avoiding the global parameterization. Its main drawback is that triangle packing demands the model to have a good mesh quality. It should be noticed that our approach basically takes a local parameterization means. The difference from the parameterization-dependent methods [14] lies in that our approach does not fulfill the LIC in the parameterization domain, and thus removes the requirement of low distortion parameterization.

The image based flow visualization approach (IBFV) [25] is regarded as the fastest algorithm for texture-based visualization of 2D unsteady vector fields. Based on this technique, two dense texture-based methods are proposed for visualizing unsteady vector fields on surfaces: IBFVS (Image Based Flow Visualization for Curved Surfaces) [26] and ISA (Image-Space Advection) [12]. Both approaches mentioned in [13] compute the LIC in the image space by leveraging a projection-and-advection pipeline. In this way, no parameterization of surfaces is needed, and the entire procedure can be accelerated using modern graphics hardware. Later, Li et al. [15] applied this image-space scheme to conventional LIC and achieved high-performance vector field visualization. Recently, Zhang et al. [28] presents an interactive visualization technique for planar and surface tensor fields. More recently, Palacios and Zhang [17] advances the image-space LIC visualization approaches to allow for interactive visualization of rotational symmetry fields both in the plane and on surfaces. Image based method has great advantages on efficiency, but suffers from inconsistencies or color blur during the user interaction because the noise texture is generated independently in each frame and has inadequate resolution. Weiskopf and Ertl [27] proposed an excellent method to address these issues: embedding the surface in an object-space 3D solid noise texture leads to consistent LIC result, and blending multiple noise textures in different resolutions makes the granularity consistent. Our method surpasses theirs in less memory requirement and the ability of similar color distribution in different scales, i.e. better consistency.

3 Our Approach

The goal of our work is to enable consistent visualization of vector fields on surfaces, that is, the LIC results share similar color distribution and granularity whenever the surface is rotated, translated or scaled. Throughout this paper, we describe our algorithm in terms of a triangular surface model, while it is easy to make it amenable for other surface representations.

Our approach can be separated into two stages: preprocessing and visualization. The pipeline (Algorithm 1) is highlighted in Figure 3. In the preprocessing stage, we generate a consistent noise texture pyramid and compute the texture coordinates for each triangle, which is the focus of our work to address the inconsistence issue (Section 3.2). The visualization part follows the image-space LIC pipeline, and also is improved to enhance the quality (Section 3.3). In the step of vector field projection, we use a similar scheme to ISA [12]: for each triangle mesh vertex p with vector v, a vertex shader is used to calculate the image-space coordinates of p and $p+v$ as p' and $(p+v)'$ respectively (the details of transforming object-space coordinates to image-space coordinates can be found in the appendix), then evaluate the projected image-space vector by normalizing the vector $(p+v)'-p'$. Given the image-space vector field defined on each triangle mesh vertex, the vector on each pixel can be interpolated in the downstream pixel shaders.

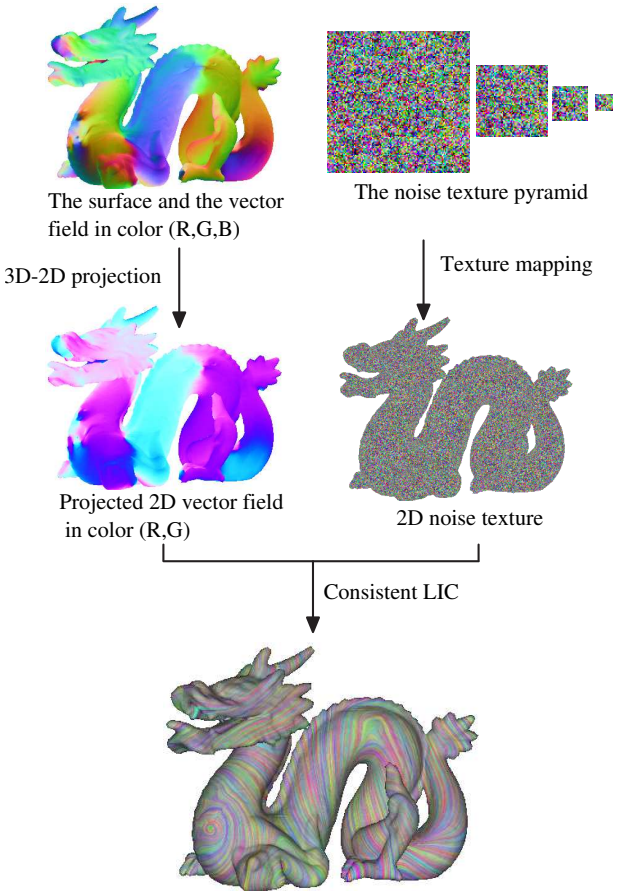


Figure 3: The pipeline of our approach.

3.1 Mapping Noise Texture to the Mesh

In previous image-space surface LIC approaches, the inconsistency appears because the noise texture in the image space is generated independently frame by frame. Our solution for that is to attach the model with a consistent noise texture by means of a simple texture mapping process:

Input : A triangle mesh with vector field on it
Output: Interactive visualization result

preprocessing stage:
Generate a consistent noise texture pyramid.
Map the texture to the mesh.

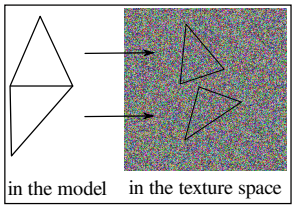
visualization stage:
foreach *frame* **do**
  Project the vector field and noise texture to the image space.
  Perform the LIC in the image space.
end

Algorithm 1: Pseudo code of the pipeline.

1. Generate a random white noise 2D texture map. The size of the texture is determined by the projected size of triangles in the image space. If it is too small, the noise texture will wrap and influence the white noise property. We use a 256×256 texture image in all the results.

2. Generate texture coordinates for each triangle by randomly projecting it into the texture space with the same scale.

   This process can be regarded as a simple local (per triangle) parameterization, and has no requirement on the continuity and distortion of the parameterization. Satisfying results are yielded because the texture image contains only white noise. The texture coordinates of each triangle remains the same during interaction as well as the noise textures. This means that each triangle will always map to the same portion of the texture every frame, regardless of the camera and the vector field, thus guaranteeing the consistent output , which is different from ISA and IBFVS and leads to consistent noise map on the surface.



3. Perform texture mapping with the given texture coordinates and generate texture noise in the image space.

The described technique leads to consistent noise in the image space when the viewpoint is changed. Because no global parameterization is required, it is naturally suitable for complex (manifold or non-manifold) and large-sized models. It should be noticed that our method requires only 2D texture image, and thus poses less requirement to memory $O(N^2)$ than [27], which uses 3D solid texture ($O(N^3)$).

3.2 Generating Consistent Noise Texture Pyramid

Even though the noise is consistent, the LIC result will still present popping artifacts caused by the texture aliasing, especially when the model is zoomed in (please see the accompanying video). In addition, the streamlines will become thicker (Figure 4 (b)) than desired (Figure 4 (c)) when the model is zoomed in.

3.2.1 Automatic mipmap generation

A straightforward solution would be using the mipmap technique (the automatic-generated texture pyramid) which is provided in standard graphics libraries like OpenGL. The texture pyramid is composed of a sequence of noise textures, of which the $n^{th} \in [n_{min}, n_{max}]$ image has the resolution of $2^n \times 2^n$. In this way, the

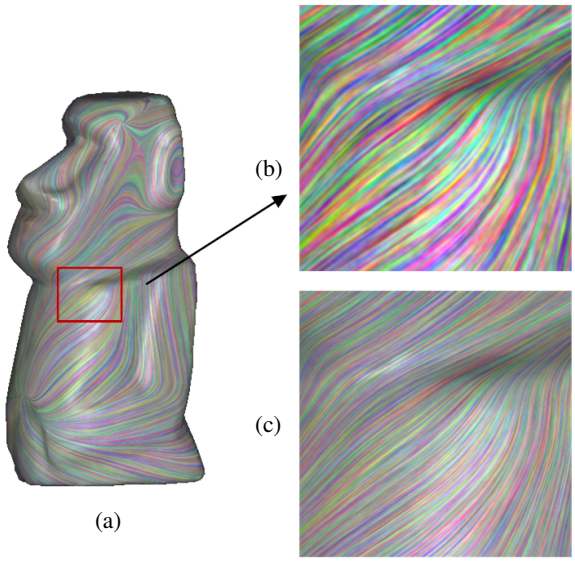


Figure 4: (a) A LIC result for the Moai model. When the model is zoomed in, the result (b) with the noise texture used in (a) presents color blur, while the result (c) with a noise texture selected from the pre-computed texture pyramid presents pleasing effect.

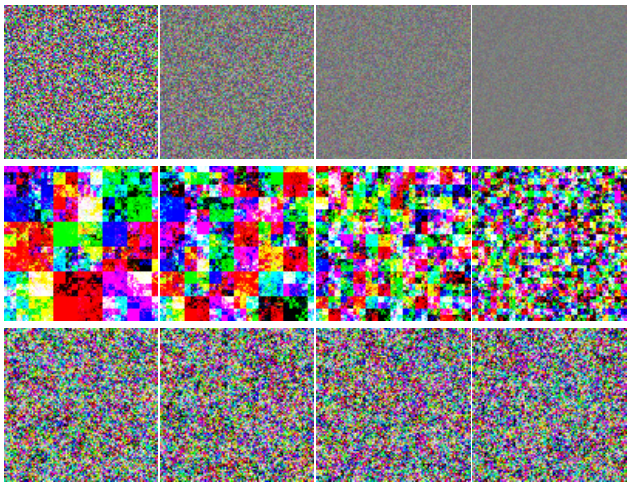


Figure 5: From left to right, the image shows a *small portion* of the texture image in the pyramid in the order of resolution gradually decreasing. The first row shows automatic-generated noise mipmap. As resolution decreases, the maps become blurred. The second row shows the customized maps without the scaling operation in Equation 2. It can be seen that the noise maps have color blocks. The maps in the last row are generated using our algorithm (Equation 3).

popping artifacts can be alleviated during the zooming operation, and the resolution of streamlines is appropriate as long as the zooming factor dose not exceed the finest level in the pyramid. However, this scheme may fail because the contrast of the LIC streamlines decreases greatly when the model is zoomed out. This problem also exists where the projected surface part is largely sheared, yielding over-blurred results (see Figures 5 and 6).

Let us briefly analyze the process of automatic mipmap generation. Both the random colors ([17]) and gray-level noise ([27]) can be used to generate the LIC result. Our approach employs the random colors. When computing the image in the $(n-1)^{th}$ level from the n^{th} level, the value of each RGB channel $I_{n-1}(x,y) \in [0,1]$ at (x,y) in the $(n-1)^{th}$ level is derived from the corresponding four pixels in the n^{th} level:

$$I_{n-1}(x,y) = \frac{\sum_{\Delta x,\Delta y \in \{0,1\}} I_n(2x+\Delta x, 2y+\Delta y)}{4} \tag{1}$$

According to Equation 1, the variance of the $(n-1)^{th}$ level is smaller than that of the n^{th} level. As a result, the $(n-1)^{th}$ level is blurrier than the n^{th} level (see the first row of Figures 5 and 6). This explains the reason of the blurred effect when the model is zoomed out, that is, the continuous diminution of the variance of higher level texture.

3.2.2 Customized noise texture pyramid

To address the aforementioned problem, two requirements should be met: the maps of the adjacent levels have adequate correlation to make the result consistent and avoid popping, and meanwhile, the variance of the map at each level is stable to avoid blurred effect.

For the first goal, the noise texture pyramid should be generated from coarse to fine instead of a fine-to-coarse process which used in standard mipmap generation techniques. Such a coarse-to-fine strategy is also used to generate stroke-based texture in [18] for preserving the stroke resolution in different scale. Beginning from the coarsest map at the resolution of 1×1, a finer map can be successively generated by means of the following rule:

$$I_{n+1}(2x+\Delta x, 2y+\Delta y) = I_n(x,y) + \sigma(2\gamma_{\Delta x,\Delta y} - 1), \quad \Delta x,\Delta y \in \{0,1\} \tag{2}$$

where $\gamma_{\Delta x,\Delta y}$ is a random number uniformly distributed between $[0,1]$, and $\sigma \in R$ is an adjustable number to tune the variance.

Applying Equation 2 can produce a consistent texture pyramid. However, the value will be out of $[0,1]$ when $I_n(x,y)$ is close to 0 or 1. Clamping it into $[0,1]$ yields obvious color blocks as shown in the second row of Figures 5 and 6. These artifacts are actually caused by inappropriate local variances.

Thus, a better solution would take both the correlation and variance into account, which is essentially an optimization problem. We design a simple and efficient scheme: $I_n(x,y)$ is linearly mapped from $[0,1]$ into $[\eta, 1-\eta], \eta \in [0,0.5]$, and then we add four random numbers $\xi_{\Delta x,\Delta y}$ to generate $I_{n+1}(2x+\Delta x, 2y+\Delta y)$ in the $(n+1)^{th}$ level. This can be expressed as:

$$\begin{aligned} I_{n+1}(2x+\Delta x, 2y+\Delta y) = \\ ((1-2\eta)I_n(x,y)+\eta) + \xi_{\Delta x,\Delta y} := I'_n(x,y) + \xi_{\Delta x,\Delta y} \end{aligned} \tag{3}$$

To ensure $I_{n+1}(2x+\Delta x, 2y+\Delta y) \in [0,1]$, $\xi_{\Delta x,\Delta y}$ should be in $[-I'_n(x,y), 1-I'_n(x,y)]$. The difference between the mean of four values $I_{n+1}(2x+\Delta x, 2y+\Delta y)$ and $I_n(x,y)$ is $\eta(1-2I_n(x,y))+\alpha$, where α stands for the mean of the four random numbers $\xi_{\Delta x,\Delta y}$. Letting $\alpha = \eta(2I_n(x,y)-1)$ will increase the correlation, but will decrease the variance especially when $I_n(x,y)$ is close to 0 or 1. Thus we choose to generate four random numbers with the zero expectation. Thus, we use the following equation to compute the required random numbers in the range of $[-I'_n(x,y), 1-I'_n(x,y)]$ with zero expectation:

$$\xi_{\Delta x,\Delta y} = \text{power}\left(\gamma_{\Delta x,\Delta y}, \frac{1-I'_n(x,y)}{I'_n(x,y)}\right) - I'_n(x,y) \tag{4}$$

Finally it turns out that the values of the four pixels in level $n+1$ are:

$$I_{n+1}(2x+\Delta x, 2y+\Delta y) = \text{power}\left(\gamma_{\Delta x,\Delta y}, \frac{1-I'_n(x,y)}{I'_n(x,y)}\right) \tag{5}$$

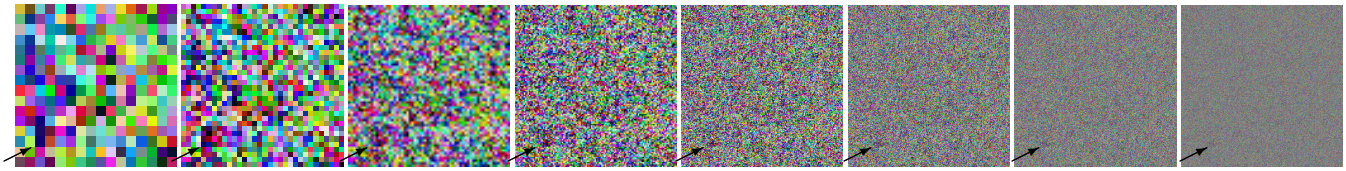


Figure 7: The sequence of the mipmap texture images (from 16×16 to 2048×2048). The arrows point to the relatively dark regions.

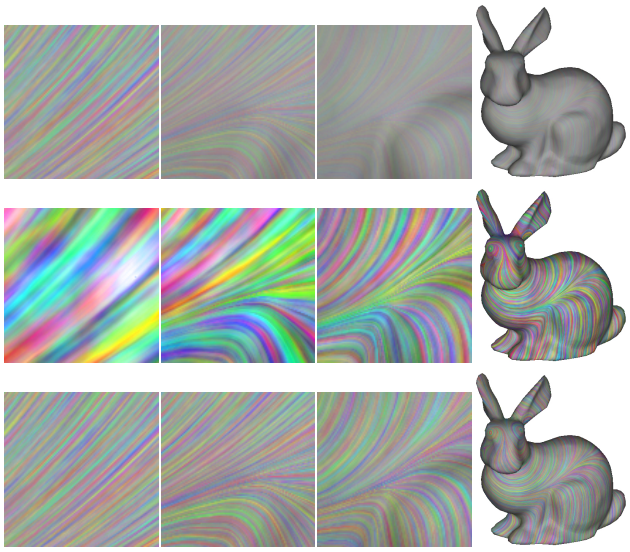


Figure 6: From left to right, the model is zoomed out. The first row shows the results at different resolutions with the automatic-generated mipmap. The results become blurred when the model is zoomed out (from left to right). The second row shows the results by employing Equation 2, where big color blocks appear when the model is zoomed in (from right to left). The last row presents the best quality with Equation 3.

In other words, four random numbers in the range $[0,1]$ with expectation $I'_n(x,y)$.

Note that η should be small to ensure high correlation. Whereas, a small η will yield small variance, i.e., blurred or low contrast result. Experimental results (e.g., Figures 5, 6 and 7) indicate that 0.25 is a good choice.

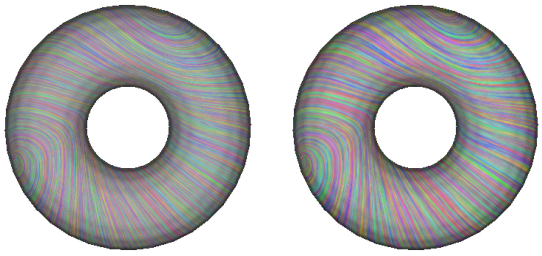
3.3 Enhanced LIC Visualization

We design several enhancement techniques to allow for easy controls on visualization effects such as the resolution, the contrast, and the length of streamlines.

Streamline resolution By modulating the parameter of GL_TEXTURE_LOD_BIAS, the noise density used for LIC is changed, and consequently is the resolution of streamlines. Figure 8 shows such an example.

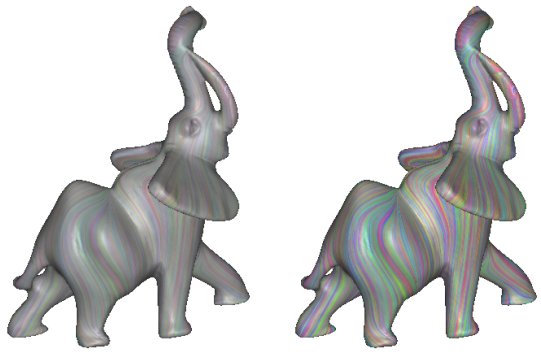
Streamline contrast Palacios and Zhang [17] proposed a contrast correction scheme to eliminate the contrast loss introduced by image blending. It works by enlarging the image variance and fixing the mean. We employ this scheme to enhance the contrast, and additionally modulate the saturation to adjust the contrast. Figure 9 demonstrates our results.

Streamline length Concerning the LIC, the streamline length is adjustable and influences the results, as shown in Figure 10. There are two ways to measure the streamline length. If the streamline length is measured in the image space, the length scales well with



(a) (b)

Figure 8: Results with different resolutions. The triangles are mapped onto the noise texture with a higher resolution noise texture in the pyramid for (a), and thus the streamlines are thinner than those in (b).



(a) (b)

Figure 9: Result comparison on the elephant model without (a) and with streamline contrast enhancement (b). This modulation leads to a balance between the details and contrast.

the model zooming. Although a degree of almost invisible inconsistency may appear, a scale-dependent LIC is achieved. On the other side, measuring the length in the object space increases the consistency, but loses the property of adaptive granularity. In practice, the user can choose the appropriate way for different purposes, e.g. varying the streamline length according to the vector magnitude.

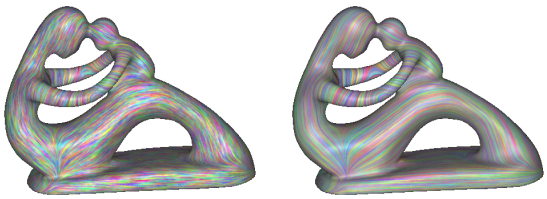
3.4 Special Care on the surface silhouette

For image-space surface LIC approaches, special care must be taken on the object-space geometric discontinuities around the surface boundaries, i.e., the silhouette in the image space. The reason is that the integration of a streamline in the image space stops at the silhouette while in the object space its counterpart will continue onto the back face of the surface.

The silhouette is composed of points that meet the following condition [12]:

$$|z_{i+1} - z_i| > \varepsilon |p_{i+1} - p_i| \tag{6}$$

where ε is an adjustable threshold. p_{i+1} and p_i are two consecutive points along the integral path in the image space, and z_{i+1} and z_i



(a) (b)

Figure 10: The streamline length on the fertility model used in (b) is longer than that of (a).

are their depth values in the object space.

To compensate the missed parts of the streamlines along the silhouette, a conventional solution [15] generates a random value for each pixel in the detected silhouette. This may lead to popping artifacts near the silhouette between two consecutive frames.

It is desired that both the streamline length and the value in the silhouette should be made consistent. We propose a two-stage scheme to drive the integration of the streamlines onto the back face of the model. First, the front and back faces of the underlying surface are extracted by rendering the surface twice, i.e., with the back-face culling and front-face culling operations respectively. In this process, the projected vector field and the noise texture in the image space that correspond to both faces are generated. Second, the conventional LIC is performed in the image space for the front face part. When the integration of certain streamline meets the silhouette, its value is computed by considering the back-face parts of the vector field and the noise texture.

4 Results

The primary outcome of our approach is that it achieves comparable quality as the flow charts approach [14] and interactive performance, yet with a simple and efficient means. The results shown in the figures and video demonstration verify the efficiency and robustness of our approach. The vector fields on Moai, bunny, Buddha, fish and CAD model are generated according to some vector field design methods like [28, 3]. The others are from real world simulations, like the cooling jacket data [9] which is from Robert S. Laramee at Swansea University.

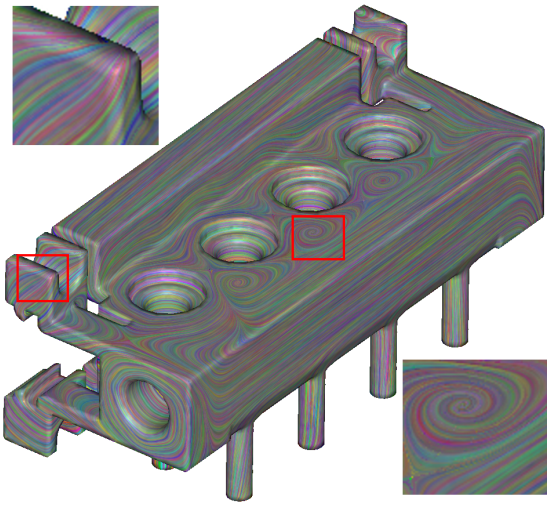


Figure 11: Result for a complex CAD model. To locate and check the singularities on the model, consistent LIC visualization is very helpful for such tasks.

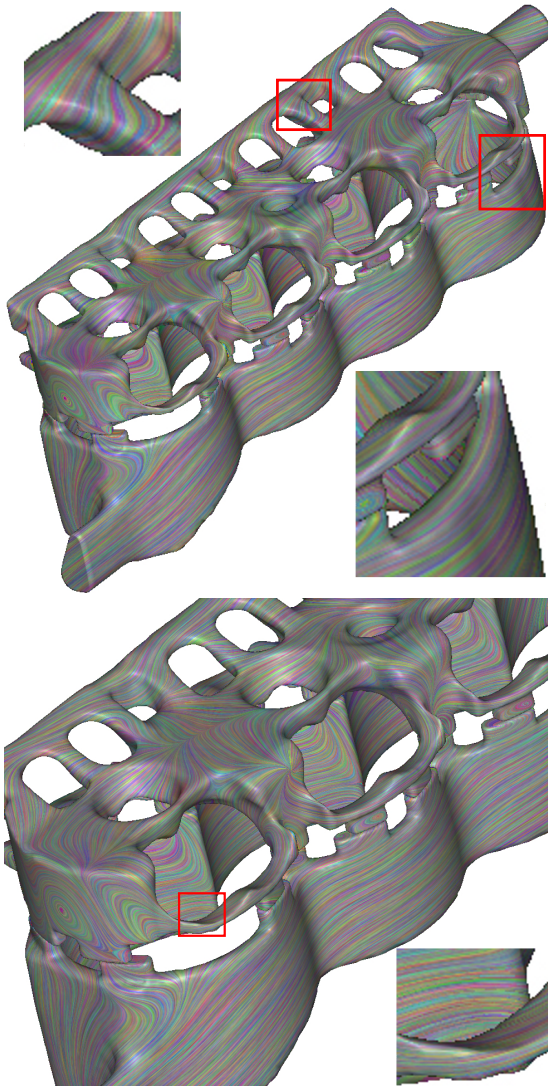
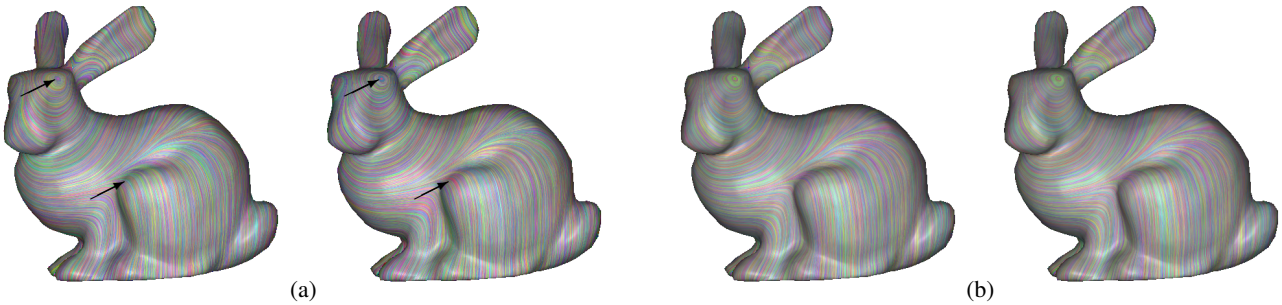


Figure 13: Visualization of a vector field on the boundary geometry of a cooling jacket.

Another benefit of our approach is that it is completely parameterization-free, and works for arbitrary mesh models. Figure 11 shows a challenging CAD models. The model is composed of many triangular patches, and is not a manifold model. In addition, a low-distortion global parameterization is intractable even if the patches are merged into a single manifold mesh. Conventional parameter space LIC approaches can hardly handle this model, while our approach can achieve satisfying results. Thanks to the consistent LIC visualization, the user can easily locate and view several interesting singularities. Without the requirement of parameterization, our method can also easily handle large dataset with complex surface topology. In Figure 13, we visualize the flow from a cooling jacket simulation. The model has over 227K faces and many holes, yielding big troubles for the parameter space method, even for the method [14] with multi-chart parameterization.

Figures 4 and 6 compare our approach with the ones without the proposed special care. It is apparent that our approach outperforms these configurations with negligible computational cost.

We demonstrate various enhancement effects in Figures 8, 9 and 10. In fact, these enhancements are fully controllable, and can be incorporated together without sacrificing the interactivity of

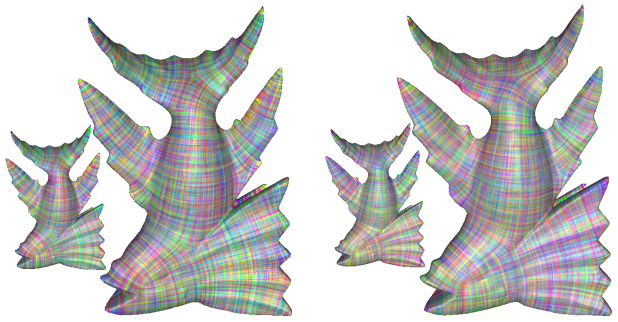


(a) (b)

Figure 12: Two consecutive frames during rotation of the bunny model. Our result in (b) is more consistent than that of [15] shown in (a). Please see the accompany video for a better illustration.

the vector field visualization (see the accompany video). This verifies the advantages of the image-space surface LIC over the object-space surface LIC schemes.

Our method can be easily integrated into other LIC algorithm. As a demonstration, we apply the noise map scheme into the LIC method [17] to visualize rotational symmetry field [7, 19, 20], and show the result in Figure 14.



(a) (b)

Figure 14: Our method is simple enough to easily integrate with other LIC technique. (a) shows the result of applying our method to visualize rotational symmetry field [17]. Compared with (b) which uses a set of uncorrelated images in the texture pyramid, our method achieves better color consistency in different scales.

4.1 Performance

Compared with conventional image-space LIC approaches, our approach requires additional cost for computing a noise texture pyramid. Fortunately, it can be pre-computed and reused for all models. Another difference lies in the way of computing the texture coordinates of each triangle, which is randomly and quickly determined in our approach. In the visualization stage, our approach projects the underlying vector field and the noise texture to the image space twice to handle the silhouette. This is still fast because every operation can be implemented in GPU.

Table 1 lists the performance measured on the bunny model with 12K faces for the image resolutions of 500×500 and 1000×1000 respectively. Our current implementation is not fully optimized compared with the same GPU implementation in [17], which can render the LIC result in less than 20 ms at the image resolution of 512×512. Although our approach needs to process the vector field twice (for the front face and the back face of the model), for the cooling jacket model with over 227K triangle faces at the resolution of 1000×1000 (Figure 13) it still achieves the frame rate of 5 fps.

5 Conclusions and Future Works

This paper proposes a novel image-space surface LIC technique to achieve consistent and smooth LIC results. Coherent visualization

	500×500	1000×1000
vector projection	31	31
LIC without back face	62	78
LIC with back face	108	127
adjusting of contrast	15	30

Table 1: Performance measurement for the bunny model in two different resolutions (the time unit is milliseconds).

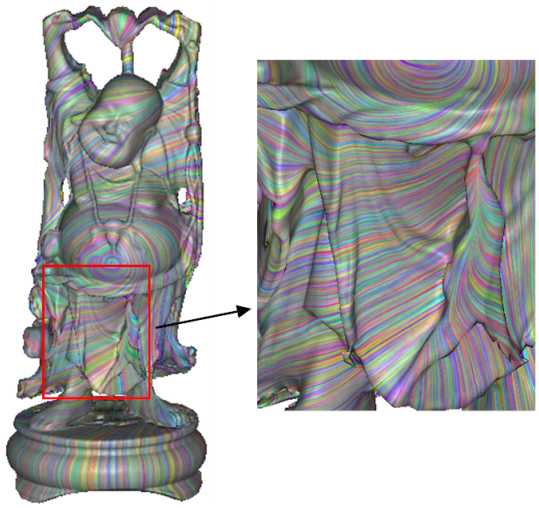


Figure 15: Result for a complicated Buddha model. The complex wrinkles still cause some popping although our silhouette treatment has dramatically reduced such artifacts.

is achieved when the model is rotated or zoomed by leveraging a consistent 2D noise texture pyramid, which can be pre-computed with the computational and memory complexity of $O(N^2)$ and used for arbitrary data. Our method not only provide smooth transition during the user manipulation, but also leads to similar color distribution in different scales. Besides several enhancement techniques to allow for easy controls on visualization effects, we continue the streamlines crossing the silhouette to the back face which alleviates the popping artifacts efficiently. These techniques are easy to be incorporated into the GPU implementation. The integrated toolkit compares favorably with existing solutions in terms of performance and quality, and enables the user to interactively study the vector fields on a surface in any resolution.

Because our method shares some similarities to [27] in using object-space texture coordinates and hierarchical texture image, it is possible that our method can also be extended to visualize the unsteady flow as shown in [27]. Thus, for the future work, it is natural to apply the present technique to unsteady flows and to

combine with the visualization of integral surfaces (e.g. stream-surfaces [10]).

Our approach can provide consistent LIC visualizations for most models. However, some complicated models (e.g., the model shown in Figure 15) have many small wrinkles, which lead to popping artifacts when the model is rotated. The technique presented in Section 3.4 for handling the silhouette effectively alleviates the popping artifacts along the silhouette. Yet, it can only handle the cases where a streamline passes the silhouette once. For a very complicated surface, a depth peeling is needed to extract more than two layers (the front and back faces) to achieve the same effect as the object-space surface LIC approaches. In addition, the popping artifacts may still exist because the numerical error near the silhouette can be large. We plan to explore more sophisticated solution to handle the silhouette issue when the model under certain view has multiple depth layers.

Although 10 level mipmap in our implementation is already adequate for most cases, the fixed level of pyramid restricts the range of zoom-in and zoom-out: the result gets popping for very far view point and gets rough when view point is very close to the surface. It is interesting to explore new method to construct the texture image on-the-fly according to the current scale without rapidly increasing the memory consuming.

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for their valuable comments. This research was partially supported by NSFC (No. 61170139 and No. 60873123), China 973 Program (No. 2009CB320801) and ZJKJGY (2011C21058). Guoning Chen was partially supported by DOE VACET.

REFERENCES

[1] H. Battke, D. Stalling, and H. Hege. Fast line integral convolution for arbitrary surfaces. *In Visualization and Mathematics*, pages 181–195, 1997.

[2] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. *In Proceedings of ACM SIGGRAPH*, pages 263–272, 1993.

[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, July 2007.

[4] M. Falk and D. Weiskopf. Output-sensitive 3D line integral convolution. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):820–834, July 2008.

[5] L. Forssell. Visualizing flow over curvilinear grid surfaces using line integral convolution. *In Proceedings of IEEE Visualization*, pages 240–247, 1994.

[6] L. K. Forssell and S. D. Cohen. Using line integral convolution for flow visualization: Curvilinear grids, variable-speed animation, and unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):133–141, 1995.

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

[8] D. H. Laidlaw, R. M. Kirby, C. D. Jackson, J. S. Davidson, T. S. Miller, M. da. Silva, W. H. Warren, and M. J. Tarr. Comparing 2D vector field visualization methods: A user study. *IEEE Transactions on Visualization and Computer Graphics*, 11(1):59–70, 2005.

[9] R. S. Laramee, C. Garth, H. Doleisch, J. Schneider, H. Hauser, and H. Hagen. Visual analysis and exploration of fluid flow in a cooling jacket. In *In Proceedings IEEE Visualization 2005*, pages 623–630, 2005.

[10] R. S. Laramee, C. Garth, J. Schneider, and H. Hauser. Texture advection on stream surfaces: A novel hybrid visualization applied to CFD simulation results in data visualization. In *Proceedings of the Joint EUROGRAPHICS - IEEE VGTC Symposium on Visualization (EuroVis 2006)*, pages 155–162, Lisbon, Portugal, May 2006.

[11] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum*, pages 203–221, 2004.

[12] R. S. Laramee, J. Schneider, and H. Hauser. Image space based visualization of unsteady flow on surfaces. *In Proceedings of IEEE Visualization*, pages 131–138, 2003.

[13] R. S. Laramee, J. van Wijk, B. Jobard, and H. Hauser. ISA and IBFVS: Image space-based visualization of flow on surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):637–648, 2004.

[14] G.-S. Li, X. Tricoche, D. Weiskopf, and C. D. Hansen. Flow charts: Visualization of vector fields on arbitrary surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1067–1080, 2008.

[15] W. C. Li, B. Vallet, N. Ray, and B. Levy. Representing higher-order singularities in vector fields on piecewise linear surfaces. *IEEE Transactions on Visualization and Computer Graphics*, pages 1315–1322, 2006.

[16] A. Okada and D. Lane. Enhanced line integral convolution with flow feature detection. In *In SPIE Vol. 3017 Visual Data Exploration and Analysis IV*, pages 206–217, San Jose, California, January 1997.

[17] J. Palacios and E. Zhang. Interactive visualization of rotational symmetry fields on surfaces. *IEEE Transections on Visualization and Computer Graphics*, 17(7):947–955, July 2011.

[18] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 581–586, New York, NY, USA, 2001. ACM.

[19] N. Ray, W. C. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic global parameterization. *ACM Trans. Graph.*, 25(4):1460–1485, October 2006.

[20] N. Ray, B. Vallet, W. C. Li, and B. Lévy. N-symmetry direction field design. *ACM Trans. Graph.*, 27(2):10:1–10:13, May 2008.

[21] H.-W. Shen, C. R. Johnson, and K.-L. Ma. Visualizing vector fields using line integral convolution and dye advection. In *Proceedings of the IEEE Symposium on Volume Visualization 1996*, pages 63–70, October 1996.

[22] H.-W. Shen and D. L. Kao. A new line integral convolution algorithm for visualizing time-varying flow fields. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):98–108, April 1998.

[23] D. Stalling and H. Hege. Fast and resolution independent line integral convolution. *In Proceedings of ACM SIGGRAPH*, pages 249–256, 1995.

[24] J. J. van Wijk. Spot noise-texture synthesis for data visualization. *Computer Graphics (Proceedings of ACM SIGGRAPH91)*, 25:309–318, 1991.

[25] J. J. van Wijk. Image based flow visualization. *ACM Trans. Graphics*, 21(3):745–754, 2002.

[26] J. J. van Wijk. Image based flow visualization for curved surfaces. *In Proceedings of IEEE Visualization*, pages 123–130, 2003.

[27] D. Weiskopf and T. Ertl. A hybrid physical/device-space approach for spatio-temporally coherent interactive texture advection on curved surfaces. In *Proceedings of Graphics Interface 2004*, GI '04, pages 263–270. Canadian Human-Computer Communications Society, 2004.

[28] E. Zhang, J. Hays, and G. Turk. Interactive tensor field design and visualization on surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):94–107, January 2007.

APPENDIX

Given the OpenGL's ModelView matrix M_{mv} and Project matrix M_{pj}, we compute the image-space coordinates $\bar{p}$ of a point p in the object space according to the standard OpenGL specification:

$$\bar{p} = \frac{M(1:2,1:3)p + M(1:2,4)}{M(4,1:3)p + M(4,4)}$$

where $M = M_{pj}M_{mv}$ and the above notation for sub-matrix follows the grammar of Matlab script.