On Time-Varying Flow Fields: a streakline-based visualization method

†Andrea Sanna †Bartolomeo Montrucchio and ‡Renzo Arina

†Dipartimento di Automatica e Informatica,
‡Dipartimento di Ingegneria Aeronautica e Spaziale,
Politecnico di Torino, corso Duca degli Abruzzi 24, 10129 Torino (Italy)

Abstract
The visualization of unsteady flows is an attractive field of research and texture-based techniques seem to provide satisfactory results. In this paper we propose a texture-based method which follows streaklines in order to produce an effective visualization of time-dependent phenomena. Our approach allows to overcome the lack of temporal coherence due to instantaneous methods which depict each frame of an animation without considering the temporal correlation with the previous time steps.

1. Introduction

Vector field data arise from computer simulations in a variety of disciplines, such as CFD (Computational Fluid Dynamics), climate modelling and so on. An adequate visualization is a challenging problem due to the difficulties in finding a suitable graphic way to represent and display vector data on a monitor display.

Of particular interest are the visualizations of time-dependent phenomena; in these cases, vector data are produced for a set of time steps. To visualize time-varying data, two kinds of approaches can be employed:

- *Instantaneous method*. Each time step is considered separately from the others and a graphical icon representing the vector field is produced. The discrete time steps are animated together to obtain the resulting visualization sequence.
- *Time-dependent method*. A graphical icon for a frame is obtained considering the correlation of the current time step with the previous ones.

Instantaneous methods are mainly based on streamlines which represent field lines tangent to the vector field. Unfortunately, these techniques often suffer from the problem of lacking coherence between frames since the streamlines shown at a given instance in time do not correspond to the paths that the particles have travelled inside an unsteady flow field. Time-dependent methods can overcome this problem. In this paper we propose a technique which follows particle traces (in particular streaklines) in order to produce a texture for each time step of an animation. A set of insertion points (an insertion point is a location from where the particles are released) is kept constant for all time steps, and at each frame a new particle is released from every insertion point (also called seed location). In the meanwhile, the positions of the particles previously emitted are up to date. In this way, each frame of the animation maintains the coherence with the previous ones and the resulting sequence can effectively show the evolution of the field in time.

The paper is organized as follows: in section 2 the main visualization methodologies are briefly reviewed while in section 3 the proposed algorithm is shown in the details; finally, an application example and remarks can be found in section 4.

2. Background and related works

An effective way to visualize unsteady flows is to compute particle traces [6]. In steady flows a streamline is a field line tangent to the vector field. For unsteady flows, which are of interest for this paper, three types of particle traces can be computed:

- *Pathline* which shows the trajectory of a single particle released from an insertion point.
- *Streakline* which is a line joining the positions, at an instant in time, of all particles released from the same insertion point.

- *Timeline* which is a line joining the positions of particles released at the same instant in time from different insertion points.

In an instantaneous flow field, streamlines, pathlines, and streaklines are identical [7] but it has been proved that streaklines and timelines can reveal information not visualizable by pathlines for unsteady flows [45]. Unfortunately, particle tracing techniques depend critically on the placement of the insertion points; depending on their placement currents in the data field can be missed. Another approach for visualizing vector field data is based on texture synthesis. In particular, the LIC (Line Integral Convolution) technique revealed to be an effective and elegant method to visualize vector field data [1]. The LIC method takes 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. The resulting intensity values of the LIC pixels along each streamline are strongly correlated so the directional patterns of the flow fields can be easily visualized. The LIC algorithm as presented in [1] is applicable only to vector fields over regular 2D Cartesian grids; moreover, the LIC shows only the direction of the flow but not its orientation and it is not adequate to visualize animation of unsteady flows. In order to solve these drawbacks several works have been published in the literature: OLIC (Oriented Line Integral Convolution) [10] was designed to show the orientation of a flow by sparse textures, Forssell and Cohen proposed in [2] an extension of LIC able to consider curvilinear grids, a 3D LIC was presented in [3], and a parallel convolution algorithm based on pathlines was proposed in [9]. In particular, in [9] has been outlined as all the previous techniques which follow traces of streamlines cannot be effectively employed to visualize unsteady flows since the approach of computing the LIC (or an its extended version) for each time step and then animating the results together may suffer of lacking coherence between animation frames. This problem can be tackled following traces of pathlines as in [9] or, more effectively [45], traces of streaklines as presented in this paper.

3. The algorithm

Since streaklines can better characterize the evolution of flow fields in time, our goal is to develop an algorithm able to depict the particle traces as a texture.

In order to do this a set of insertion points must be selected; since we produce a texture for each time step of the flow field evolution, the insertion points will correspond to certain pixels of the output texture. The choice of the seed locations must be as much as possible uniform in order to inspect each part of the field. A random positioning of these locations can be obtained, for instance, by a Sobol's distribution. At each time step a particle has to be released from every insertion point; the locations of the insertion points are unchanged for all time steps.

```
void streakline (void)
{
   int ind, ind2, x, y;
   for(ind = 1;ind <= num_frame; ind++)
   {
      Read_vector_field( );
      for(ind2=1;ind2<=num_ins_point;ind2++)
      {
         Sobol_distribution(&x, &y);
         Add_particle_to(x,y);
         Up_to_date_position(x,y);
      }
      Compute_texture_out( );
      Display_frame( );
   }
}
```

Figure 1: *C-like pseudo code of the algorithm.*

The length of the streaklines can be set from the user at the beginning of a visualization. For each time step a new particle is released from the insertion points and the positions of the particles previously emitted are up to date according their position inside the vector field. When a particle goes outside from the image borders, i.e. exits from the vector field, it is marked as an invalid particle and its position is not more considered. Moreover, when the streaklines have reached the length set from the user, a new emission of particles means that the "oldest" ones must not be more considered. This means that, after an initial transitory, if the number of insertion points is denoted by N and the streakline length is L, the complexity of the algorithm is $O(N \cdot L)$.

After up to dating the position of all particles, the pixels affected by each streakline must be computed. An initial grey tone is randomly assigned to the pixel corresponding to the particle closest in time to the insertion point. The next particle of the streakline is searched and the corresponding pixel determined; a grey tone equal to the previous one plus a constant value is assigned to the pixel as long as a color has not been previously computed for it. This procedure continues until the end of the streakline is not reached. This approach allows to denote the orientation of a streakline, even if this is not a so important topic as for instantaneous methods such as LIC, OLIC and so on.

In Fig. 1 a C-like pseudo code of the algorithm is shown. A simple two-dimensional array can be used in order to effectively manage the particles. The index of row denotes the insertion points while the index of column indicates the time steps. Each element of the array can be a record which stores the information about a particle released at a time step. When the number of particles released from an insertion point is equal to the streakline length the index of column is set to

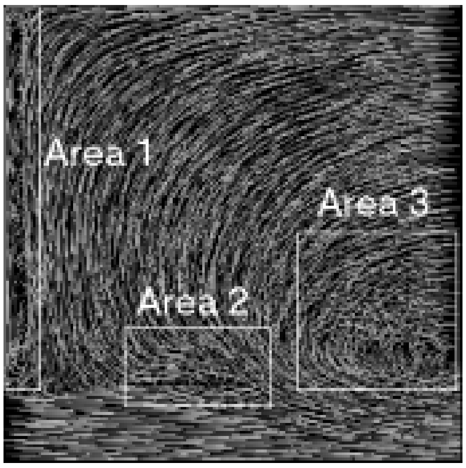

Figure 2: *This image has been produced following streaklines.*

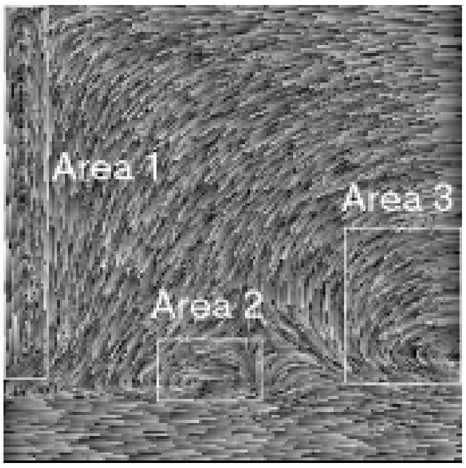

Figure 3: *This image has been produced following streamlines.*

zero, in this way the information about the new particle will overwrite the oldest one.

4. Results and remarks

In this section an application example of our algorithm is presented; in particular, we propose a sequence obtained by a numerical simulation. The numerical simulation in asymmetric geometry represents the early stages of the flow of an incompressible fluid continually ejected through a circular orifice into a quiescent fluid.

The process consists into the formation of a large vortical structure along the wall which grows in time and subsequently lift off from the wall itself forming a vortex ring. Behind it, the interface between the quiescent fluid and the developing jet undergoes to flow instabilities creating periodic vortical structures. The sequence is composed by 84 time steps; a comparison between the proposed algorithm and an instantaneous method [8] which can reveal orientation and direction of the flow is presented.

The analysis of a single frame does not allow to completely evaluate the differences between the two approaches since time coherence between frames cannot be appreciated, but significant differences can be noted. In Fig. 2 and Fig. 3 the 65th frame is shown for both techniques; Fig. 2 has been produced by the proposed approach while Fig. 3 has been obtained by an instantaneous method.

Although the instantaneous method reveals the periodic flow instabilities along the jet interface (areas 2 and 3) it cannot well depict the secondary vorticity along the wall (area 1). The same effect can be noted by comparing the 77th frame for both approaches (see Fig. 4 and Fig. 5). We have tested our software on a 433 MHz Alpha Digital workstation with 640 MB of RAM; the 84 frames (282x282 pixels) have

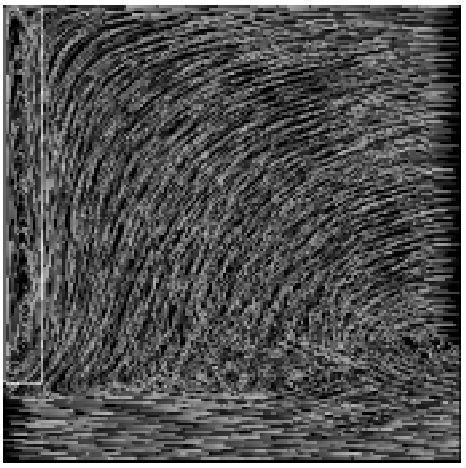

Figure 4: *This image has been produced following streaklines.*

been computed and displayed in 6.6 seconds for a frame rate of about 13 frames per second.

Streaklines can reveal characteristics of a flow otherwise not detectable; on the other hand, the proposed technique must be further improved in order to enhance the comprehension of the vector field structure which may seem not clear if a single frame is analyzed singularly.

5. Conclusion and future work

An algorithm for visualizing time-varying vector fields has been presented. The proposed approach follows the streaklines which can better characterize the evolution in time showing phenomena otherwise not detectable following streamlines as well as pathlines.

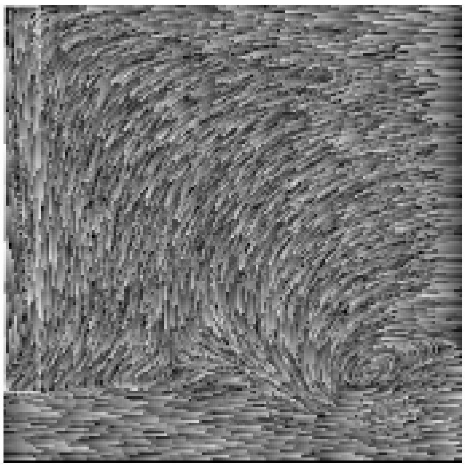

Figure 5: *This image has been produced following streamlines.*

The proposed method: does not require an input texture, its computational cost can be controlled by the number of insertion points, can be easily implemented, and it does not suffer of lacking of temporal coherence between frames.

Future work will be aimed to enhance the comprehension of the vector data since the graphical icons resulting following streaklines may sometimes confuse the user. In particular, streaklines can overlap easier than streamlines and the streakline orientation may be not clear if a frame of the sequence is analyzed singularly.

References

1. B. Cabral and L.C. Leedom. Imaging Vector Fields Using Line Integral Convolution. *ACM Computer Graphics (Proc. of SIGGRAPH '93)*, 27:262–270, 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. V. Interrante and C. Grosch. Visualizing 3D Flow. *IEEE Computer Graphics & Applications*, 18(4):49–53, 1998.
4. D.A. Lane. Visualization of Time-Dependent Flow Fields. *In IEEE Visualization'93 Proceedings*, 32–38, 1993.
5. D.A. Lane. Visualizing Time-Varying Phenomena in Numerical Simulations of Unsteady Flows. *AIAA 96-0048, 34th AIAA Aerospace Sciences Meeting*, Reno, Nevada, 1996.
6. G.M. Nielson, H. Hagen and H. Müller. *Scientific Visualization*. *IEEE* Computer Society, Los Alamitos, California, 1997.
7. H. Schlichting. *Boundary Layer-theory*. McGraw-Hill, New York, 1979.
8. A. Sanna, B. Montrucchio, R. Arina and L. Massasso. A 3D Fluid Flow Visualizer for Entry Level Computers. *In Proceedings of WSCG'99*, I249–I256, 1999.
9. Han-Wei Shen and D.L. Kao. A New Line Integral Convolution Algorithm for Visualizing Time-Varying Flow Fields. *IEEE Transaction on Visualization and Computer Graphics*, 4(2):98–108, 1998.
10. R. Wegenkittl, E. Gröller and W. Purgathofer. Animating Flowfields: Rendering of Oriented Line Integral Convolution. *In IEEE Visualization'97 Proceedings*, 15–21, 1997.