EUROGRAPHICS 2000 / M. Gross and F.R.A. Hopgood
(Guest Editors)

Volume 19, (2000), Number 3

Unsteady Flow Visualization by Animating Evenly-Spaced Streamlines

Bruno Jobard and Wilfrid Lefer[†]

Université du Littoral Côte d'Opale, France

Abstract

In recent years the work on vector field visualization has been concentrated on LIC-based methods. In this paper we propose an alternative solution for the visualization of unsteady flow fields. Our approach is based on the computation of temporal series of correlated images. While other methods are based on pathlines and try to correlate successive images at the pixel level, our approach consists in correlating instantaneous visualizations of the vector field at the streamline level. For each frame a feed forward algorithm computes a set of evenly-spaced streamlines as a function of the streamlines generated for the previous frame. This is achieved by establishing a correspondence between streamlines at successive time steps. A cyclical texture is mapped onto every streamline and textures of corresponding streamlines at different time steps are correlated together so that, during the animation, they move along the streamlines, giving the illusion that the flow is moving in the direction defined by the streamline. Our method gives full control on the image density so that we are able to produce smooth animations of arbitrary density, covering the field of representations from sparse, that is classical streamline-based images, to dense, that is texture-like images.

1. Introduction

Vector field data are produced by scientific experimentations and numerical simulations, which are now widely used to study complex dynamic phenomena, with various areas of applicability, such as global climate modelling and computational fluid dynamics. Large-scale, time-varying simulations are able to produce large amounts of data in a short time and raises the need for effective techniques to get insight in the data and to extract meaningful information. While several effective techniques have been proposed to visualize steady flow fields, only a few methods exist for visualizing unsteady vector fields. Due to its important impact on a wide application area, visualization of time-varying vector fields is a critical issue of today research. This paper presents a new approach for the visualization of two-dimensional unsteady vector fields.

Several techniques have been proposed to visualize steady flow fields, including icon plots, line representations, and textures. A streamline is a line tangential to the vector field at any point. Covering an image with a set of streamlines is a good way to visualize the flow features. Image quality enhancement can be achieved by using streamline placement algorithms, which optimize the placement of a set of streamlines according to an image-based criterion [10,4,7].

For an unsteady flow, streamlines can be viewed as an instantaneous representation of the vector field because their computation occurs independently for each time step, and thus streamlines do not truly depict the time-varying physical phenomenon over time. Pathlines, streaklines and timelines are particularly devoted to the visualization of unsteady flows [6], but they do not allow to obtain a global representation of the vector field.

Texture-like representations [12,1] allow to increase the spatial resolution and to depict small details accurately. The LIC method is based on the convolution of an input noise

[†] LIL - B.P. 719
62228 Calais
FRANCE
e-mail: {jobard,lefer}@lil.univ-littoral.fr

texture with a one-dimensional filter kernel [1,9]. An extension to handle time-varying vector fields has been proposed in [3], where the convolution is performed on pathlines rather than on streamlines. This approach suffers several drawbacks, such as lack of spatial coherence, non accurate time stepping and problems in establishing a good temporal coherence.

Another approach based on LIC is the UFLIC method [8], which uses a time-accurate value depositing scheme, which performs a temporal convolution of an initial white noise texture, and a successive feed forward algorithm to maintain a high temporal coherence between successive frames. This approach achieves good spatial and temporal coherence but during the animation it is difficult to understand the motion of the flow. Because each image is obtained by a temporal convolution over a number of time steps, regions with low turbulence give rise to highly contrasted areas with zebra stripes of arbitrary thickness, while highly turbulent regions tend to produce blurred zones. Indeed the zebra stripes obtained in regions of low turbulence are oriented in the direction of the flow, they look like thick streamlines and during the animation their shapes are slowly modified so that they actually seem to move in a direction orthogonal to the direction of the flow.

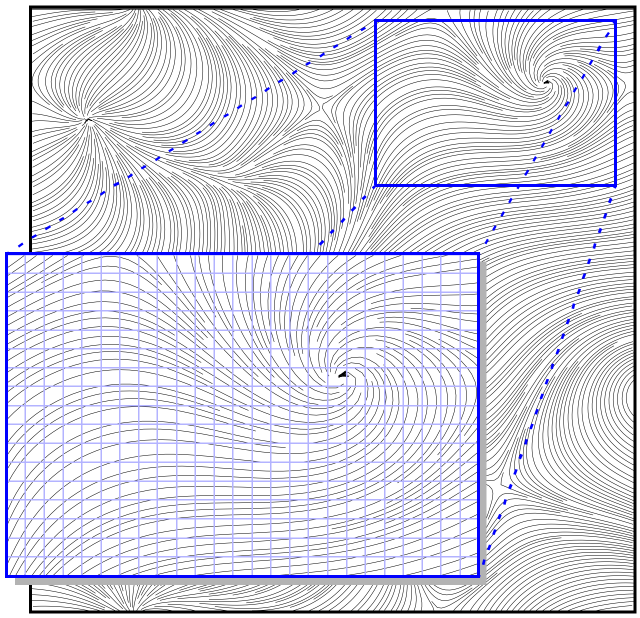


Figure 1: *Decreasing the separating distance allows us to produce dense representations of the flow field.*

In this paper we propose to use the streamline representation to visualize a two-dimensional time-varying flow field. The goal is to show the evolution of the instantaneous representation of the vector field, which is visualized by way of a set of streamlines. Our approach gives full control on the density of the representation, that is we are able to produce frames of arbitrary density, including traditional streamline-based sparse representations and LIC-like dense representations. To obtain a dense representation, we just have to make the separating distance be the size of a pixel so that every pixel will be covered by a streamline (see Figure 1).

While previous approaches try to correlate successive frames at the pixel level, we make it at the streamline level. This is important because it allows us to state the conditions for a good animation as conditions on a set of streamlines. For instance we are able to maintain a good streamline placement at any time, which is impossible by just dealing with pixels. Indeed a good correlation at the streamline level yields a good correlation at the pixel level because all information is conveyed by streamlines, i.e. by pixels lying on them. Our algorithm produces a correlated sequence of streamline-based representations of the flow. Its main features are:

- Optimized streamline placement for each frame using our streamline placement algorithm presented in [4]. The image density can be controlled easily by setting the separating distance between streamlines.
- Motion encoding along each streamline by mapping a set of one-dimensional cyclical textures depicting the orientation of the flow. We propose an extension of the Motion-Map method [5,2] suitable for unsteady flows. Animating the texture gives the illusion that the flow moves along the streamlines.
- Correlation between successive frames using a streamline-oriented correlation algorithm. The basic principle of the algorithm consists in finding the best matching streamline at the next time step and by maximizing the number of correlated streamlines from one frame to the next. Both shape and texture of a streamline are correlated.

As a result our method produces smooth and accurate animations of the structure of the flow over an arbitrary number of frames. Direction and orientation of the flow are visualized at any time. Our correlation algorithm is robust enough so that good results are obtained both for sparse and dense representations. Since our streamline placement algorithm gives a total control on the density of the representation, we have a full control of the image density at any time.

The remaining of this paper is organized as follow. Section 2 describes our streamline placement algorithm presented in [4]. Section 3 explains how to encode motion along streamlines and describes our streamline correlation algorithm in details. Section 5 explains how streamline textures are correlated together. In Section 6 results are presented and we compare our method with previous approaches. Section 7 concludes and gives directions for future research.

2. Streamline Placement

Streamlines are a good way to visualize a vector field. By covering a domain on which a vector field is defined by a set of streamlines, we are able to show the global flow structure, its degree of turbulence, and all critical points, such as sinks and saddle points. To make this information accurate and smooth it is necessary to select only a subset of all possible streamlines, according to some criterion. We can show that the image quality is a function of the streamline length. For a sparse representation it would be difficult to understand the real topology of the flow if the streamlines are too short (see Figure 2 left) because the eye tend to follow the path described by the streamlines in order to understand the structure of the flow. Indeed streamlines' ends tend to act as artifacts and to disturb the observer. In case of a dense representation, pixels along a streamline are correlated together, thus the longer the streamline, the higher degree of correlation.

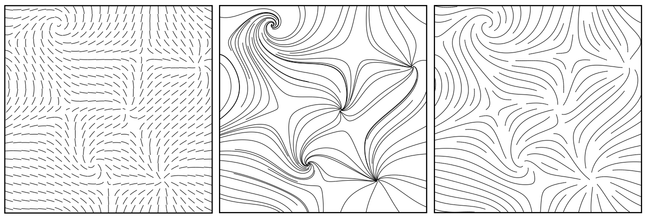


Figure 2: *Left and middle figures have been obtained by placing seed points at the intersection of a regular grid. Left: short streamlines. Middle: long streamlines. Right: image obtained by our streamline placement algorithm.*

Another important criterion to evaluate the quality of such an image is the distribution of white and black pixels in the image. Regions with higher density of black pixels tend to appear as more important, concentrating more flow features and catching the eye, while this is generally a visualization artifact (see Figure 2 middle). To address this issue Turk and Banks proposed an image-guided streamline placement algorithm based on a progressive refinement scheme [10]. In a previous paper we proposed a more efficient and direct approach to obtain similar results [4] (see Figure 2 right), and recently Mao et al. extended Turk's method to deal with curvilinear grids [7].

The method proposed in this paper for time-varying vector fields uses our streamline placement algorithm, which is described in details in [4]. The following subsections recall the main steps of the former method.

2.1. Streamline Integration and Density Control

A streamline is defined as a sequence of so-called sample points. Each sample point is obtained by integrating its position as a function of the position of the previous sample point. Sample points are computed so that they are equally spaced in space. Once a seed point has been determined (see Section 2.2), integration occurs backward and forward until either the separating distance with the closest streamline falls under a fixed threshold, or the boundary of the image has been reached, or a source or a sink point has been encountered. To speed up the process, instead of really computing the separating distance between a candidate sample point with all streamlines, we use a grid as an acceleration structure (see [4]). In order to maintain the desired level of correlation of the image pixels, only the streamlines whose length is above a fixed threshold are considered as valid.

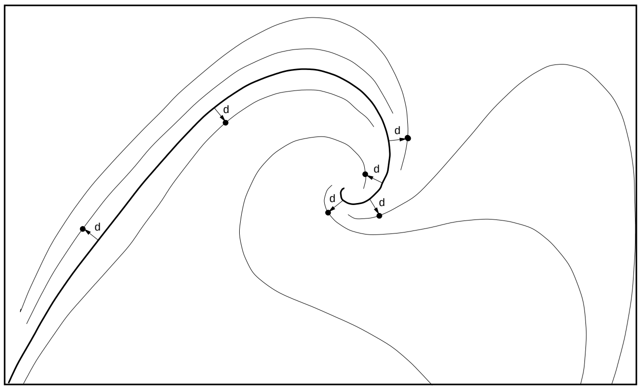


Figure 3: *Streamlines are derived from the first (drawn as thick here) one by selecting seed points at the separating distance d.*

2.2. Seed Point Selection and Domain Coverage

Our algorithm starts by selecting an initial seed point from which a first streamline is integrated and put into a stack. Then the main loop of the algorithm is as follow. A streamline is extracted from the stack and all seed points that are valid at the separating distance from the current streamline are determined. From each of these seed points a new streamline is computed (see Figure 3) and if the streamline is valid it is pushed into the stack. The algorithm finishes when the stack becomes empty. A detailed algorithm is given in [4]. It has to be noted that, as compared to streaklines, pathlines, and timelines, we obtain a complete and uniform coverage of the image, thus information about the vector field is available at any point. For unsteady flows, in order to achieve good correlation between successive frames, a new seed point selection algorithm has been designed.

3. Encoding Motion Along Streamlines

In addition to the direction, orientation is an important feature to help understanding the behavior of the flow. On a static image orientation can be visualized by mapping an oriented texture onto the streamline according to the orientation of the flow, such as a saw-teeth texture for instance. By stretching the texture, the velocity of the flow can be visualized, although results obtained with this technique for static images are not convincing. Visualizing the velocity properly requires to animate the flow, which can be achieved by moving a texture along each streamline [11].

In case of a steady flow the same set of streamlines can be used for every time step and only the texture correlation issue has to be addressed. In [5] we proposed the Motion Map, an original data structure and algorithm to compute a dense set of streamlines with textures mapped on them. By using a cyclical set of one-dimensional textures, which are shifted from one frame to the next, we are able to produce smooth animations of the flow.

In case of a time-varying vector field, the structure of the flow changes at each time step, hence the set of streamlines used for the previous frame is no longer valid and new streamlines have to be computed. If we simply compute a new set of streamlines, without any correlation between streamlines at different time steps, it is easy to figure that we would not obtain a smooth animation but rather a superposition of blinking effects that would make it impossible to understand the behavior of the flow. Thus it is necessary to highly correlate streamlines at consecutive time steps together.

In this paper we present an original method to compute a sequence of correlated streamline sets that are able to properly depict the behavior of the flow over time. The correlation is performed at the streamline level and both the shapes of the streamlines and the textures that are mapped on them are correlated.

4. Streamline Correlation over Time

When a steady flow field is animated using the Motion Map method, two important features are visualized: the structure of the flow, i.e. the direction at any point, and the orientation, by the animated streamline textures. All the motion that appears on the image is due to the *real* motion of the flow, that is nothing is moving but the flow itself, and the flow is moving in the right direction. When dealing with an unsteady vector field, the streamline sets at two consecutive time steps are different, even if they have been correlated together. During the animation, replacing streamlines of a time step by corresponding streamlines of the next time step will give the illusion that streamlines are moving in a direction orthogonal to the direction of the flow. If we deal with quality of the visualization, all motion appearing during the animation, except those due to the texture motion along streamlines, which depicts the real motion of the flow, should be considered as a visualization artifact and is a potential source of misunderstanding for the observer. For this reason we have concentrated our efforts in minimizing the impact of those effects on the image quality. Following a set of observations we have made on several image series, we have identified three main sources of motion artifacts, which are listed below by order of increasing impact on the quality of the visualization, in the sense of the ability for an observer to understand the behavior of the flow:

- big shape changes between 2 *corresponding* streamlines in two consecutive frames (see Section 4.1),
- disappearance of streamlines,
- appearance of new (uncorrelated) streamlines.

For a single time step the number of streamlines that could be drawn is potentially infinite and we reach the desired density by selecting a subset of those streamlines. The goal of our correlation algorithm is to select the best set of streamlines according to a correlation criterion.

4.1. Algorithm Overview

To find the best set of streamlines, we use a so-called *feed forward algorithm*, where streamlines at time step t are used to compute streamlines at time step t+1. The first frame is generated using the algorithm described in Section 2. For subsequent frames our algorithm processes in two steps. First we compute all streamlines at time step t+1 that match a streamline at time step t according to a defined criterion, which measures the difference between two streamlines in terms of position and shape. In the remaining of this document we will call *reference streamline* the streamline at time step t and *corresponding streamline* the streamline at time t+1 that matches the reference streamline. The second step of our algorithm consists in adding new streamlines to fill the image in order to obtain a uniform representation with the desired density. These new streamlines are computed at time step t+1. Figure 4 shows the different stages of our algorithm.

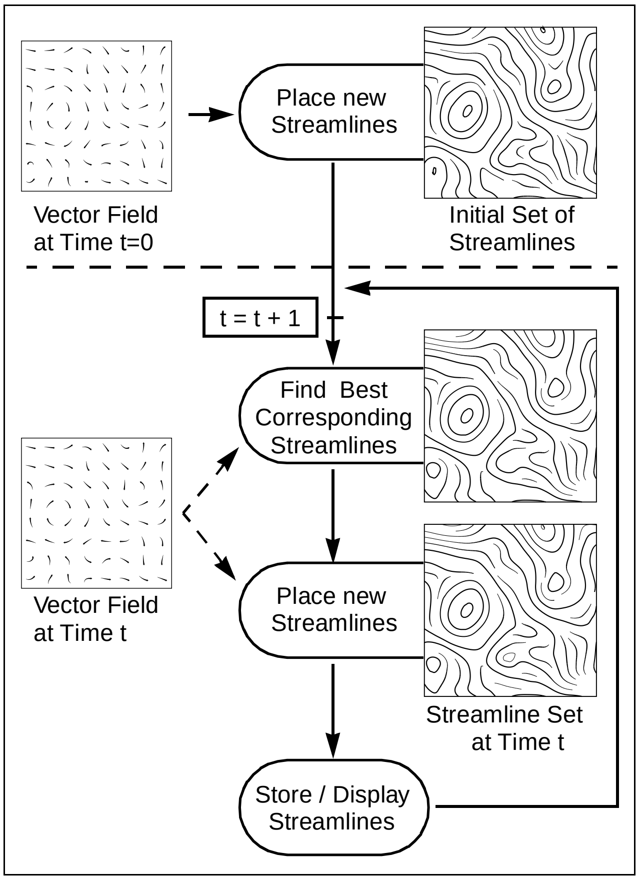
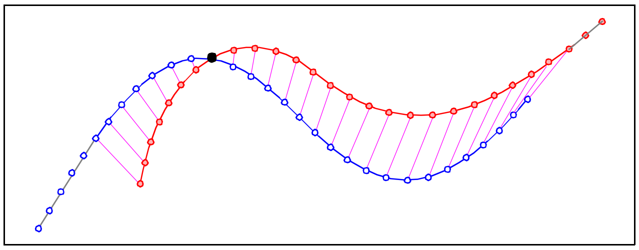


Figure 4: *Feed forward streamline placement for unsteady flows.*

4.2. Best Corresponding Streamline Selection

For each streamline at time step t we determine the corresponding streamline at time step t+1. This is achieved by computing a so called *candidate streamline* at time step t+1 for each sample point of the reference streamline. The streamline is integrated according to the desired image density, that is we check the separating distance between this streamline and every other streamline in the current frame. As for the first frame only streamlines whose length is above a fixed threshold are kept as candidates. Our experiences showed that a good threshold is a function of the integration step and the flow structure since a trade-off has to be found between having long streamlines, which increases the spatial correlation between pixels of the image, and having a good correlation between corresponding streamlines in successive frames. In practice we use a threshold of 10 sample points. Once all candidate streamlines have been computed, a correlation criterion between each candidate and the reference streamline is evaluated and the streamline with the highest score is elected as the corresponding streamline.



● Corresponding Sample Point

Reference and Candidate Streamlines Corresponding Portions

$$correlation = \frac{\sum_{(sp_1,sp_2)\in C(S_1,S_2)} dist(sp_1,sp_2)}{Card(C(S_1,S_2))}$$

where $C(S_1,S_2)$ is the corresponding portion of both streamlines.

Figure 5: *Corresponding portions are defined in order to evaluate the corresponding criterion between reference and candidate streamlines.*

The correlation criterion has been defined as the average distance between all pairs of corresponding sample points in the *corresponding portion* of each streamline. The corresponding portion is the portion of each streamline so that for each sample point a corresponding sample point exists in the other streamline. To determine the corresponding portion we start from the common sample point of both streamlines (remember that the streamline at the current time step has been generated by integrating backward and forward from a sample point of the reference streamline) and then sample points on both streamlines are associated to each other in order in both directions (see Figure 5). This criterion allows us to consider the longest corresponding portions of both streamlines. Streamlines for which their portions tend to go away from the reference streamline will obtain lower scores while those that keep close to their reference streamline will obtain higher scores. Once the best corresponding streamline has been determined, it is inserted in the list of streamlines of the current frame.

The order used for processing reference streamlines is the same at each time step, so that a corresponding streamline will be constructed in the same conditions as its reference streamline. For instance if some streamlines were already created at the time the reference streamline was built, their corresponding streamlines will be created before the current corresponding streamline be computed. In this way the constraints under which a corresponding streamline is constructed are the same as for the construction of its reference streamline. By processing in this way, the overall correlation scores between streamlines have increased substantially.

4.3. Image Completion with New Streamlines

The second step of the algorithm consists in completing the image with new streamlines in order to ensure an uniform image density. This step is necessary because the first step ensures that the separating distance between streamlines does not fall below the desired distance but it does not ensure an uniform coverage of the image, as for the method described in Section 2. Thus, especially if important changes have occurred in the structure of the flow, some areas in the image may not have the desired density.

To complete the image we use the algorithm described in Section 2, except that, instead of an arbitrary initial streamline, the stack is initialized with the set of streamlines generated at step 1. The new streamlines generated at step 2 are inserted at the end of the streamline list, in order for the future corresponding streamlines to be generated under the same constraints as for their reference streamlines. As a consequence the rank of a streamline in the list is a function of its relative age in the list, older streamlines being located at the beginning of the list. Remember we process reference streamlines from the beginning of the list and that only valid streamlines, that is streamlines that are longer enough, are considered valid. It is easy to understand that the chance for a streamline to be declared as valid decreases as the number of already created streamlines in the image increases. Hence reference streamlines at the beginning of the list will more likely give rise to valid corresponding streamlines, since they are processed at the beginning, when the image contains a small number of streamlines.

At the beginning of Section 4 we explained that an important streamline shape modification was visually less disturbing than the appearance or disappearance of a streamline. By ordering reference streamlines from the oldest to the newest, we increase the probability of long life for most of the streamlines. The results obtained have shown that less than 5% of new streamlines are created for each frame in average. Indeed those streamlines tend to be shorter than those created at the first step because they are generated when more streamlines have been already drawn in this image. It means that the number of pixels covered by new streamlines is far below 5% of the number of black pixels in the image. As a consequence the effect of the appearance of new streamlines is minimized. Moreover we propose in Section 4.4 a technique to attenuate the visual effect of streamline appearance.

4.4. Improvements of Animation Quality

Although the correlating algorithm described above produces animations of good quality, a couple of aliasing effects remain. Some techniques are proposed here to enhance the smoothness of the animation.

Priority to Circular Streamlines

Circular streamlines are important features in an image because they show critical points, which are generally of significant importance for analyzing the underlying dynamic system. For instance in meteorology, circular streamlines can represent anticyclone or atmospheric depression centers. In order to keep track of these features during the animation it is important to avoid them appearing and disappearing several times but rather just to update their shapes from one frame to the next. For this reason we have modified the ordering algorithm, all circular streamlines being located at the beginning of the list at any time. This is achieved by detecting circular streamlines at the end of step 2, which are immediately moved to the beginning of the list. Circular streamlines are detected during the construction of the streamline by performing an additional test, which consists in evaluating the distance between the current sample point and the first sample point of the streamline. If this distance falls under a certain threshold (actually d/2 where d is the separating distance), the construction process is stopped and the streamline is declared as circular.

Tapering Effect

For sparse representations of vector fields with streamlines, it is better to draw anti-aliassed streamlines, which implies to draw streamlines whose thickness is more than one pixel. The odd effect is that streamlines' ends tend to debase the overall image quality. Tapering for streamlines is a technique introduced in [10] to attenuate the visual effect of streamlines' ends. It consists in progressively decreasing the streamline thickness as we go closer to one of its extremities. This technique is an important factor of the improvement of streamline-based image quality. While in [10] tapering requires a post-processing step, it is part of our streamline generation algorithm.

Progressively Increasing Thickness

In order to attenuate the flickering effect due to the appearance of new streamlines, the thickness of the streamline is a function of its age. A new streamline is always created with a thickness of 1 and those thickness will grow up until it reaches the *adult thickness*.

Train Effect

As for the visualization of small objects, such as particles, for which a train effect is used to keep track of the object from one frame to the next, train effect can be enabled to increase the correlation between successive frames. Our experience showed that this effect is really helpful to track flow features during the animation. At each time step, the streamlines of a couple of frames are displayed together, each streamline being drawn with a color intensity computed as a function of its age (recent frames being drawn with a higher intensity). Train effect can be used at any density. It is important to note that when train effect is enabled, streamlines may cut each other if they have been computed at different time steps.

5. Correlation of Streamline Textures

A one-dimensional texture is mapped onto each streamline. Visualization of the orientation of the flow is achieved by moving the texture along the streamline.

5.1. Moving Textures

The quality of the visualization can be assessed as the ability for an observer to keep track of the textures that are moving on the streamlines from one frame to the next. Thus textures on corresponding streamlines at successive time steps should be placed so that, during the animation, an illusion that the flow is moving in the direction described by the streamline will appear. This is achieved by using a cyclical set of textures, each one being a shifted version of the other, as shown on Figure 6. The same texture is used for all streamlines of a given frame. In order to obtain the animation, we shift the texture at every time step. Each texture is cyclical, that is the first and last pixels have the same value. This property allows us to duplicate the texture on a streamline while ensuring continuity between consecutive pixels. In order to save memory we store only one version of the texture and a pointer is used to mark the beginning of the texture.

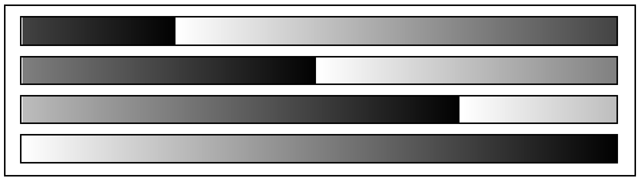


Figure 6: *Example of a cyclical set of textures that can be used to add motion to streamlines.*

For a steady flow field the same set of streamlines is used for all frames and hence we just have to shift the texture for each time step (see [5]). For an unsteady flow a streamline does not longer exist at the next time step but rather we use the corresponding streamline, which differs from its reference streamline in terms of position, shape, and length. If we would map the texture of the reference streamline onto the corresponding streamline directly, the texture would be stretched at some places, compressed at some others. Thus it would be difficult to keep track of the texture during the animation. In order to correlate textures properly we need to decompose each streamline in a sequence of streamline segments that we call *texture supports*. The purpose of texture supports is to have supports of about the same length for texture mapping in order to avoid stretching and compressing effects of great amplitude. Actually we define a minimum and a maximum value for texture support lengths as 0.5 x Ls and 1.5 x Ls respectively, where Ls is the mean support length. A good value for Ls is a function of the number of different colors in the one-dimensional texture, common values being 10 or 15 sample points. The texture is mapped onto each support independently.

5.2. Computation of Texture Supports

Texture supports are defined as a set of successive sample points (actually the sample points of the streamline to which they are related). We call *control points*, resp. reference control points and corresponding control points, the sample points that belong to two supports. A support is defined by two control points and a list of interior sample points. To correlate two streamline textures together it is necessary to make a correspondence between the texture supports of both streamlines. For this reason most of the texture supports of the corresponding streamline are computed from the texture supports of the reference streamline. To achieve this, for every reference control point, a corresponding control point is determined. This corresponding control point is the closest sample point of the corresponding streamline. In order to keep a good correlation it is necessary for the reference control point and the corresponding control point to be close to each other. For this reason only sample points in the neighborhood of the reference control point are considered as candidate control points. It means that, after this step, some reference control points do not have any corresponding control point and hence the supports defined by the corresponding control points are not necessary the same length. In a second step we make all supports about the same length. This is achieved by dividing supports that are too long in several supports of the right length and by collapsing too short supports.

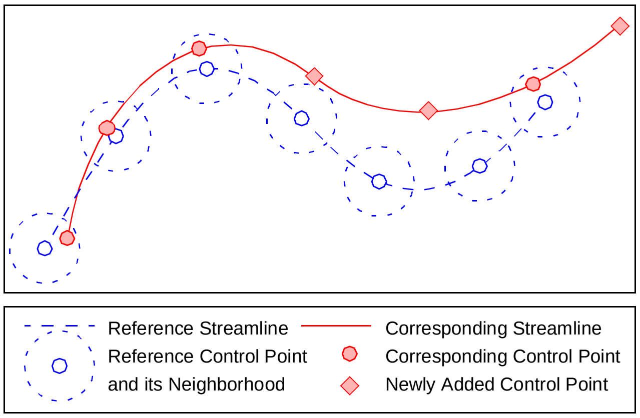


Figure 7: *Computation of texture supports.*

Figure 7 shows a reference streamline, its control points and their neighborhoods, and a corresponding streamline and its control points. On the corresponding streamline, circle control points have been determined from the reference control points, while square control points have been created in the second step. It should be noted that inserting or collapsing texture supports decreases the correlation degree between textures at different time steps, but this modification is local and only a few supports are generated in this way. When all texture supports of the corresponding streamline have been computed, the current texture is mapped independently onto each support.

The algorithm for correlating textures of corresponding streamlines is as follows:

```
For each reference control point Do
  Determine a corresponding control point if any
EndFor
For each support in the corresponding streamline Do
  Compute support length
  If support length is too long Then
    Divide support in as many supports as needed
  Else If support is too short Then
    Collapse support with the shortest adjacent
support
  EndIf
EndFor
For each corresponding support Do
  For each sample point of the support Do
    Compute texture coordinate in current texture
    Set sample point color accordingly
  EndFor
EndFor
```

For the streamlines created at the current time step (see Section 4.1), no correspondence with any reference streamline exists. In this case the streamline is partitioned in as many supports as necessary and the current texture is mapped onto every support.

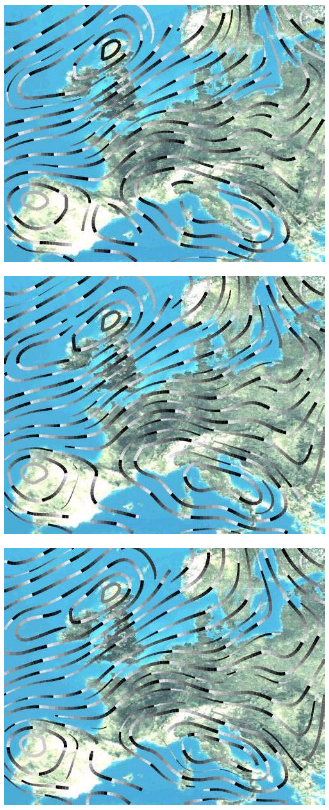


Figure 8: *Visualization of wind speed predictions over Europe. In order to show the direction of the flow, streamlines are shaded with an oriented one-dimensional texture.*

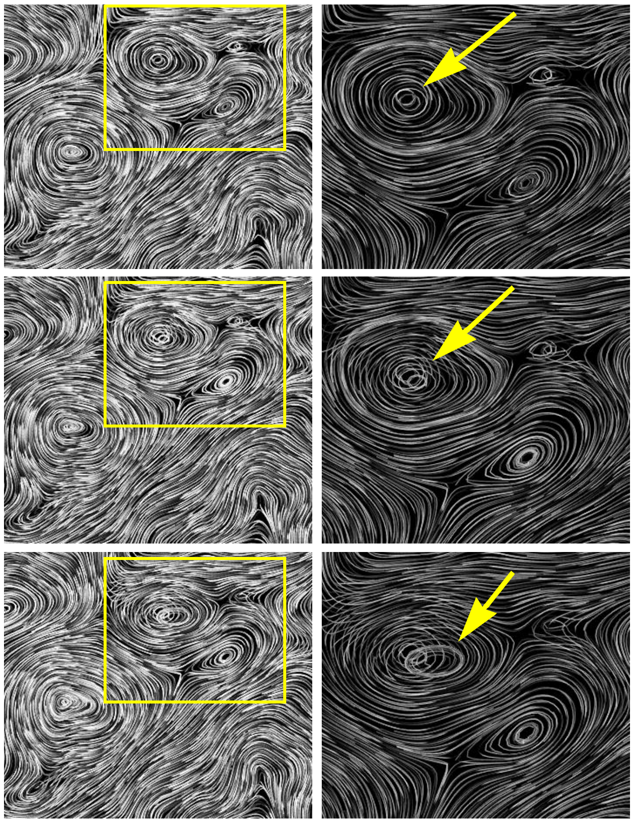


Figure 9: *Three consecutive frames of a time-varying vector field at two different scales.*

6. Results and Discussion

The animations produced by our method allow an easy tracking of the features of the flow while depicting information on the vector field with a constant density everywhere in the image. Figure 8 and Figure 9 show 2 different visualizations, for sparse and dense representations respectively. In both cases 3 consecutive frames of the animation are shown. Now let us remark that, in the case of time-varying vector fields, streamlines do not correspond to the paths that would be followed by a massless particle released in the flow. Instead they are just instantaneous virtual trajectories aimed at understanding the flow topology and its evolutions. Thus the images of Figure 8 and Figure 9 show the real topology of the flow at a given time step.

It is often useful to be able to display an additional quantity, for instance pressure, temperature, or vorticity. This is easily achieved with our method since we work with a true colors visual, so that a texture can be defined as any sequence of colors. In particular we can define a texture as a sequence of intensities or saturations and, at the time the texture is mapped onto a streamline, the hue can be derived from the value of an additional scalar quantity at the current point.

Unlike the methods based on pathlines, our approach does not require to have the complete vector field for several time steps in memory. Actually only the vector field for the current time step is required to compute the corresponding frame, in addition to the set of streamlines computed for the previous frame. This is an important feature since time-varying data are generally memory consuming. Computation times obtained for an image of 512x512 pixels on a SGI O2 R5000SC at 180Mz range from 5 seconds per frame for sparse representations to 30 seconds per frame for dense representations. As compared to LIC-based approaches for visualization of unsteady vector fields, our solution is more effective and achieves better visual results.

7. Conclusion

We have proposed an original approach to the visualization of time-varying 2D vector fields. While previous approaches were based on pathlines and try to correlate successive images at the pixel level, our method consists in correlating instantaneous visualizations of the vector field at the streamline level. The movement of the flow is obtained by moving a one-dimensional texture along the streamlines. In order to obtain a smooth animation, both shapes and textures of the streamline pairs at different time steps are correlated together. Using an accurate streamline placement algorithm, our method is able to produce animated sequences of arbitrary density, covering the field of representations from sparse to dense. Indeed the image density can be controlled easily by setting the separating distance between streamlines. Our placement algorithm ensures an uniform coverage of the entire image and does not require any further knowledge about the structure of the vector field.

The spectrum of applications of this work is wide and numerous scientific areas should be able to take benefit from this method. For instance sparse animations could be used to produce animated cartography, for the widespread dissemination of meteorological information through the Web or for educational purpose. Dense representations are useful when a high degree of accuracy is required and could be used for the purpose of analyzing time-varying simulations in aeronautics for instance.

Future research should address the generalization of this approach to 3D, which raises new issues for the placement of streamlines and for texture mapping. In addition visualizing 3D vector fields raises perceptual issues that should be addressed. There is also a need for a more accurate corresponding criterion between streamlines, particularly a mathematically defined criterion based on curve matching should be investigated.

Acknowledgements

We would like to thank Robert van Liere and Willem de Leeuw for the vector field data sets for wind speed prediction over Europe.

References

1. Cabral, B. and L. Leedom. Imaging Vector Fields Using Line Convolution. In Computer Graphics (*Proceedings of SIGGRAPH'93)*, pages 263-272. ACM Press.

2. Chédot, C. and W. Lefer. Multi-modal Flow Animation with the Motion Map. In C. Wittenbrink and A. Varshney, Editors (*Proc. of IEEE Visualization'97 Late Breaking Hot Topics*, pages 17-20), October 19-24, 1997.

3. Forsell, L.K. 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), pages 133-141, 1995.

4. Jobard, B. and W. Lefer. Creating Evenly-Spaced Streamlines of Arbitrary Density. In W. Lefer and M. Grave, Editors (*Proceedings of the 8th Eurographics Workshop on Visualization in Scientific Computing'97*, April 28-30, 1997), pages 43-55, Springer-Wien-NewYork.

5. Jobard, B. and W. Lefer. The Motion Map: Efficient Computation of Steady Flow Animations. In R. Yagel and H. Hagen, Editors (*Proceedings of IEEE Visualization'97*, October 19-24, 1997), pages 323-328, IEEE Press.

6. Lane, D.A. Visualization of Time-Dependant Flow Fields. In G.M. Nielson and R. Dan Bergeron, Editors (*Proceedings of IEEE Visualization'93*, October 25-29, 1993), pages 32-38, IEEE Press.

7. Mao, X., Y. Hatanaka, H. Higashida and A. Imamiya. Image-Guided Streamline Placement on Curvilinear Grid Surfaces. In D. Ebert, H. Rushmeier and H. Hagen, Editors (*Proceedings of IEEE Visualization'98*, October 18-23, 1998), pages 135-142, IEEE Press.

8. Shen, H.W. and D.L. Kao. UFLIC: a Line Integral Convolution Algorithm for Visualizing Unsteady Flows. In R. Yagel and H. Hagen, Editors (*Proceedings of IEEE Visualization'97*, October 19-24, 1997), pages 317-322, IEEE Press.

9. Stalling, D. and H-C. Hege. Fast and Resolution Independent Line Integral Convolution. Computer Graphics Annual Conference Series (*Proceedings of SIGGRAPH'95*, August 6-11, 1995), pages 249-258, ACM Press.

10. Turk, G. and D. Banks. Image-Guided Streamline Placement. Computer Graphics Annual Conference Series (*Proceedings of SIGGRAPH'96*, August 4-9, 1996), pages 453-460, ACM Press.

11. van Gelder, A. and J. Wilhelms. Interactive Animated Visualization of Flow Fields. Proceedings of the Workshop on Volume Visualization, pages 47-54, ACM Press, 1992.

12. van Wijk, J. Spot Noise-Texture Synthesis for Data Visualization. Computer Graphics 25(4) (*Proccedings of SIGGRAPH'91*, July 28 - August 2, 1991), pages 309-318, ACM Press.