Visualization of Time-Dependent Flow Fields

David A. Lane

Computer Sciences Corporation
NASA Ames Research Center
M/S N258-6
Moffett Field, CA 94035

Abstract

Presently, there are very few visualization systems available for time-dependent flow fields. Although existing visualization systems for instantaneous flow fields may be used to view time-dependent flow fields at discrete points in time, the time variable is usually not considered in the visualization technique. We present a simple and effective approach for visualizing time-dependent flow fields using streaklines. A system was developed to demonstrate this approach. The system can process many time frames of flow fields without requiring that all the data be in memory simultaneously, and it also handles flow fields with moving grids. We have used the system to visualize streaklines from several large 3D time-dependent flow fields with moving grids. The system was able to provide useful insights to the physical phenomena in the flow fields.

1 Introduction

Flow visualization plays an important role in Computational Fluid Dynamics (CFD) simulations. There are several effective techniques for visualizing instantaneous flow fields: cutting planes, isosurfaces, and volume rendering are some of the techniques for visualizing instantaneous scalar fields. Visualization of scalar fields alone may not be sufficient in some flow analyses; vector fields are often visualized simultaneously. Several techniques have also been developed for visualizing instantaneous vector fields, and some of them are described in [7,10,14,19]. Most of these techniques are based on instantaneous streamlines. An instantaneous streamline is a curve which is tangent to the vector field at an instant in time [6]. Ribbons are instantaneous streamlines with width and rotational orientation based on the local vorticity. Stream tubes are formed by constructing a set of instantaneous streamlines through points on a closed curve. Stream surfaces are constructed in the same manner as stream tubes except the streamlines pass through a set of points on a line segment called a rake [15]. Finally, stream polygons consist of particles which are n-sided polygons oriented normal to the local vector [18]. These techniques are considered to be instantaneous visualization techniques because they are based on flow fields at some instant in time.

For time-dependent (unsteady) flow fields, dynamic visualization techniques can reveal several physical phenomena that sometimes cannot be seen in instantaneous visualization techniques. Pathlines and streaklines are two of the dynamic visualization techniques. A pathline shows the trajectory of a particle released from a given location. In an instantaneous flow field, pathlines and streamlines are identical. A streakline is a line joining the positions, at an instant in time, of all particles that have been released previously from a specified location [6]. Thus, a streakline through a given location is the line joining all the particles that have been released from that location. In hydrodynamics, streaklines are simulated by releasing hydrogen bubbles rapidly from a fixed location. Smoke injection and dye advection are other techniques commonly used.

Because streamlines are computed from instantaneous flow fields, only one time step of the flow data is needed in the calculation. However, pathlines and streaklines require several time steps of the flow data in the calculation. This requires more disk space and physical memory in the system where the visualization is performed. Some systems cannot provide unsteady flow visualization because of the lack of disk space and physical memory. In this paper, we present a real-world problem in which large 3D unsteady flow data sets are visualized using streaklines. First, two unsteady flow data sets are described. Then, the numerical methods for particle tracing in steady and unsteady flow are described. A system which implements

1070-2385/93 $3.00 © 1993 IEEE

the numerical model described was developed to compute particle traces in unsteady flow. The implementation of this system is discussed. Finally, several images that depict streaklines computed from the two unsteady flow data sets are shown.

2 Problem

Several large data sets of 3D unsteady flow fields are currently being studied at NASA Ames Research Center. This includes the Descending Delta Wing [5], which consists of 900 thousand grid points, and the V-22 Osprey tiltrotor aircraft [17], which consists of 1.3 million grid points. The geometry of these data sets is represented using Chimera grids. This type of grid allows an aircraft with complex geometry to be defined using several sub-blocks [2]. Both the Delta Wing and the V-22 tiltrotor data sets have moving grids; there is one grid file per time step for both data sets. The flow data, which contains the solutions of the Navier-Stokes equations, is stored in a number of solution files. There is one solution file per time step for each data set. Table 1 shows the disk space required for each time step of the two data sets.

Data Set	Grid File Size	Solution File Size
Delta Wing	16 MB	20 MB
V-22	23 MB	29 MB

Table 1. The disk space (in megabytes) required for each time step of the Descending Delta Wing and the V-22 tiltrotor.

In an unsteady flow simulation, solution files are generated for a number of time steps. The number of time steps can reach thousands in some simulations. For example, the Descending Delta Wing has 90,000 time steps and the V-22 tiltrotor has 1,450 time steps per rotor blade revolution. However, only a subset of the data could be visualized because of the current hardware limitation. For the Delta Wing, every 50th time step of the data was used. Three rotor blade revolutions of the V-22 tiltrotor were used and the data was sub-sampled at every 15th time step. Table 2 shows the disk space required to visualize the two data sets.

Data Set	# of TS Used	Disk Space Used
Delta Wing	1,800	64.8 GB
V-22	290	15.1 GB

Table 2. The number of time steps and the disk space (in gigabytes) used.

Currently, Plot3D [4] and FAST [1] are used to visualize these data sets. However, both Plot3D and FAST do not compute streaklines. For some analyses, it is desirable to compute streaklines that provide unsteady flow visualization. One of our challenges is to use as many time steps of flow data as possible in the calculation of streaklines. Another challenge is to compute streaklines in moving grids.

3 Related Work

There are several unsteady flow visualization systems. Plot4D [20] visualizes 3D unsteady flow by using variables in two-dimensional computational planes with time as the third coordinate. Thus, several time slices of a computational plane from the 3D flow data can be visualized. The Virtual Wind Tunnel (VWT) [3] allows interactive interrogation of unsteady flow fields using a stereo head-tracked display and a data glove. VWT allows the user to interactively investigate the flow data in a virtual environment. Although Plot4D and VWT are effective tools for interactive visualization of unsteady flow data, they require a preprocessing of the input data such that a number of time steps of flow data is stored in one input file. For very large unsteady flow data sets with thousands of time steps, it is sometimes impossible to preprocess a large number of time steps in the flow data. In [16], unsteady flow visualization is demonstrated on a massively parallel machine. The system saves particles traces at the end of each integration step, and the file is then later retrieved for visualization. The system does not allow adaptive time step in particle integration, which is often required when velocity changes rapidly between two consecutive time steps. Visual3 [13] is a system for visualizing steady and unsteady flow data. The system reads unstructured and structured grids. Visual3 can easily process many time steps because it does not require any preprocessing of the data. However, Visual3 runs locally on a workstation and the size of the flow data is limited. Recently, a distributed architecture for visualizing large unsteady flow data using multiple concurrent processors across the network was proposed in [12]. In [11], unsteady flow visualization is obtained by animating the results computed from a number of instantaneous visualization techniques. A discussion of two visualization packages used for their animation is also presented in [11]. A new model was proposed in [9] for unsteady flow visualization. This model uses "extracts" to express the flow data, where extracts are N-dimensional fields that can be used to generate graphics objects.

Though all of the systems mentioned above can visualize unsteady flow data, the size of the flow data is often limited and some systems do no provide adaptive time step in particle integration. Furthermore, many of them do not allow particle integration in flow fields with moving grids. In this paper, we present a system that computes streaklines from unsteady flow data. The system is different from the other systems mentioned earlier in that it can handle a very large number of time steps in the flow data, and it allows particle integration in flow fields with moving grids. The system also allows adaptive time steps in particle integration.

4 Numerical Model

This section describes the numerical methods for computing particle traces in steady and unsteady flow fields.

4.1 Steady Flow

Assume that the vector function $\vec{V}(p)$ is defined for all p in the domain D. The path of a particle at the point p can then be defined parametrically as

$$\frac{dp}{dt} = \vec{V}(p). \tag{1}$$

The particle path can be computed by numerically integrating (1). Depending on the desired accuracy, different integration methods can be used. A discussion on the accuracy of particle integration is presented in [8]. A simple scheme is the second-order Runge-Kutta integration with adaptive time steps. Let $k = 0$ and p_0 be the initial position of the particle. Then, for $p_k \in D$ and $\vec{V}(p_k) \neq 0$, let

$$p^* = p_k + S(p_k)\vec{V}(p_k),$$

$$p_{k+1} = p_k + S(p_k)(\vec{V}(p_k) + \vec{V}(p^*))/2, \text{ and}$$

$$k = k + 1, \tag{2}$$

where $S(p_k) = c||\vec{V}(p_k)||$ and $0 < c \leq 1$. The scalar factor c controls the step size of the particle. Small values of c yield small step sizes. Since the vector field $\vec{V}$ only has quantities at some discrete points in the grid, a trilinear interpolation is commonly used to interpolate $\vec{V}(p)$ when p does not coincide with a grid point.

4.2 Unsteady Flow

Assume that the vector function $\vec{V}(p,t)$ is defined for all $p \in D$ and $t \in [t_1, t_n]$, where n is the number of time steps in the unsteady flow. The path of a particle at the point p at time t is then defined as

$$\frac{dp}{dt} = \vec{V}(p,t). \tag{3}$$

Using the second-order Runge-Kutta scheme with adaptive time steps, Equation (3) can be numerically integrated as follows. Let $t = t_0$, $k = 0$, and p_0 be the initial position of the particle. Then, for $t < t_n$, $p_k \in D$, and $\vec{V}(p_k, t) \neq 0$, let

$$h = S(p_k), \qquad p^* = p_k + h\vec{V}(p_k, t),$$

$$p_{k+1} = p_k + h(\vec{V}(p_k, t) + \vec{V}(p^*, t + h))/2,$$

$$t = t + h, \text{ and } k = k + 1. \tag{4}$$

Since $\vec{V}(p,t)$ is known only at discrete points in the grid and at some time steps, if $t \neq t_i$ for $i = 1, \ldots, n$, then an interpolation in time is performed. If p does not coincide with a grid point, then a trilinear interpolation in space is also performed.

4.3 Moving Grid

Let the vector function $\vec{V}(p, t, g(p,t))$ be defined for all $p \in D$, $t \in [t_1, t_n]$, and $g(p,t) \in [G_1, G_n]$, where $g(p,t) = \{(x_u, y_v, z_w), \ldots, (x_{u+1}, y_{v+1}, z_{w+1})\}$ is the grid cell that p lies in at time t, G_i is the grid of the flow data at time t_i, $G_i \in D$, and n is the number of time steps in the flow data. The grid cell $g(p,t)$ is represented by the eight vertices of the cell and a search for the cell is sometimes required, especially when the grid is a function of time. If $t = t_i$, where $1 \leq i \leq n$, then $g(p,t)$ can be computed by searching for p in the current grid G_i. However, if $t \neq t_i$, for $i = 1, \ldots, n$, then interpolation in space is required. This can be done as follows. During particle integration, the computational coordinates of p, which is denoted by (ξ, η, ζ), is often known. The grid cell $g(p,t)$ can be computed by interpolating the grid cells that (ξ, η, ζ) lies in at time t_i and t_{i+1}, where $t_i \leq t \leq t_{i+1}$. Let $g(p, t_i)$ and $g(p, t_{i+1})$ be the grid cells that p lies in at time t_i and t_{i+1}, respectively, where $g(p, t_i) \in G_i$ and $g(p, t_{i+1}) \in G_{i+1}$. Then, $g(p,t) = (1 - \alpha)g(p, t_i) + \alpha g(p, t_{i+1})$, where $\alpha = (t - t_i)/(t_{i+1} - t_i)$. This will

require eight linear interpolations, one interpolation for each vertex of the grid cell. The quantities for p^* and p_{k+1} in (4) then become

$$p^* = p_k + h\vec{V}(p_k, t, g(p,t)) \quad \text{and}$$

$$\begin{aligned} p_{k+1} &= p_k + h(\vec{V}(p_k, t, g(p_k, t)) \\ &\quad + \vec{V}(p^*, t+h, g(p^*, t+h)))/2. \end{aligned} \tag{5}$$

5 Implementation

A system was developed to compute streamlines, pathlines, and streaklines using the numerical model described in the previous section. This system was developed using C, and the integration methods were implemented in FORTRAN. Plot3D's particle tracing library, which handles Chimera grids, was used to develop an unsteady particle tracing library. The Plot3D library computes particle traces in steady flow using the second-order Runge-Kutta adaptive time scheme described in (2). In order to compute particle traces in unsteady flow, new subroutines were written to implement the time-dependent integration given in (4). At any time during the integration, there are at most two consecutive time steps of flow data stored in memory. The flow data in memory is updated as soon as time is advanced to the next time step. The reason that two time steps are stored in memory is because during particle integration, interpolation between two consecutive time steps is required. In addition, if the flow field has moving grids, then the grid cells are interpolated in time.

For pathline calculation, particles are released from a set of points, which are referred to as the seed locations, at the first time step and the trajectories of the particles are computed using (4). In streakline calculation, new particles are released from the seed locations at each time step and the current position of all active particles are computed. The previous positions of the particles are not stored. If the grid is moving in time, then particles are released at new locations as the grid moves and (5) is used to compute the path of the new particles.

Our system saves the computed particle traces in a graphics metafile. This is implemented for two reasons: (1) a large number of time steps of flow data can be visualized and (2) the computed particle traces can be animated repeatedly without any re-calculation. A disadvantage is that a graphics metafile must be created. We found that this is not a significant problem for most flow data sets, since the size of the graphics metafile is considerably smaller than the size of the flow data. The graphics primitives used in the graphics metafile are very simple. Since we are only interested in rendering particles, the following graphics primitives are needed: color(), move(), draw(), and point(). This made the system very easy to port to other systems.

We have built our system on a Cray Y-MP, a Convex C3240, and an SGI 4D graphics workstation. The data sets given in Table 2 are stored on the Convex C3240. The Convex C3240 in the Numerical Aerodynamic Simulation (NAS) facility is configured with 100 gigabytes of disk storage and one gigabyte of main memory. The graphics metafile generated on the Convex is transferred to the SGI 4D workstation where visualization is performed. For small unsteady flow data sets, the system can run on the graphics workstation, and the particle traces can be visualized locally on the workstation.

6 Visualization

We used an existing graphics system to render particle traces. One of our requirements for the graphics system is that it provides an animation utility. Another requirement is that the graphics system can render Chimera grids. FAST met both requirements and was used for the visualization. FAST also has a utility that allows us to record the animation for video production.

Our system allows simultaneous visualization of scalar and vector fields. The system assigns color to the particles with respect to a specified scalar quantity. The scalar quantity can be any one of the following: (x, y, or z) coordinate of the particle, the time at which the particle was released, or a scalar function computed from the given scalar quantities stored in the solution files. For the visualization, we colored the particles by the time of their release. This is very useful for visualizing particle trajectories. It also shows how fast the particles travel in the unsteady flow field. Sometimes, when there are many seed locations it may be difficult to determine the seed location in which a particle was released from. Our system allows the user to assign colors to the particles based on their seed locations.

Sometimes it may not be possible to store all time steps of an unsteady flow data set in the system because of the available disk space. It is desirable to save the particles at the end of a run and resume the particle tracing at a later run. This feature was implemented in our system, and it allowed users to visualize their unsteady flow data sets without requiring all time steps to be available on the disk at once.

When the number of particles becomes too large, they may be difficult to visualize because they obstruct one another. The number of particles generated is a function of the number of time steps and the seed locations. At each time step, new particles are released from the seed locations. A feature that keeps particles from the last n time steps was implemented, where n is the specified by the user. Any particles that were released in the previous $n-1$ time steps are removed from the particle trace. This feature is useful especially when the particles circulate in a closed volume.

7 Results

In this section, we show several snapshots from the animation of the streaklines computed using the two data sets given in Table 2. The streaklines are simulated by releasing particles from a number of seed locations at each time step. Our first example is the Descending Delta Wing which has two thrust-reverser jets in descent with a descent rate of Mach 0.004. The Mach number of the jet itself is 1.0. A total of 139 time steps was used in the animation. Figure 1 shows the geometry of the Delta Wing with the initial seed locations where the particles were released. A total of 340 seed locations was placed near the ground, on the leading edge of the Delta Wing, and near the two jet exits. Figure 2 shows the streaklines at time 138. The particles are colored by the time of their release from the seed locations. Blue represents the earliest time and magenta represents the most recent time. At each time step, 340 particles are released from the seed locations. Hence, at time 138 there are at most 139×340 particles (time starts at 0). Some particles may not be shown because they reached the grid boundary. The seed locations near the jet exits and on the ground were chosen so that the interaction of the particles released from the jet and from the ground could be seen. This interaction is very evident in the animation.

Our second example is the V-22 tiltrotor aircraft. This aircraft has two propellers that rotate in opposite directions. Figure 3 shows a partial geometry of the V-22 tiltrotor with two rectangular patches of seed locations positioned near a rotor blade in each propeller. Particles are released relative to the rotor blade's position. There are 400 seed locations. Figure 4 shows the simulated streaklines at time 49, at which the rotor blades have made one-half revolution. At this time, there are approximately 20,000 particles (400 particles $\times$ 50 time steps.) Figure 5 shows the streaklines at time 98, at which the rotor blades have made one full revolution. It can seen that the tip of a rotor blade has cut through several patches of particles. This is visible from the particles that were released during time steps 70-80 (the light-green particles); see the color bar shown in the figure. Figure 6 depicts the streaklines at time 196, at which the rotor blades have made another full revolution. The breakup of particles by the tip of a rotor blade is also visible from the particles that were released during time steps 170-180 (the dark-red particles). Furthermore, the particles that were released from earlier time steps have moved downstream.

8 Conclusions

We have developed a post-processing system to compute streaklines from large 3D unsteady flow fields. Because of the data sizes involved (see Table 2), it is impossible to store all time steps in the physical memory of the system. However, in order to obtain a reasonable animation rate, ideally all time steps should be available in memory. A possible approach is to preprocess the data so that only a number of time steps can be visualized. This type of approach was implemented in [3,20]. The maximum number of time steps is then determined by the size of the physical memory. This technique is attractive because it allows the user to interactively examine the flow field. However, this approach limits the number of time steps that can be visualized. We used an approach where at most two time steps of flow data are stored in memory at any time. This removes the limitation on the number of time steps that the system can visualize. The accuracy of the integration scheme implemented in our system is only second order. A higher order integration scheme is currently being implemented. We are also developing an interactive visualization system that will allow the user to view the particle traces while they are being generated.

Acknowledgments

The author would like to thank Al Globus, Bob Haimes, Jeff Hultquist, and Tom Woodrow for their helpful comments. Pieter Buning and Kalpana Chawla provided several interesting discussions on particle calculation. The author also thank the reviewers for their helpful comments. The flow data for the Delta Wing was provided by Kalpana Chawla, and Bob Meakin provided the flow data for the V-22 tiltrotor. The geometry of the V-22 tiltrotor was provided by Bell Helicopter Textron, Inc. and Boeing Helicopters. This work was supported by NASA under contract NAS 2-12961.

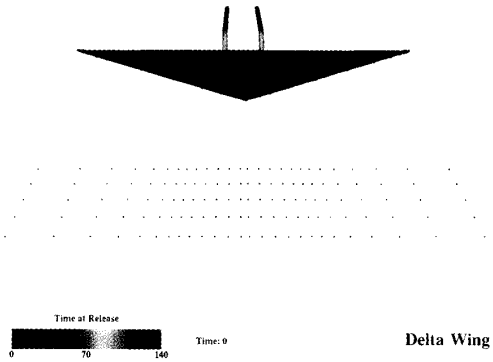

Figure 1: Particles are released on the leading edge of the wing, on the ground, and near the jet exits.

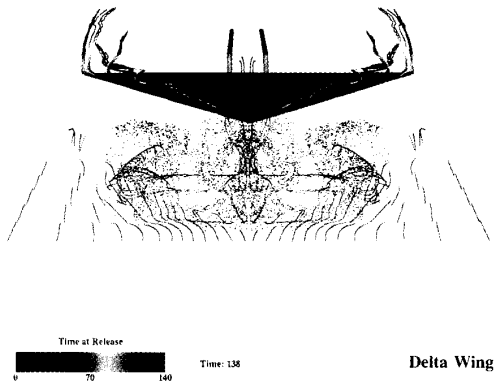

Figure 2: Streaklines at time 138. Particles are colored by the time of their release.

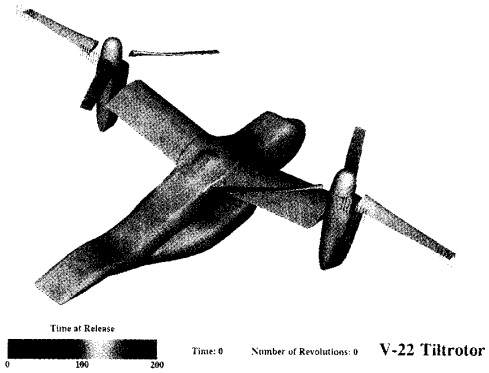

Figure 3: Particles are released near a rotor blade of each propeller. A partial geometry of the tiltrotor is shown.

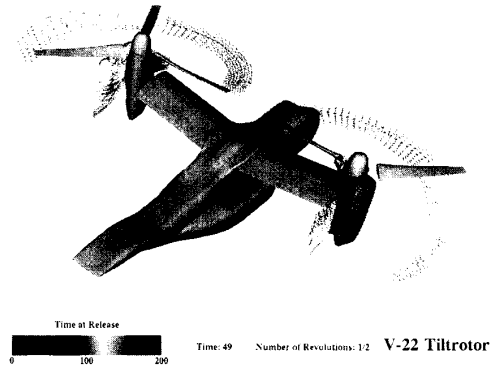

Figure 4: Streaklines at time 49. The rotor blades have completed half revolution.

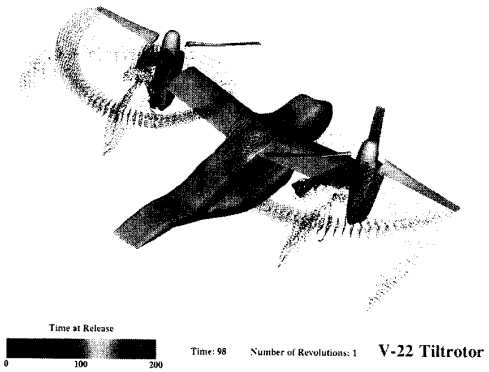

Figure 5: Streaklines at time 98. The rotor blades have completed one full revolution.

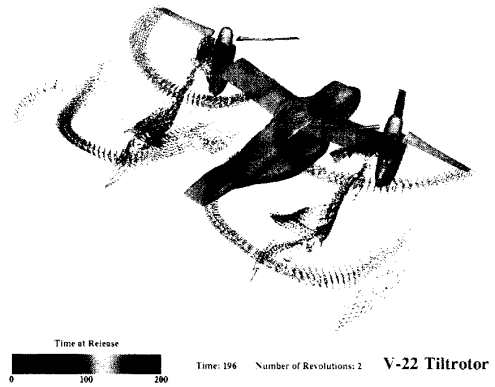

Figure 6: Streaklines at time 196. The rotor blades have completed two full revolutions.

(See color plates, p. CP-4.)

References

[1] Bancroft, G., Merritt, F., Plessel, T., Kelaita, P., McCabe, K., and Globus, A., FAST: A Multi-Processed Environment for Visualization of Computational Fluid Dynamics, in: A. Kaufman, ed., *Proceedings of Visualization '90,* San Francisco, California, October 1990, pp. 14-27.

[2] Benek, J., Buning, P., and Steger, J., A 3-D Chimera Grid Embedding Technique, *7th Computational Fluid Dynamics Conference,* Cincinnati, Ohio, July 1985, AIAA 85-1523.

[3] Bryson, S. and Levit, C., The Virtual Wind Tunnel, *IEEE Computer Graphics & Applications,* Vol. 12, No. 4, July 1992, pp. 25-34.

[4] Buning, P. and Steger, J., Graphics and Flow Visualization in Computational Fluid Dynamics, *7th Computational Fluid Dynamics Conference,* Cincinnati, Ohio, July 1985, AIAA 85-1507.

[5] Chawla, K. and Van Dalsem, W., Numerical Simulation of STOL Operations Using Thrust-Vectoring, *AIAA Aircraft Design Systems Meeting,* Hilton Head, South Carolina, August 1992, AIAA 92-4254.

[6] Corrie, I., The Fundamental Mechanics of Fluids, McGraw-Hill, New York, 1974.

[7] Crawfis, R. and Max, N., Direct Volume Visualization of Three-Dimensional Vector Fields, in: A. Kaufman and W. Lorensen, eds., *1992 Workshop on Volume Visualization,* Boston, Massachusetts, October 1992, pp. 55-60.

[8] Darmofal, D. and Haimes, R., Visualization of 3-D Vector Fields: Variations on a Stream, *30th AIAA Aerospace Sciences Meeting and Exhibit,* Reno, Nevada, January 1992, AIAA 92-0074.

[9] Globus, A., A Software Model for Visualization of Time Dependent 3-D Computational Fluid Dynamics Results, *NAS Applied Research Technical Report,* NASA Ames Research Center, RNR 92-031, November 1992.

[10] Globus, A., Levit, C., and Lasinski, T., A Tool for Visualizing the Topology of Three-Dimensional Vector Fields, in: G. Nielson and L. Rosenblum, eds., *Proceedings of Visualization '91,* San Diego, California, October 1991, pp. 33-40.

[11] Grinstein, F., Obeysekare, U., and Patnaik, G., Flow Visualization as a Basic Tool to Investigate the Dynamics and Topology of Jets, in: A. Kaufman and G. Nielson, eds., *Proceedings of Visualization '92,* Boston, Massachusetts, October 1992, pp. 164-170.

[12] Haimes, R., pV3: A Distributed System for Large-Scale Unsteady CFD Visualization, submitted to *32nd AIAA Aerospace Sciences Meeting and Exhibit,* Reno, Nevada, January, 1994.

[13] Haimes, R. and Giles, M., VISUAL3: Interactive Unsteady Unstructured 3D Visualization, *29th AIAA Aerospace Sciences Meeting and Exhibit,* Reno, Nevada, January 1991, AIAA 91-0794.

[14] Helman, J. and Hesselink, L., Visualizing Vector Field Topology in Fluid Flows, *IEEE Computer Graphics & Applications,* Vol. 11, No. 3, May 1991, pp. 36-46.

[15] Hultquist, J., Constructing Stream Surfaces in Steady 3D Vector Fields, in: A. Kaufman and G. Nielson, eds., *Proceedings of Visualization '92,* Boston, Massachusetts, October 1992, pp. 171-178.

[16] Jespersen, D. and Levit, C., Numerical Simulation of Flow Past a Tapered Cylinder, *29th AIAA Aerospace Sciences Meeting and Exhibit,* Reno, Nevada, January 1991, AIAA 91-0751.

[17] Meakin, R., Moving Body Overset Grid Methods for Complete Aircraft Tiltrotor Simulations, *11th AIAA Computational Fluid Dynamics Conference,* Orlando, Florida, July 1993.

[18] Schroeder, W., Volpe, C., and Lorensen, W., The Stream Polygon: A Technique for 3D Vector Field Visualization, in: G. Nielson and L. Rosenblum, eds., *Proceedings of Visualization '91,* San Diego, California, October 1991, pp. 126-132.

[19] Shirayama, S., Visualization of Vector Fields in Flow Analysis I, *29th AIAA Aerospace Sciences Meeting and Exhibit,* Reno, Nevada, January 1991, AIAA 91-0801.

[20] Smith, M., Van Dalsem, W., Dougherty, F., and Buning, P., Analysis and Visualization of Complex Unsteady Three-Dimensional Flows, *27th AIAA Aerospace Sciences Meeting and Exhibit,* Reno, Nevada, January 1989, AIAA 89-0139.

Visualization of Time-Dependent Flow Fields, D.A. Lane, pp. 32-38.

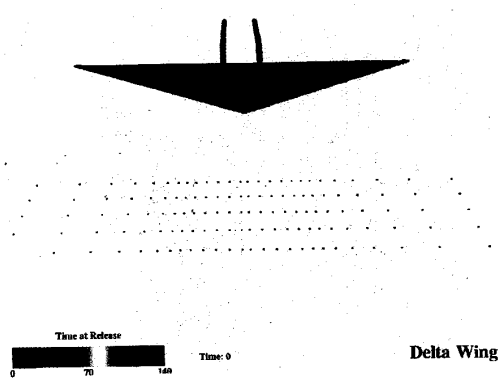

Figure 1: Particles are released on the leading edge of the wing, on the ground, and near the jet exits.

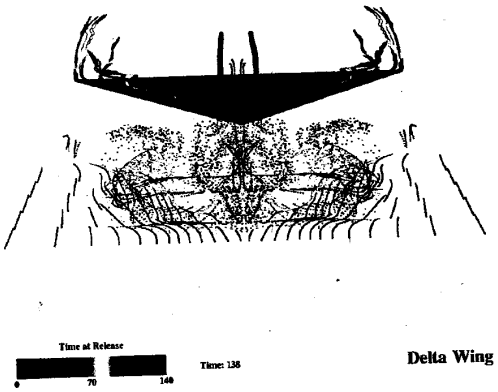

Figure 2: Streaklines at time 138. Particles are colored by the time of their release.

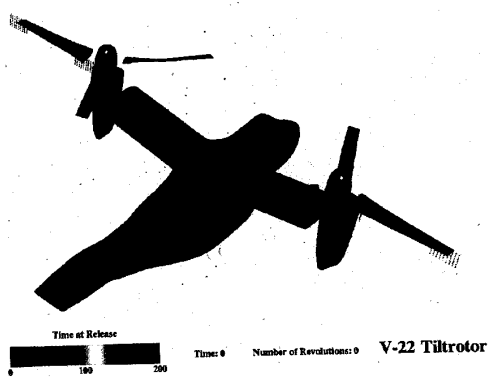

Figure 3: Particles are released near a rotor blade of each propeller. A partial geometry of the tiltrotor is shown.

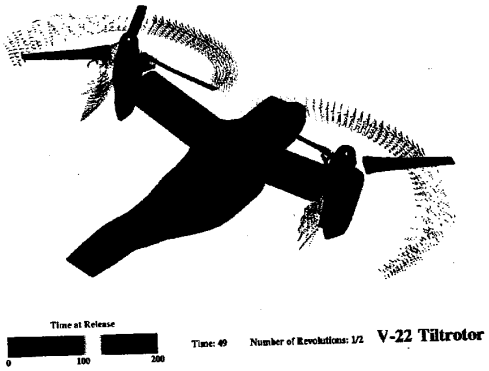

Figure 4: Streaklines at time 49. The rotor blades have completed half revolution.

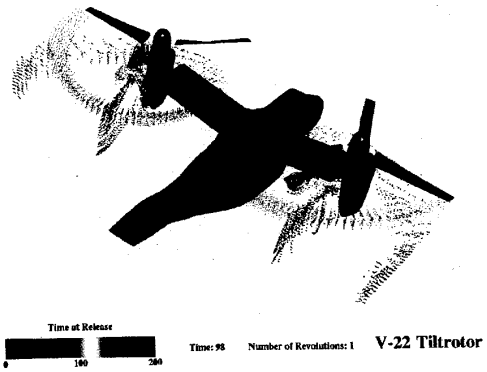

Figure 5: Streaklines at time 98. The rotor blades have completed one full revolution.

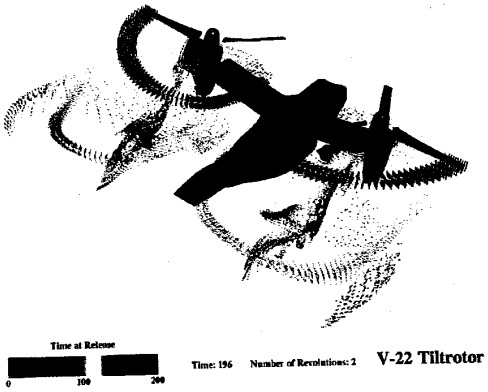

Figure 6: Streaklines at time 196. The rotor blades have completed two full revolutions.

BEST COPY AVAILABLE