

UFLIC: A Line Integral Convolution Algorithm For Visualizing Unsteady Flows

Han-Wei Shen*
MRJ Technology Solutions
NASA Ames Research Center

David L. Kao†
NASA Ames Research Center

Abstract

This paper presents an algorithm, UFLIC (Unsteady Flow LIC), to visualize vector data in unsteady flow fields. Using the Line Integral Convolution (LIC) as the underlying method, a new convolution algorithm is proposed that can effectively trace the flow's global features over time. The new algorithm consists of a time-accurate value depositing scheme and a successive feed-forward method. The value depositing scheme accurately models the flow advection, and the successive feed-forward method maintains the coherence between animation frames. Our new algorithm can produce time-accurate, highly coherent flow animations to highlight global features in unsteady flow fields. CFD scientists, for the first time, are able to visualize unsteady surface flows using our algorithm.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation ; I.3.6 [Computer Graphics]: Methodology and Techniques; I.4.3 [Image Processing]: Enhancement.

Additional Keywords: flow visualization, vector field visualization, image convolution, line integral convolution, flow animation, unsteady flows, surface flows, texture synthesis.

1 Introduction

Vector field data arise from computer simulations in a variety of disciplines such as computational fluid dynamics (CFD), global climate modeling, and electromagnetism. Visualizing these vector data effectively is a challenging problem due to the difficulties in finding suitable graphical icons to represent and display vectors on two-dimensional computer displays. Presently, new challenges emerge as time-dependent simulations become ubiquitous. These simulations produce large-scale solutions of multiple time steps, which carry complex dynamic information about the underlying simulation model. To visualize these time-varying data, two types of methods are generally used. One may be referred to as the *instantaneous method* where calculations are based on an instance of the data field in time. Examples are streamlines and vector plots. The other method is the *time-dependent method* which can better characterize the evolution of the flow field by continuously tracking the visualization results over time. Examples are streaklines and pathlines computed from unsteady flow data [4, 5].

This paper presents a time-dependent method for visualizing vector data in unsteady flow fields. Using the *Line Integral Convolution* (LIC) [1] as the underlying method, we propose a new convolution algorithm, called *UFLIC* (Unsteady Flow LIC), to accurately model the unsteady flow advection. The Line Integral Convolution method, originally proposed by Cabral and Leedom, is a visualization technique that synthesizes textures based on the vec-

tor data input to characterize the flow field's global features. While this method is effective, it is primarily for steady flow data. Forssell and Cohen [2] propose an extension using a streakline convolution¹ for the time-varying vector field. However, several problems associated with the algorithm, such as obscure coherence within and among the output image frames and less accurate temporal manipulation, greatly limit its effectiveness in practice. A detailed analysis of their algorithm is provided in the next section. In our algorithm, we propose a time-accurate value depositing scheme and a successive feed-forward method to perform the line integral convolution. By progressively updating the visualization results in time, UFLIC can produce highly coherent animation frames and accurately trace the dynamic flow movement. The main contribution of this work is to provide a time-accurate global visualization technique to analyze unsteady flow data.

We begin this paper by giving an overview of the LIC method. Next, we describe and analyze Forssell and Cohen's algorithm for unsteady flows. We then present our UFLIC algorithm. We conclude this paper by presenting results of applying our method to several unsteady flow data sets from CFD simulations.

2 Background and Related Work

In this section, we briefly review the LIC method proposed by Cabral and Leedom [1]. We then describe and analyze the method proposed by Forssell and Cohen [2] for unsteady flow field data.

2.1 Line Integral Convolution

The Line Integral Convolution method is a texture synthesis technique that can be used to visualize vector field data. Taking a vector field and a white noise image as the input, the algorithm uses a low pass filter to perform one-dimensional convolution on the noise image. The convolution kernel follows the paths of streamlines originating from each pixel in both positive and negative directions. As a result, the output intensity values of the LIC pixels along each streamline are strongly correlated so the global features of the flow field can be easily visualized. To perform the convolution, different periodic filter kernels can be used. Examples are the Hanning filter [1] and the box filter [8][7].

Recently, several extensions to the original LIC algorithm have been proposed. Forssell and Cohen [2] adapt the LIC method for curvilinear grid data. Stalling and Hege [8] propose an efficient convolution method to speed up the LIC computation. Shen, Johnson, and Ma [7] combine dye advection with three dimensional LIC to visualize global and local flow features at the same time. Okada and Kao [6] use post-filtering techniques to sharpen the LIC output and highlight flow features such as flow separations and reattachments. Recently, Kiu and Banks [3] propose using multi-frequency noise

*NASA Ames Research Center, Mail Stop T27A-2, Moffett Field, CA 94035 (hwshen@nas.nasa.gov)

†NASA Ames Research Center, Mail Stop T27A-2, Moffett Field, CA 94035 (davidkao@nas.nasa.gov)

¹As described in their paper. However, in a strict sense, their convolution follows pathlines of advected particles.

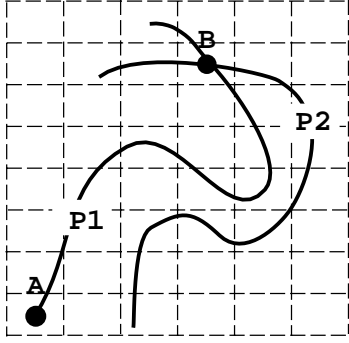


Figure 1: Uncorrelated values of A and B

input for LIC to enhance the contrasts among regions with different velocity magnitudes.

The texture outputs from the Line Integral Convolution method provide an excellent visual representation of the flow field. This effectiveness generally comes from two types of coherence. The first is *spatial coherence*, which is used to highlight the flow lines of the field in the output image. The LIC method establishes this coherence by correlating the pixel values along a streamline as the result of the line integral convolution. The second type of coherence is *temporal coherence*. This coherence is required for animating the flow motion. The LIC method achieves this temporal coherence by shifting the filter phase used in the convolution so that the convolved noise texture can periodically move along the streamlines in time.

2.2 Line Integral Convolution for Unsteady Flows

The LIC technique proposed originally is primarily for visualizing data in steady flow fields. To visualize unsteady flow data, Forssell and Cohen propose an extension [2]. In contrast with convolving streamlines in the steady flow field, the algorithm convolves forward and backward pathlines originated from each pixel at every time step. A pathline is the path that a particle travels through the unsteady flow field in time. To animate the flow motion, the algorithm shifts the filter’s phase at every time step to generate the animation sequence.

Forsell and Cohen’s algorithm for visualizing unsteady flows has several problems. First, their algorithm does not establish clear spatial coherence. To explain this, in Figure 1 a pathline P_1 that starts from pixel A at time T_1 and passes through pixel B at time T_2 is the convolution path for pixel A . Similarly, pathline P_2 starting from B at time T_1 is the convolution path for pixel B . Since pathlines P_1 and P_2 pass through B at different times (T_2 and T_1 , respectively), they have different traces. As a result, convolution values of A and B are uncorrelated because two different sets of pixel values are used. Hence, no spatial coherence is established in either pathline P_1 or P_2 . In Forsell and Cohen’s paper, it is mentioned that flow lines in the output images become ambiguous when the convolution window is set too wide. The problem mentioned here can explain this phenomenon.

The second problem of the algorithm is that the convolution value of each pixel at any given time is derived from a mixture of past and future flow information. For instance, in Figure 1, pathline P_1 starting from A does not reach B until T_2 . However, in the algorithm, B ’s pixel value is included in the convolution of pixel A at time T_1 . This mixture implies that the convolution result at any given time can be influenced by what will happen in the future, which is unphysical.

The third problem of the algorithm is that the temporal coherence in the unsteady flow field is difficult to establish by using the phase-shift method. This is because the pathlines from the same seed point vary over time unless the flow is relatively steady. Therefore, in the algorithm the same filter with shifted phases is applied to different convolution paths. However, the effectiveness of using the phase-shift method to create artificial motion effects mainly relies on the fact that the convolution is applied to the same path over time. As a result, the temporal coherence between consecutive frames to represent the flow motion using this phase-shift method becomes rather obscure for unsteady flow data.

Another problem is the lack of accurate time stepping in Forsell and Cohen’s method. In the algorithm, the time variable in the pathline integration advances only when the particle crosses cell boundaries. This approximation greatly sacrifices the time accuracy because it is the step size in the numeric integration and the physical velocity, instead of the grid geometry, which determine the advancement of time. This assumption poses a major drawback of the algorithm because time accuracy is crucial in analyzing time-dependent flow data.

Due to the above problems, the effectiveness of Forsell and Cohen’s algorithm is greatly limited. In the following section, we propose a new algorithm for visualizing unsteady flow fields. The new algorithm consists of a time-accurate value depositing scheme and a successive feed-forward method. Our algorithm provides a time accurate, highly coherent solution to highlight dynamic global features in unsteady flow fields.

3 Algorithm

In this section, we present a new convolution algorithm, UFLIC, for visualizing unsteady flows. First, we present a time-accurate value depositing scheme. Next, we discuss a successive feed-forward method. The combination of these two techniques can provide effective spatial and temporal coherence for tracking dynamic flow features.

3.1 Time-Accurate Value Depositting

The convolution method proposed originally by Cabral and Leedom [1] uses a *value gathering* scheme. In this algorithm, each pixel in the field travels in both positive and negative streamline directions to gather pixel values and compute the convolution. Stalling and Hege [8] propose a different approach for speeding up the computation. Rather than gathering, each pixel deposits its value along the streamlines. To compute the convolution, values deposited at each pixel are collected to perform the integration. This method can be referred to as a *value depositing scheme*. For steady state vector fields, value gathering and value depositing are equivalent. For time-varying vector fields, value gathering and value depositing can produce different results when the convolution of a streamline is extended to the pathline. To illustrate this, again in Figure 1, the pathline starting from pixel A to pixel B enables B to receive a contribution from A if the depositing scheme is used. However, when using the gathering scheme, B does not gather A ’s value because the pathline P_2 starting from B at T_1 does not pass A . To accurately reflect the physical phenomena in unsteady flow fields, value depositing is superior to value gathering. This is because value depositing by nature corresponds to flow advection. On the other hand, the value gathering scheme, as the one used in Forsell and Cohen’s algorithm, possesses several difficulties as we discussed in the previous section.

We propose a new algorithm to compute the line integral convolution for unsteady flows. In the algorithm, we use a *time-accurate value depositing scheme*, which incorporates time into the convolution, to simulate the flow advection in unsteady fields. Our deposit

scheme works as follows. As in the original LIC algorithm, initially a white noise image is used as the input texture. To “smear” this input texture, each pixel in the field serves as a *seed* point to advect forwards following the pathline direction. The pathline can be defined as:

$$\mathbf{p}(t + \Delta t) = \mathbf{p}(t) + \int_t^{t+\Delta t} \mathbf{v}(\mathbf{p}(t), t) dt$$

where $\mathbf{p}(t)$ is the position of the particle at time t , $\mathbf{p}(t + \Delta t)$ is the new position after time Δt , and $\mathbf{v}(\mathbf{p}(t), t)$ is the velocity of the particle at $\mathbf{p}(t)$ at time t . The *Runge-Kutta* second or fourth order numeric integration scheme can be used to evaluate the above equation and generate particle traces. At each integration step, the seed point deposits a triplet (ν, ω, τ) at the pixel where the particle is currently located. ν is the input texture value of the seed pixel, ω is the arc length used to normalize ν , and τ is the physical time at the current integration step. Assuming that the seed pixel’s value is V , the pathline starts from the seed pixel at physical time T_0 , the i^{th} integration step size is Δt_i , then at step n , the physical time is:

$$T_n = T_0 + \sum_{i=1}^{i=n} \Delta t_i$$

The deposited weight ω is the distance between the particle locations in the current and previous integration steps:

$$\omega_n = \|\mathbf{p}(T_{n-1}) - \mathbf{p}(T_n)\|$$

The deposited pixel value ν is:

$$\nu_n = V$$

And the deposited time τ is:

$$\tau_n = T_n$$

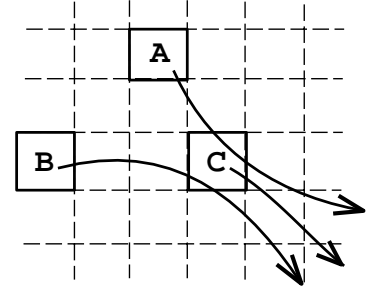
For each seed pixel, the distance that its pathline can travel is defined as the convolution length. We determine this convolution length indirectly by specifying a period of physical time as the pathline’s life span. This life span is a global property that we use for all the pathlines in the convolution. The advantage of using this global life span to control the convolution length is that the lengths of different pathlines can be automatically scaled to be proportional to the velocity magnitudes, which is a desirable effect as described in [1, 2]. It is worth mentioning that in our method, each pixel deposits its value along only the forward pathline direction but not the backward direction. This is because the backward depositing does not correspond to actual physical phenomena in practice (flows do not advect backwards). In addition, the symmetry issue mentioned in [1] does not appear as a problem in our unsteady flow animations.

To receive the deposits, each pixel keeps a “bucket” which stores deposits from different seeds at different time. In Figure 2(a), pixels A , B , and C have pathlines stepping through C . As a result, C has a deposit from each source as shown in Figure 2(b).

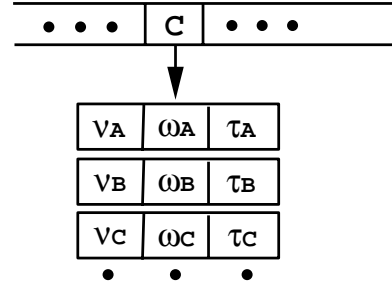
To compute the convolution frame at time T , we integrate only the deposits that are made at or before T . These correspond to deposit triplets (ν', ω', τ') that have time-stamp τ' smaller than T . For each pixel (x, y) , the convolution result $C(x, y)$ can be computed using the formula:

$$C(x, y) = \sum_{\tau' < T} (\nu' * \omega') / \sum_{\tau' < T} \omega'$$

where triplets (ν', ω', τ') were deposited to pixel (x, y) ’s bucket, and the deposited pixel values ν' are normalized by weights ω' .



(a)



(b)

Figure 2: Time-Accurate Value Deposit Scheme

Once the convolution is completed, the used deposits are discarded from the bucket.

In our algorithm, the above convolution process is repeated at each time step. However, values of time-stamps in the deposit triplets, determined by the step sizes of pathline integration, can be real numbers anywhere in the time interval of the simulation’s entire course. Therefore, when computing the convolution, we need to integrate any deposit that has a time-stamp between the physical time in the current and the previous step. This explains why the deposits with time-stamp values smaller than the current physical time should be selected.

Our value depositing scheme provides a time-accurate model simulating the flow advection to create spatial coherence in the output texture. In the next section, we describe the process that successively transports the convolution results over time to maintain the temporal coherence for visualizing unsteady flows.

3.2 Successive Feed-Forward

As mentioned previously, we define a time-dependent method as one that progressively tracks the visualization results over time. In this section, we propose a time-dependent process, called *Successive Feed-Forward*, which is combined with the value depositing scheme to create temporal coherence. Our algorithm works as follows. Initially, the input to our convolution algorithm is a regular white noise texture. Our value depositing scheme advects and convolves the noise texture to obtain the convolution result at the first time step. For the subsequent convolutions, instead of using the noise texture again, we use the output from the previous convolution as the input texture. This input texture, showing patterns that have been formed by previous steps of the flow field, is then further advected. As a result, the output frames in consecutive time steps are

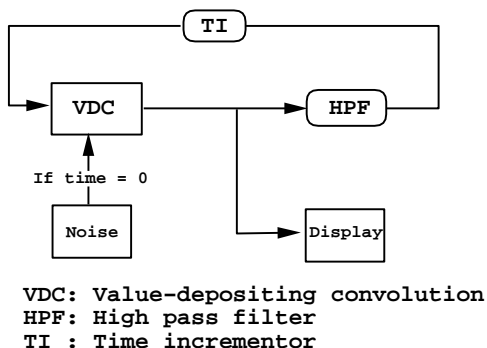


Figure 3: Algorithm Flowchart

highly coherent because the flow texture is continuously convolved and advected throughout space and time.

An important issue for the successive feed-forward method that must be addressed stems from the fact that the line integral convolution method in general, or our value depositing scheme in particular, is a process of low-pass filtering. This low-pass filtering can gradually reduce contrasts among flow lines when repeatedly being applied to the input texture over time. This would cause problems if one tries to visualize a long sequence of unsteady flow data. To correct this problem, we apply a high-pass filter (HPF) to the input texture, which is the result from the previous convolution, before it is used by the value depositing scheme at the next step. This high-pass filter can enhance the flow lines and maintain the contrast in the input texture so the value depositing scheme can produce images with restored contrast and clearer flow traces. Figure 3 gives an overview of our entire algorithm.

The high-pass filter used in our method is a *Laplacian* operator. A two-dimensional Laplacian can be written as a 3×3 mask:

$$\begin{vmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{vmatrix}$$

The result computed from the mask does not have exclusively positive value. In order to display the result, a common technique used in digital image processing applications is to subtract the Laplacian results from the original image. The filter mask of overall operations of Laplacian and subtraction can be derived as:

$$HPF = \begin{vmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{vmatrix} - \begin{vmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{vmatrix}$$

For flow data in three dimensional space, this filter needs to be extended to three dimensions.

4 Results and Discussions

We have used UFLIC to visualize surface flow data from several CFD simulations. CFD scientists, for the first time, are able to visualize unsteady surface flows using our method. In this section, we show snapshots from animations of the surface flows, which are shown in the accompanying video.

The first example is a two-dimensional flow simulation with dynamic vortices. There are three vortices in the flow. Figure 4 shows a snapshot in a sequence of the surface flow evolving over time. The simulation has two vortices orbiting clockwise around the central

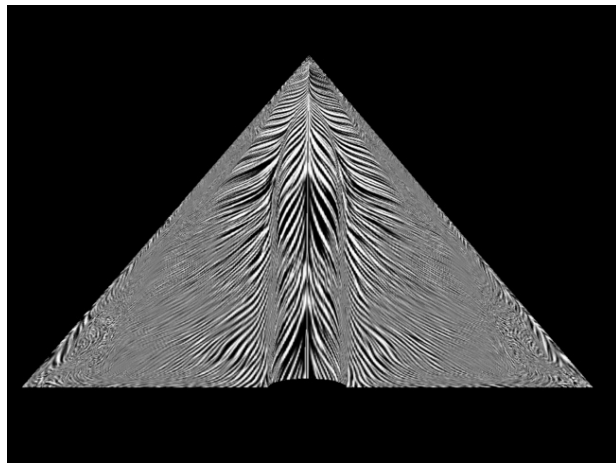


Figure 8: A snapshot of the surface flow on a delta wing at 30 degree angle of attack. The flow is relatively steady along the center of the wing body.

vortex, which spirals counterclockwise. These three vortices generate well-defined flow lines. In the video, the spiraling of the vortices is very dramatic.

Figure 5 shows the same time step as shown in Figure 4, except the flow pattern is now color-mapped by the velocity magnitude. Near the center of each vortex, the velocity is high, as illustrated in the figure. A disadvantage of combining color contours with surface flow patterns is that the color contours tend to appear more visually dominant than the flow patterns, and one needs to pay closer attention to see the surface flow patterns.

Figure 6 shows a time series of unsteady two-dimensional flow over an oscillating airfoil. The airfoil, which is colored in black, pitches down and then up eleven degrees. Images from left to right show the formation of the primary vortex spiraling clockwise above the airfoil, and a counterclockwise secondary vortex forming behind the trailing edge of the airfoil. As the airfoil pitches up and the secondary vortex gains strength, the primary vortex is separated from the airfoil.

Figure 7 shows the same time series as in Figure 6, except the flow is now colored by velocity magnitude contours. Again, the color contours seem to dominate more visually. However, the color contours also add additional insight to the physical phenomena of the flow. For example, the velocity was high near the leading edge of the airfoil at first. Then, as the airfoil pitches down first and then up, the velocity decreases along the leading edge and increases near the trailing edge. This gives strength to the secondary vortex near the trailing edge.

Unsteady surface flows are very useful in detecting flow separations and reattachments. Figure 8 shows surface flow over a delta wing at 30 degree angle of attack. Figure 9 is a close-up view of the delta wing. The image reveals the flow separation and reattachment lines along the leading edge of the delta wing. These flow features are even clearer in the video.

Finally, Figure 10 shows surface flow over a vertical tail of a twin-tailed F18 fighter aircraft at high-angle-of-attack. The aerodynamic load on the fighter aircraft was simulated. During high-angle-of-attack, the flow surrounding the vertical tail of the aircraft becomes highly unsteady, and bursting of vortices near the leading edge of the tail becomes very frequent. The image shows a snapshot of the surface flow that results from the highly unsteady flow.

Some observations can be made regarding visualizing surface

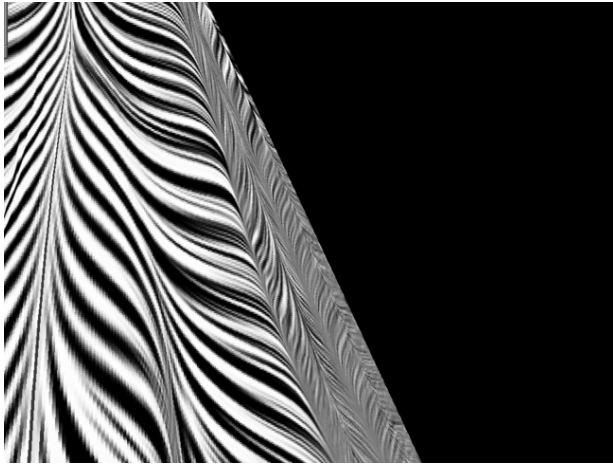


Figure 9: A close-up of the delta wing reveals flow separations and reattachments along the leading edge of the wing.

flows using our algorithm. Blurriness may occur in regions where the direction of the flow is changing rapidly. It may also occur at flow separation and reattachment locations. Unlike the surface flow patterns generated from regular LIC, the flow lines generated using our unsteady LIC may have different linewidths. This can be attributed to changes in the velocity magnitude and the flow direction. Regions where the flows are relatively steady or diverging tend to exhibit thicker flow lines. Since the surface flow pattern is displayed using texture mapping, aliasing effects are visible at different viewing resolutions. In our algorithm, we have not applied any anti-aliasing techniques to the result images.

5 Conclusions and Future Work

We have presented UFLIC, an Unsteady Flow Line Integral Convolution algorithm, for visualizing vector data in unsteady flow fields. Using the time-accurate value deposit scheme and the successive feed-forward method, our new convolution algorithm can accurately model the flow advection and create highly coherent flow animations. The results from several case studies using our algorithm have shown that the new technique is very effective in capturing dynamic features in unsteady flow fields.

Future work includes adapting the acceleration technique proposed in [8] into our unsteady flow algorithm and applying our method to 3D data sets. In addition, we would like to compare our unsteady LIC method with the *Spot Noise* technique introduced by van Wijk [9]. Spot Noise is an effective method for creating flow texture patterns. The final texture image quality is based on the distribution and the shape of spots. Recently, de Leeuw and van Wijk [10] enhanced the spot noise technique by bending spots based on local stream surfaces. The objective is to produce more accurate flow texture patterns near flow regions with high curvature. To extend the spot noise technique for unsteady flows, there are a few key issues to be resolved. For instance, as spots are advected over time the distribution of the spots can change rapidly. A challenge is to maintain the coherence of the spots over time. Another consideration is that spot bending assumes the flow is steady over the local stream surfaces; however, for unsteady flows this assumption may not be true. We plan to look into these issues.

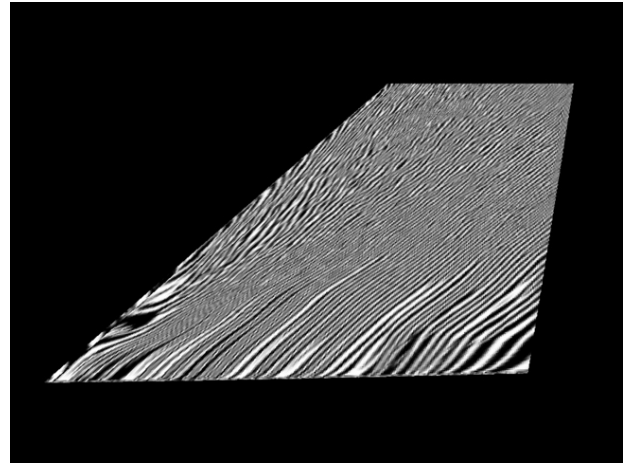


Figure 10: Surface flow over a vertical tail of a twin-tailed F18 fighter aircraft at high-angle-of-attack. The flow is highly unsteady and vortex bursting is frequent near the leading edge of the tail.

Acknowledgments

This work was supported in part by NASA contract NAS2-14303. We would like to thank Neal Chaderjian, Ken Gee, Shigeru Obayashi, and Ravi Samtaney for providing their data sets. We also thank Tim Sandstrom, Gail Felchle, Chris Henze, and other members in the Data Analysis Group at NASA Ames Research Center for their helpful comments, suggestions, and technical support.

References

- [1] B. Cabral and C. Leedom. Imaging vector fields using line integral convolution. In *Proceedings of SIGGRAPH 93*, pages 263–270. ACM SIGGRAPH, 1993.
- [2] 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.
- [3] M.-H. Kiu and D. Banks. Multi-Frequency noise for LIC. In *Proceedings of Visualization '96*, pages 121–126. IEEE Computer Society Press, Los Alamitos, CA, 1996.
- [4] D.A. Lane. Visualization of time-dependent flow fields. In *Proceedings of Visualization '93*, pages 32–38. IEEE Computer Society Press, Los Alamitos, CA, 1993.
- [5] D.A. Lane. Visualizing time-varying phenomena in numerical simulations of unsteady flows. In *Proceedings of 34th Aerospace Science Meeting and Exhibit*. AIAA-96-0048, 1996.
- [6] A. Okada and D. L. Kao. Enhanced line integral convolution with flow feature detection. In *Proceedings of IS&T/SPIE Electronic Imaging '97*, pages 206–217, 1997.
- [7] H.-W. Shen, C.R. Johnson, and K.-L. Ma. Visualizing vector fields using line integral convolution and dye advection. In *Proceedings of 1996 Symposium on Volume Visualization*, pages 63–70. IEEE Computer Society Press, Los Alamitos, CA, 1996.

- [8] D. Stalling and H.-C. Hege. Fast and resolution independent line integral convolution. In *Proceedings of SIGGRAPH 95*, pages 249–256. ACM SIGGRAPH, 1995.
- [9] J.J. van Wijk. Spot noise: Texture synthesis for data visualization. *Computer Graphics*, 25(4):309–318, 1991.
- [10] W. de Leeuw and J.J. van Wijk. Enhanced spot noise for vector field visualization. In *Proceedings of Visualization '95*, pages 233–239. IEEE Computer Society Press, Los Alamitos, CA, 1995.