ANIMATION OF INTERACTIVE FLUID FLOW VISUALIZATION TOOLS ON A DATA PARALLEL MACHINE

J. A. Sethian*

DEPARTMENT OF MATHEMATICS
UNIVERSITY OF CALIFORNIA, BERKELEY
BERKELEY, CALIFORNIA 94720

James B. Salem

THINKING MACHINES CORPORATION
CAMBRIDGE, MASSACHUSETTS 02142

Summary

We describe a new graphics environment for essentially real-time interactive visualization of computational fluid mechanics. The researcher may interactively examine fluid data on a graphics display using animated flow visualization diagnostics that mimic those in the experimental laboratory. These tools include display of moving color contours for scalar fields, smoke or dye injection of passive particles to identify coherent flow structures, and bubble wire tracers for velocity profiles, as well as three-dimensional interactive rotation and zoom and pan. The system is implemented on a data parallel supercomputer attached to a framebuffer. Since most fluid visualization techniques are highly parallel in nature, this allows rapid animation of fluid motion. We demonstrate our interactive graphics fluid flow system by analyzing data generated by numerical simulations of viscous, incompressible, laminar and turbulent flow over a backward-facing step and in a closed cavity. Input parameters are menu-driven, and images are updated at nine frames per second.

*J. A. Sethian is partially supported by the Sloan Foundation and the Applied Mathematics Subprogram of the Office of Energy Research under contract DE-AC03-76SF00098.

The International Journal of Supercomputer Applications, Volume 3, No. 2, Summer 1989, pp. 10–39.

Introduction

The long-range goal of this project is to build a numerical fluid simulation environment that provides interactive flow visualization tools together with rapid animation to allow feedback between display information, input parameters, the algorithm chosen, and, ultimately, the underlying model. In this paper, we present a step toward that goal with the introduction of a new graphics environment for essentially real-time visualization of the results of numerical simulations of fluid mechanics. Starting from a precomputed discrete set of time-dependent flow quantities, such as velocity and density, the user may interactively examine the data on a framebuffer using animated flow visualization diagnostics that mimic those in the experimental laboratory. The graphics environment accepts data written in a general format, and can handle a wide variety of flow geometries. Images are updated at nine frames per second, providing an effective way to study the solution, analyze fluid flow mechanisms, and compare numerical results with experiment.

As in a laboratory experiment, our graphics environment allows the researcher to study fluid velocity by injecting color-coded dye that is passively advected as the animation unfolds. Because the animation is both interactive and in real time, moving eddy structures can be tagged and tracked with dye as they form. Different colored dyes can be used to watch regions merge or diffuse, and mixing mechanisms may be studied. Time-dependent scalar fields, such as temperature and density, can be color-tagged, and the range of displayed values may be changed during the animation. For three-dimensional flows, interactive rotation is possible as the animation proceeds, as well as zoom and pan. In this environment, the researcher typically performs dozens of realistic flow visualization experiments in rapid succession.

The system is implemented on a data parallel supercomputer with parallel input/output to a framebuffer. Rapid display results from fluid visualization techniques that are highly parallel. The key components of this interactive graphics environment are a Connection Machine CM-2 data parallel supercomputer and a CM-2 framebuffer.

The environment assumes a stored data field, obtained through either numerical simulation or physical experiment. The data are preprocessed so that the rele-

vant physical quantities, such as velocity, vorticity, density, pressure, and temperature, are given at each time step on a fixed discrete grid in the computational domain. The full set of data is then loaded into the memory of the front-end machine. At each time step, the next data field is downloaded into the CM-2's memory, the appropriate numerical calculations are performed in parallel on the CM-2, and the images are displayed on the framebuffer.

More detailed information about the CM-2 is presented in Section 3. The algorithms are described in Sections 4 and 5, and memory storage requirements are discussed in Section 6. Examples of flows analyzed using this environment are given in Section 7. The key visualization tools are summarized below.

Vector Quantities: The tools for both two- and three-dimensional vector quantities, such as velocity, are

1) injection of passively advected tracer particles;
2) mouse-driven location of injection points;
3) menu-specified frequency and number of injections;
4) menu-driven color-tagging according to location, frequency, time, and number of injections; and
5) menu-driven injection of tracer particles,
   a) along a line ("hydrogen bubble" tracers),
   b) concentrated in a disk or ball ("smoke/dye" injection),
   c) on a ring or sphere, or by
   d) continuous injection at a point.

Scalar Quantities: For scalar functions of two space variables, such as temperature, density, and pressure, the current tools are

1) motion of regions of selected contoured scalar values, generated by appropriate coloring of all points in selected range; and
2) mouse or menu-driven selection of range of scalar values and color map to be displayed.

Display Attributes: As the animation sequence proceeds, the user may interactively

1) rotate the viewpoint for three-dimensional flows,
2) change the color display map, and
3) zoom and pan.

On an 8,192-processor CM-2, images are updated on the framebuffer at nine frames per second, producing essentially real-time motion.

"Starting from a precomputed discrete set of time-dependent flow quantities, such as velocity and density, the user may interactively examine the data on a framebuffer using animated flow visualization diagnostics that mimic those in the experimental laboratory."

As a demonstration, we analyze data produced from numerical simulations of viscous, incompressible, laminar and turbulent flow. We examine two-dimensional flow over a backward-facing step and in a closed square. Using our graphics environment, we isolate and identify a variety of physical flow phenomena, such as eddy formation and merger, propagation and decay, horseshoe vortices, mixing and intertwining of fluid structures, and pairing of counteroscillatory vortical structures.

A prototype version of this environment, as well as the following background summary of some flow visualization work was described in a technical report presented at Supercomputing '88 (Sethian, Salem, and Ghoniem, 1988). However, the implementation described here uses a different method for storing and retrieving data in the Connection Machine. This new implementation allows far more data to be stored (see Section 6).

Results and Discussion

1. PHYSICAL AND NUMERICAL FLOW VISUALIZATION

1.1 PHYSICAL VISUALIZATION

Flow visualization has a long and rich history as a technique for analyzing fluid mechanics in laboratory apparatus. Starting with the experiments of Reynolds and Prandtl, flow visualization techniques can be divided into three main categories (as described in Gad-el-Hak, 1987; Merzkirch, 1987; Werle, 1973). The first involves the addition of foreign materials, either fluid or solid, into the flow. The added substance is visible, and the flow is indirectly viewed through the motion of the foreign substance. In such techniques, it is important to minimize the effect of the added substance on the original flow. Visualization techniques of the second category rely on the determination of physical qualities of the flow itself. An example is variation in fluid density, which relates to the refractive index of the fluid. Measurement of optical changes in the refracted light shows density variations in the flowing fluid. The third category involves the introduction of physical changes into

the flow. Examples are the release of luminescent particles and the introduction of localized energy at specific points. Experiments using these techniques and photographs illustrating a variety of fluid phenomena can be found elsewhere (Billet, Kim, and Heidrick, 1985; Honji, 1975; Merzkirch, 1987; Reed, 1987; Van Dyke, 1982; Veret, 1987; Werle, 1973).

The technique chosen depends on what one wants to see. In the first category, namely the addition of foreign objects, one might wish to examine three different objects: (1) a *streamline,* which is a curve in the domain that is tangent to the instantaneous (time-frozen) velocity field; (2) a *pathline,* which is the path in the domain a particle takes as it moves through the flow as a function of time; and (3) a *streakline,* which is the position of all particles in the flow that have passed through a fixed point in the domain during a given time interval. For stationary flow, the three are equivalent. The discussion below illustrates the distinction between the various objects.

One way to reveal these flow structures is to inject dye into fluids and smoke into gases (see Gad-el-Hak, 1987; Merzkirch, 1987). Injection is either through small holes or slots in the sides of the apparatus, or externally into the body of the flow, by means of a hypodermic needle for dye or wires/tubes for smoke. The exposure time and the number and frequency of injection points determine the type of curve obtained. Sprinkling a large number of particles throughout a domain (such as aluminum dust, see Honji, 1975), and photographing for a short time exposure shows streamlines. A long time exposure of a single particle shows a pathline, whereas continuous injection of dye through a fixed point shows a collection of streaklines passing through the same point.

Smoke or dye injection is particularly effective for visualizing the development of large, coherent flow structures, such as vortices (see Merzkirch, 1987). As these large structures roll up, smoke or dye can concentrate into confined areas connected by thin wisps. To illustrate velocity profiles, one common technique uses a hydrogen bubble wire placed in the physical domain. Relying on the electrolysis of water (Merzkirch, 1987), a voltage is applied through the wire, and hydrogen

"As a demonstration, we analyze data produced from numerical simulations of viscous, incompressible, laminar and turbulent flow. We examine two-dimensional flow over a backward-facing step and in a closed square."

"The exposure time and the number and frequency of injection points determines the type of curve obtained. Sprinkling a large number of particles throughout a domain and photographing for a short time exposure shows streamlines. A long time exposure of a single particle shows a pathline, whereas continuous injection of dye through a fixed point shows a collection of streaklines passing through the same point."

bubbles are emitted (as well as smaller oxygen bubbles). If the wire is placed normal to the flow, the bubbles are carried away from the wire, showing the velocity profile.

The second category of fluid visualization relies on optical effects due to the density variations in the flow. As outlined by Merzkirch (1987), this can be used to illustrate such mechanisms as (1) shock formation in compressible flow, (2) mixing of fluids of differing densities, and (3) combustion. The most straightforward technique is a shadowgraph, which is simply a photograph of the image of the shadow from a point-shaped light source, yielding qualitative information about density variations. A more involved, but highly effective, technique results from schlieren methods, which use optical devices to qualitatively measure the diffraction of light and hence fluid density variation.

Photographs produced from the above are often enhanced through image processing. The image is digitized, and streaklines and streamlines can be identified. Mathematical techniques such as splines and Fourier filtering are used to extract quantitative flow data (see a series of articles in Billet, Kim, and Heidrick, 1985; Veret, 1987). A particularly intriguing application of image processing is in the identification of large coherent structures in turbulent flow. Such structures are often masked by small-scale noise; Corke (1984) has used a filtering technique to bleed out high wave numbers and identify large, coherent, low wave fluid structures. The application of some of these techniques to numerical flow data will be discussed elsewhere (Sethian, 1989).

1.2 NUMERICAL VISUALIZATION

Extracting flow properties from numerical data is equally challenging. In numerical simulations, immense amounts of data are produced, far more than can be understood as raw data. For example, Winkler and associates (1987) have calculated that their two-dimensional code with 250,000 computational zones yielding five flow variables every four time steps gives a sustained data generation rate of 3.47 megabits per second, where a megabit is defined to be 1,024 × 1,024 bits. As an alternative, the researcher tries to extract the "right"

quantities from the data to verify conjectures and generate new ones. A natural way of performing such extraction is through visualization. As has been pointed out (Winkler et al., 1987; Zabusky, 1987), the human brain is able to interpret massive amounts of raw data through images (Smarr, 1987). Another role of visualization, as discussed by Smarr (1987), is to avoid the prohibitively high cost of central mass storage; it may be more cost-effective to store computed images instead of all the intermediate results. For further discussion of scientific visualization, see McCormick, DeFanti, and Brown (1987).

Flow visualization in numerical experiments performs a role similar to that in physical experiment—namely, to isolate particular variables or fluid quantities, such as wave interactions, surface instabilities, and vortex generation (see Winkler et al., 1987). Large aggregate structures, such as moving fluid eddies, are difficult to quantify, but are easy to compare with experiment and with other simulations. Motion (animation) provides history, which allows one to see dynamic changes in the flow, on both the small scale (sharp gradients) and the large scale (topological changes). Such mechanisms may be very hard to extract from raw data or vector velocity plots. As pointed out by Zabusky (1987), visualization also provides an avenue for discovering new phenomena, such as the strong supersonic vortex ring behind a shock (Winkler et al., 1987). Using computer versions of the flow visualization techniques described above, images of a variety of flow phenomena have been generated in recent years (Reed, 1987; Rogers, Buning, and Merritt, 1987; Sethian, 1987; Smarr, 1987; Winkler et al., 1987; Zabusky, 1987).

"Flow visualization in numerical experiments performs a role similar to that in physical experiment—namely, to isolate particular variables or fluid quantities, such as wave interactions, surface instabilities, and vortex generation. Large aggregate structures, such as moving fluid eddies, are difficult to quantify, but are easy to compare with experiments and with other simulations."

2. PAST WORK

Several different approaches to visualizing numerical flow simulations have been taken. A major difference in the approaches lies in the division of computer resources between the algorithms that approximate the equation of motion and those that generate the image.

A sophisticated graphics environment has been developed by Winkler et al. at Los Alamos National Laboratory, and duplicated at other institutions (Smarr, 1987; Winkler et al., 1987). The basic arrangement, referred

to as a "numerical laboratory," is that the algorithms that approximate the equations of motion in terms of key variables run on a large supercomputer, while a secondary machine accepts intermediate values as they are computed and generates auxiliary variables, along with visual images, which are turned into movies. An important issue is to maximize communication rates between the various components. Some spectacular movies have been made with this facility, and the ability to generate a movie of virtually every numerical simulation provides a record of variations in computed results and images as the model and numerical parameters are changed. In this arrangement, the researcher must reside at the facility to produce both the numerical solution and the images, or have someone else produce videotapes and mail them.

At the other extreme, the user is located far from the supercomputer facility. A low-cost environment for producing local images from data generated at a remote site has been described in (Johnston et al., 1988).

In the above arrangements, the generation of images is noninteractive. A different approach with some interactive features has been developed at the Numerical Aerodynamic Simulation program at NASA Ames Research Center (Rogers, Buning, and Merritt, 1987). Here, the full set of results of a numerical simulation is stored before images are made. Images are then produced from the stored data using a distributed interactive graphics environment. Working from a graphics workstation, the user can display contours and surfaces for scalar variables and arrows representing velocity vectors. To display particle trajectories, the coordinates of the injected particles are entered into the workstation. The trajectories are then computed on a remote supercomputer, shipped back to the workstation, and displayed. (Images computed in this manner are shown in Rogers, Buning, and Merritt, 1987.)

In this paper, we present a third approach, namely, that of using a data parallel supercomputer with parallel I/O to a framebuffer to analyze precomputed data. Since most flow visualization algorithms are highly parallel, this allows rapid animation of flow visualization. The suitability of a massively parallel processor to scientific visualization was mentioned previously (Winkler et

al., 1987), and the Connection Machine in particular has been discussed (Zabusky, 1987).

3. THE CONNECTION MACHINE CM-2 SYSTEM

The Connection Machine CM-2 data parallel computer contains up to 65,536 1-bit processors, each with 8,000 bytes of local memory. The processor connection topology is essentially a hypercube; however, a general purpose communications system called the "router" allows messages to be sent from any processor to any other while the exact route is hidden from the user. Nearest neighbor routing patterns for n-dimensional grids are preprogrammed for efficiency. Each processor is also directly connected to one of eight possible 75 Mbyte/sec internal I/O channels.

The CM-2 is a single-instruction, multiple-data (SIMD) computer. It is programmed in a data parallel fashion: one assigns a processor to each data element in the problem. The user writes and debugs CM programs using high-level languages which are parallel extensions of Fortran 8X, C, or Lisp, on a front-end computer such as a Sun or a VAX. The serial portions of the program are executed on the front-end machine, and the parallel instructions, including communications, I/O, and indirect addressing, are broadcast to all the processors via a sequencer. The sequencer's microcode library translates the front end's parallel instruction set to the processors' very simple RISC instruction set. Each group of 32 processors, hereafter called a "node," shares an optional floating point accelerator (either a 32-bit single precision instruction or 64-bit double precision) and indirect addressing hardware. A table may be stored at each node, and using the indirect addressing hardware, each processor in the node may access a different value in the table.

The flexibility of the machine is greatly enhanced by the notion of "virtual processors." Each physical processor simulates a number of virtual processors by segmenting its memory and time, multiplexing the processor hardware. Its implementation is transparent, in that the user simply declares the number of virtual processors needed. The number of virtual processors may be changed dynamically and tuned to fit the problem's data set.

"We present an approach to visualizing numerical flow simulations that uses a data parallel supercomputer with parallel I/O to a framebuffer to analyze precomputed data. Since most flow visualization algorithms are highly parallel, this allows rapid animation of flow visualization."

"A data-parallel approach dictates that a processor be assigned to each data element in the problem. Since the data elements in our problem are particles, one processor is assigned to each injected particle. Since the particles are passively advected, each may be moved in parallel."

The Connection Machine system's I/O devices include a framebuffer, a high-speed parallel disk, and a VME interface. The framebuffer produces a high-resolution 1,280 × 1,024 image with up to 24 bits per pixel and normally displays it on a noninterlaced Sony monitor. The framebuffer is directly connected to the internal I/O bus. Thus, the Connection Machine system can update a television-quality (NTSC) image in real time (30 frames per second).

A 65,536-processor Connection Machine system typically sustains 2,500 MIPS in normal computations. With the floating point accelerator, it sustains a single precision floating rate of 3,500 MFLOPS (measured in a 4K × 4K matrix multiply benchmark). The performance scales nearly linearly with the number of processors in the system. The programs used in this paper are written in *Lisp (pronounced star-lisp), a set of parallel extensions to the Common Lisp language. For more information on the Connection Machine system and *Lisp, see Hillis (1985) and Thinking Machines Corp. (1987).

4. DISPLAY OF TIME-DEPENDENT TWO-DIMENSIONAL FIELDS

4.1 TWO-DIMENSIONAL VELOCITY FIELDS

Velocity fields are displayed by injecting tracer particles into the flow. A data-parallel approach dictates that a processor be assigned to each data element in the problem. Since the data elements in our problem are particles, one processor is assigned to each injected particle. Since the particles are passively advected, each may be moved in parallel.

The algorithm works as follows. We set the initial positions of the injected particles (possible injection layouts are described below). Each injected particle is assigned to a processor, whose memory contains the position and color of the injected particle. At each time step, to advance the location of the injected particles, each processor retrieves the velocity components located on the corners of the grid cell containing its particular particle. (Various techniques for storing the velocity field in the Connection Machine's memory are discussed in Sec-

tion 6.) The velocity at the particle's position is obtained by using bilinear interpolation in space and time between neighboring grid points and the two velocity fields (current and next). The position of each particle is then updated using a first-order Euler scheme. Finally, the positions and colors of the particles are displayed by accessing in parallel the pixels on the attached CM-2 framebuffer. Zoom and pan are provided, and the number of interpolations between one time step and the next may be changed. Several different injection layouts are possible, depending on the phenomenon under study.

Smoke/Dye Injection: Smoke/dye injection is used in the laboratory to visualize the evolution of flow. As mentioned above, this technique is useful in isolating and identifying coherent fluid structures (Merzkirch, 1987). The user specifies the number of particles injected, injection point (or points), injection radius R_{inj}, frequency of injection (number of time steps between injections), and color-tagging scheme. Particles are then injected inside a circle of radius R_{inj} around each injection point. To give the appearance of dye, the distribution of injected particles is a truncated Gaussian with mean at the injection point. A large number of particles is released to give a dense concentration similar to dye. The injected particles are then advected by the underlying velocity field.

Bubble Wire Injection: In this technique, used to illustrate velocity profiles (Merzkirch, 1987), fewer particles are released along a thin, straight line in the flow. We typically release between 20 and 100 particles evenly along the wire, either continuously or every fixed number of time steps. This tool is used to illustrate velocity gradients, and mimics the creation of a thin line of gas bubbles in the flow. The user specifies the beginning and end of the wire, as well as the number and frequency of released particles.

Ring Injection: In an incompressible flow, one may wish to follow the evolution of a patch of constant area. Here, we might examine the change in the length of the boundary in an attempt to understand what happens to a free surface. Since the boundary of the region is passively advected in an incompressible flow, there is no flow through the boundary and the area inside remains

constant, even though the boundary becomes wildly distorted. In this technique, we place a large number of particles along a ring around a specified center. We use enough particles so that even with large stretch and distortion, the boundary of the region is discernible. Other initial configurations can be programmed.

Continuous Injection: In this technique, we inject a stream of particles continuously at a point in the flow. This gives a collection of streaklines and can provide a good way to gauge unsteadiness in a flow (Merzkirch, 1987).

"Bilinear interpolation and the Euler method were chosen to calculate the velocity at each particle and for time integration along the particle trajectory because they were simplest. However, it is important to stress that the advection of these particles is a purely passive exercise, which does not in any way affect the velocity field used to transport them."

The use of bilinear interpolation to calculate the velocity at each particle and the Euler method for time integration along the particle trajectory warrant some discussion. Bilinear interpolation and the Euler method were chosen because they were the simplest. However, it is important to stress that the advection of these particles is a purely passive exercise, which does not in any way affect the velocity field used to transport them. There are two sources of numerical error in the approximate trajectory calculation. One is spatial error, owing to the bilinear interpolation of the velocity field, and the other is temporal error, resulting from the use of a first-order method in the time integration. Indeed, one noticeable effect of these errors is that particles moving in a circular velocity field (such as the initial pattern in the closed square) do not return to their exact starting points after completing the rotation; the size of this error depends on the order of the numerical method used in the time integration and the size of the time step. We have experimented with higher order methods (second-order Heun's method). Note, however, that this requires an extra function evaluation at each time step. The current algorithm allows one to choose a smaller time step for Euler's method, thus reducing the error, and simply execute several position updates between displays. This seems satisfactory. In practice, even with the complicated flows shown, the error in this advection is small compared with the accuracy associated with the solution of the underlying equations of motion.

4.2 TWO-DIMENSIONAL SCALAR FIELDS

For scalar variables Ψ, such as stream function, temperature, density, energy, or pressure, moving color-coded

contours are displayed. Here, $\Psi = \Psi(x,y,t)$ is a function of space variables (x,y) and time t. The user may choose to display only a limited range of Ψ, allowing the isolation of particular phenomena, such as the merger of vortex structures.

A standard technique for two-dimensional scalar fields is to display contour lines. However, the calculation of contour lines is time-consuming and, ultimately, not necessary. Instead, each pixel lying in the interior of the fluid flow geometry is colored according to the current magnitude of the variable under examination at each time step. The algorithm works as follows. We configure the CM-2 with enough virtual processors to assign one per pixel. Let C_i be the value of the color of the pixel corresponding to processor i. At the beginning of the run, the full data set of scalar variables is loaded in memory of the CM-2. Since there are more pixels than discrete grid values for Ψ, bilinear interpolation is used to go from the coarse data grid to the fine processor pixel grid. The user then specifies Ψ_{min} and Ψ_{max}, which are minimum and maximum display values for Ψ. Every time step, each processor i sets C_i to Ψ if $\Psi_{min} < \Psi(x,y,t) < \Psi_{max}$ (otherwise, C_i equals the background color), and displays its value of C_i at the appropriate place on the CM-2 framebuffer display. The region of values in the selected range moves as the flow evolves. The image may be panned or zoomed, and the number of interpolations between one time step and the next may be changed.

"A standard technique for two-dimensional scalar fields is to display contour lines. However, the calculation of contour lines is time-consuming and, ultimately, not necessary. Instead, each pixel lying in the interior of the fluid flow geometry is colored according to the current magnitude of the variable under examination at each time step."

5. DISPLAY OF TIME-DEPENDENT THREE-DIMENSIONAL FIELDS

5.1 THREE-DIMENSIONAL VELOCITY FIELDS

For three-dimensional velocity fields, we use essentially the same passive particle advection techniques as those described above. Particles may be injected either intermittently or continuously as (1) smoke/dye, in either a ball or sphere, as well as two-dimensional disks at any angle in the flow, or (2) hydrogen bubbles, along a plane or line at any angle in the flow. To advance the particle positions, the neighboring eight velocity components on the cell grid points are retrieved and used in a first-order Euler time integration.

To display the results, we visualize a three-dimensional velocity field as if we were analyzing a flow encased in a glass tank. One then "walks" around the tank and looks at the flow from a variety of angles. The tank may be rotated in real time using the mouse, as well as zoom and pan. Thus, the user is easily able to compare one portion of the flow with another.

Using the simplest technique, each particle is projected onto the graphics screen by a parallel projection. We choose a "display plane" and project the particle's position onto the plane along a line perpendicular to the plane. Particles that lie directly behind other particles are not visible. The user selects which portion of the display plane is presented on the graphics screen. By using the mouse, the user rotates the display plane to view the flow from different angles.

It is often convenient to look at only a portion of the entire flow volume. We can reduce the width and the height by changing the portion of the display plane mapped onto the graphics screen. To reduce the depth of the volume visualized, the user chooses two "clipping planes," both parallel to the display plane. Only particles lying between the two planes appear on the screen. Of course, one might observe particles entering and leaving the viewing area at any time. Using the glass tank analogy, this is as if one could make a portion of the tank's contents invisible, allowing one to focus on only the area of interest.

To enhance the perception of depth, a number of other straightforward techniques are available within our environment. First, the intensity of the displayed particle's color can be related to the particle's distance from the display plane. Thus, near particles become brighter and far particles become fainter. Second, perspective can be added by choosing a "viewpoint" not in the display plane. Each particle's position is projected in perspective from the viewpoint. Third, several simultaneous views can be presented to the user. Similar to an architect's drawing, both a front view and a side view may be presented.

Despite these more advanced techniques, the ability to rotate the view in real time is the most important feature in providing the user with an intuitive feeling of a

three-dimensional flow. Our experience in viewing these flows indicates that it is *relative* motion of viewing perspective that provides the most information. Further discussion of three-dimensional display techniques can be found in a number of standard computer graphics texts (see, for example, Foley and Van Dam, 1984).

In the future, we expect to explore and compare more exotic three-dimensional display techniques. For example, one device mounts on top of a normal graphics monitor and permits two different images to be presented with polarized light of perpendicular angles. The images are computed using perspective projections from two different viewpoints. By wearing specially polarized glasses, the user is able to fuse the images into a coherent three-dimensional view much as with a stereoscope. By combining this with the ability to rotate the volume in real time, we hope to come close to real three-dimensional perspective.

"The ability to rotate the view in real time is the most important feature in providing the user with an intuitive feeling of a three-dimensional flow."

5.2 THREE-DIMENSIONAL SCALAR FIELDS

Three-dimensional scalar fields are somewhat more difficult to display, mostly because finding the level set corresponding to a particular level value efficiently is not a trivial issue. The most straightforward technique is to simply slice the domain into a discrete set of two-dimensional planes, and display the corresponding level set on each plane using the techniques described above. While this technique works fairly well, its effectiveness is highly dependent on the perspective and the number of discrete planes chosen.

In the full three-dimensional problem, given $\Psi = \Psi(x,y,z,t)$ and a scalar value $\Psi = \Psi_0$, we are looking for $\{x,y,z|\Psi(x,y,z,t) = \Psi_0\}$. The result is a two-dimensional surface lying in R^3. Given not one value, but a range of continuous scalar values, the resulting shape is a shell. These shapes are displayed using volume-rendering techniques, such as those discussed by Lorenson and Cline (1987) and Upson and Keeler (1988). The efficient implementation of a volume-rendering algorithm on a massively parallel machine has been mentioned previously (Salem, 1988) and is currently being explored by the authors.

6. STORAGE, TIMINGS, AND EXPERIENCE

6.1 STORAGE

The distribution of data in the memory of the Connection Machine is a significant factor in the overall frame update speed. Each processor has two available array reference techniques: (1) indirect addressing of a local array or (2) array lookup using the communication network.

In the first technique, a copy of the data array is stored at each node of 32 physical processors. Indirect addressing hardware allows each virtual processor to reference elements of the array in parallel in a time similar to normal memory references. Although this technique is fast, it is limited in terms of the size of the largest array that can be stored. In the current CM-2, a total of approximately 60,000 single precision (32-bit) floating point numbers can be stored at each node. For a two-dimensional array of two element velocity vectors this gives us a maximum array size of 250 × 120, for example. For three-dimensional arrays, the storage is acutely limited. For example, given three element velocity vectors, the maximum array size is under 30 × 30 × 30. This technique was used in the two-dimensional flows described in the next section.

For large arrays, the second technique is used. We assign each element of the array to a separate virtual processor and look up its value using the communication network. The Connection Machine "GET" instruction efficiently returns a value stored in these "array element processors" to the requesting particle processor. Although this method is significantly slower than indirect addressing, much larger arrays can be stored, up to 100 million single precision floating point numbers. For example, the maximum array size of a three-dimensional array of three element vectors would be about 350 × 350 × 350.

In the flows we have analyzed so far, both techniques have been employed. In the three-dimensional flows we have analyzed, we used the latter technique to store our three-dimensional flow, which is 32 × 64 × 32 elements in size. For the two-dimensional flows the first technique is used with a simple modification. While each time step's array fits into a node, there is not

enough storage for all the time steps. Therefore, we store all the arrays spread out through CM memory. At the beginning of each time step we broadcast the array associated with that time step to all the nodes and then use indirect addressing.

Between these two options of (a) a copy of the current velocity field in each node and (b) a single copy of the current velocity field accessed by all the processors, lies an optimum balance for data fields of a given size. For example, one might spread a copy of the data over every three nodes. Other techniques also can be used. First, the amount of data that can be put into each node's memory can be increased by truncating the floating point numbers of the numerical data, since there is little need to carry more significant digits than the accuracy of the time integration in Euler's method, or more than the capabilities of the output device. A hardware solution is to increase the memory per node. Finally, one can download the data from a high-speed parallel storage device, such as the CM Data Vault (see Thinking Machines Corp., 1987).

"One measure of system efficiency is the total number of frame updates per second. This depends on many variables, including the number of physical processors in the machine, the dimensionality of the situation, the number of particles injected, the number of bits used to store the fluid data in memory, and the number of time steps used in the Euler integration between displays."

6.2 TIMINGS

One measure of the efficiency of the system is the total number of frame updates achieved per second. As an illustration, for the time-dependent velocity field display, each frame update requires

1) making a copy of the current time velocity data in each node's memory
2) advancement of the particles' positions through Euler's method
3) display of particles, which consists of
   a) assignment of particles to display grid
   b) setting processors to background colors
   c) displaying colors.

The total number of frame updates depends on many variables, such as the number of physical processors in the machine (and the corresponding ratio of virtual to physical processors), the dimensionality of the problem, the number of particles injected, the number of bits used to store the fluid data in memory, and the number of time steps used in the Euler integration between dis-

plays, as well as the manner in which data are stored in the CM-2's memory.

In the two examples that follow, the fluid calculation of flow over a backward-facing step discussed in the next section yields 2,200 floating point numbers (two velocity components on a 11 × 100 grid) every time step, and the closed square calculation yields 6,442 floating point numbers (two velocity components on a 61 × 61 grid) every time step. The simulation software fully supports three-dimensional grids. For this two-dimensional problem, the z-coordinate is set to zero. The visualization is displayed on a 1,280 × 128 grid of pixels. To inject as many particles as processors, it takes 34 milliseconds to download the data, and 41 milliseconds to update 8,000 injected particles for one Euler time step. The time for display depends on the ratio of virtual processors to physical processors, since there is one virtual processor per pixel. On an 8K processor machine, with a virtual processor (VP) ratio of 20, display takes 55 milliseconds; on a 16K machine (VP ratio 10), display takes about 38 milliseconds. The total image update time on a 16K machine is 0.113 seconds, and we typically clock a sustained, running frame update speed of about nine frames a second.

6.3 EXPERIENCE

The process of developing and fine-tuning this graphics environment provided some good "rules of thumb" for conducting flow visualization experiments. Here, we would like to briefly touch on two issues—namely, the number of grid points necessary to show results, and the number of particles required to obtain adequate description of the flow mechanisms.

In a full three-dimensional simulation, it is likely that there will be so many grid points used in the numerical solution of the equations of motion that it will be impossible to copy a particular time step's data into each node. Possible solutions have been addressed above. However, it is important to point out that just because a calculation requires an especially fine grid (such as those in aeronautics) in order to obtain accurate enough results (such as pressure and density values), it may not be necessary to feed all those grid points to the visualization environment at once. Most believable flow features exist

across several grid points; thus, one might safely feed a subset of the available points (say, every third point) to the visualization simulation. In addition, one is free to send all the points in a subsection of the flow domain to the environment for closer examination.

In general, we have been able to obtain perfectly acceptable results with about 8,000 injected particles using an 8K processor machine; this, of course, is a virtual processor ratio of unity. By acceptable, we mean that large structures were captured, most of the flow subtleties were shown, and the animation was rapid enough to provide a smooth sense of motion. Videotape could be successfully shot directly off the screen, showing the relevant flow features. For some of the still photographs, it was often advantageous to use large numbers of particles, in the neighborhood of 100,000 particles, in order to capture some of the finest wisps and strands that develop. On an 8K machine, the speed animation is, of course, noticeably slower, with this large virtual processor ratio. However, even with frame update speeds of approximately a second, a sense of motion and continuity between frames was evident.

"It is important to point out that even when a calculation requires an especially fine grid to obtain accurate enough results, it may not be necessary to feed all those grid points to the visualization environment at once."

7. DEMONSTRATIONS

We have used the above graphics environment to study four different flows: (1) two-dimensional viscous flow over a backward-facing step (Sethian and Ghoniem, 1988); (2) two-dimensional viscous flow in a closed square (Baden and Puckett, 1989), (3) three-dimensional flow over a flat plate, and (4) three-dimensional channel flow (Kim, Moin, and Moser, 1987). All of these flows are incompressible, and extend into the turbulent regime. The physical mechanisms of vortex stretching, merging, and breaking, as well as regions of sharp change and high vorticity are present. The first three simulations are computed using vortex-type methods (discussed below), and the last is a spectral calculation. To demonstrate the use of our graphics environment, we show results from the first two flows, namely, the two-dimensional simulations. We display the results of smoke/dye injection, bubble wires, and rings of tracer particles at various points in the domain. In addition, we present the scalar stream function Ψ, where $\mathbf{u}$ is the velocity field and $\mathbf{u} = (u,v) = (\Psi_y, -\Psi_x)$. Thus, the

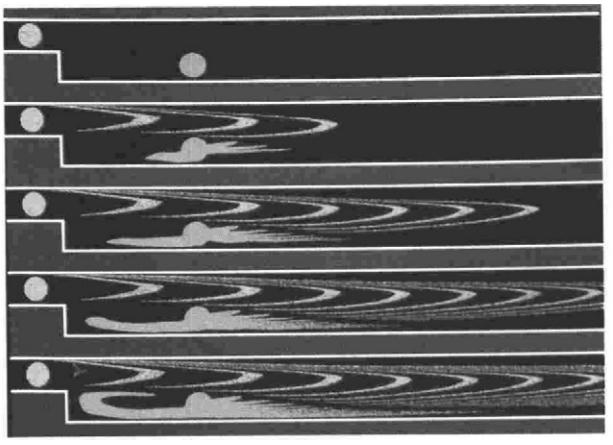

Fig. 1 Dye injection: laminar step flow (data from Sethian and Ghoniem, 1988).

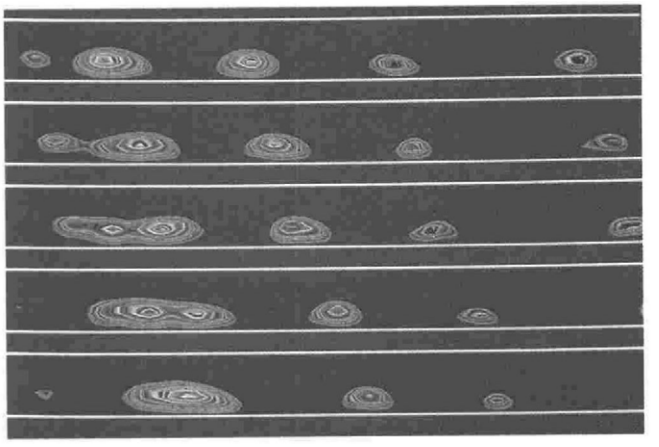

Fig. 2 Contours of stream function: transitional step flow (data from Sethian and Ghoniem, 1988).

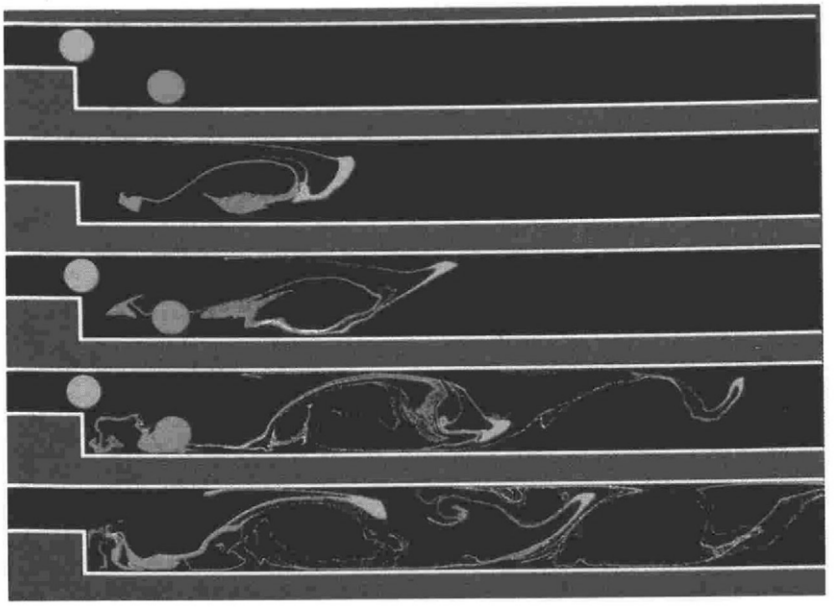

Fig. 3 Dye injection: transitional step flow (data from Sethian and Ghoniem, 1988).

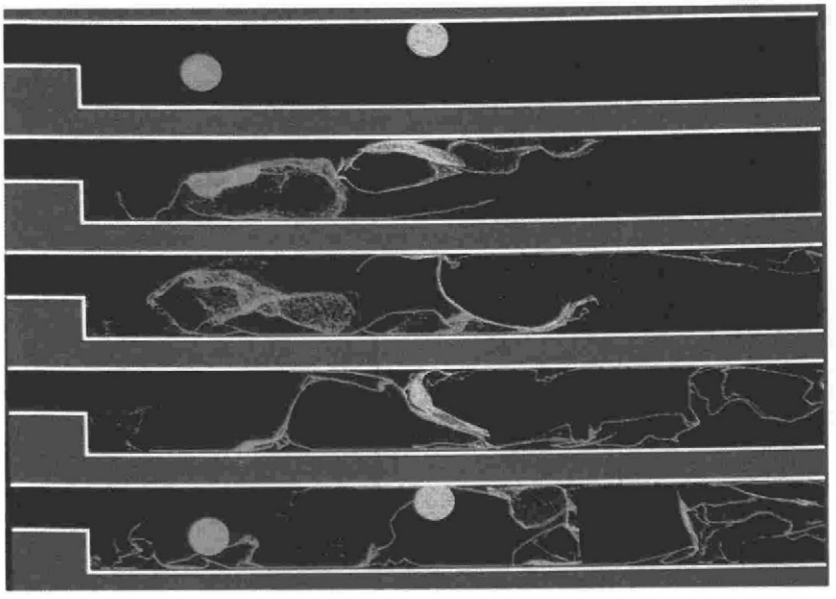

Fig. 4 Dye injection: turbulent step flow (data from Sethian and Ghoniem, 1988).

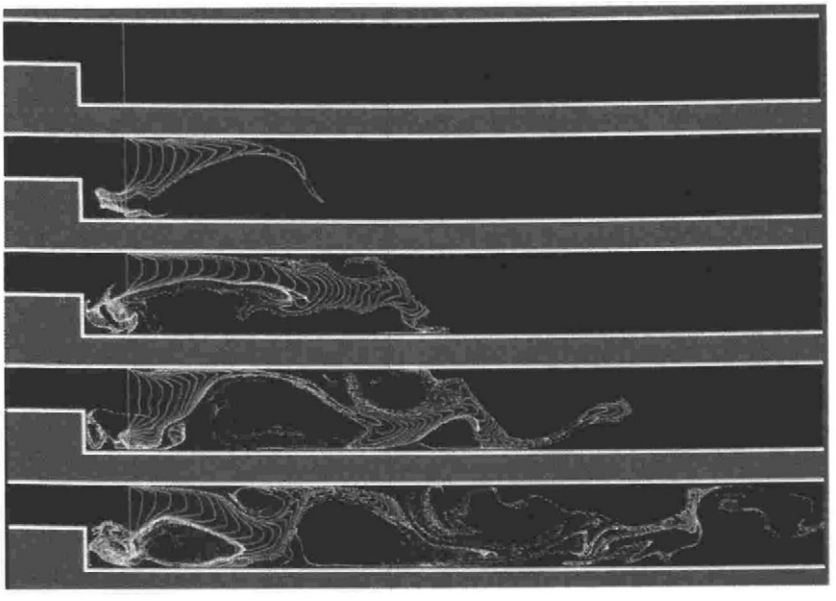

Fig. 5 Hydrogen bubble wire: turbulent step flow (data from Sethian and Ghoniem, 1988).

"Both of these calculations are two-dimensional, which is a limiting simplification of real three-dimensional flow, since such three-dimensional mechanisms as vortex stretching are ignored."

boundaries between constant colors of Ψ are parallel to the flow. Our results illustrate many of the fluid phenomena described above. Analysis of the three-dimensional calculations described above will be presented elsewhere.

Both the step flow calculation and the square flow calculation were performed using a particle-particle type of algorithm known as the random vortex method, introduced by Chorin (1973, 1978). The random vortex method is a Lagrangian particle method that relies on a discretization of the vorticity field of the flow. Vorticity, which is the curl of the velocity, describes the amount of rotation in the flow, and is a particularly appropriate approximation variable for flow which exhibits large global rotating structures. In the random vortex method, the velocity is approximated by a set of discrete vortex elements, or "blobs," whose relative positions and strengths determine the overall velocity field. The motion of the fluid is approximated by updating the position of the individual vortex elements according to the influence of the aggregate velocity field, plus a random step to model diffusion processes. Boundary conditions are met through the addition of potential flow to satisfy the normal boundary condition and a vorticity creation algorithm to satisfy the tangential no-slip condition on solid walls. This method is particularly adept at handling high Reynolds number flow (the Reynolds number is the parameter that describes the balance between inertial and viscous forces). At high Reynolds number (that is, where the flow becomes turbulent), sharp flow gradients exist in the boundary layer near solid walls and standard finite-difference or finite element methods may require a prohibitively large number of mesh points in those regions. Because the random vortex method is a grid-free method, such restrictions are considerably reduced. (For details, see Chorin, 1973, 1978; Sethian, 1984; Sethian and Ghoniem, 1988.)

Both of these calculations are two-dimensional, which is a limiting simplification of real three-dimensional flow, since such three-dimensional mechanisms as vortex stretching are ignored. (Note, however, that this is not a limitation of the numerical algorithm. For analysis of three-dimensional vortex techniques, see Chorin, 1988, 1989.) The validity of such an assumption for a specific problem has been discussed in considerable de-

tail (Sethian, 1984; Sethian and Ghoniem, 1988). However, for our purposes in this paper, the computed flows demonstrate the sort of complicated flow structures that may be effectively analyzed with this graphics environment.

Since the vortex method is a particle method that is grid-free, in both calculations the data were post-processed for display before being fed into the graphics environment. The data were generated as follows. A uniform mesh was placed on the computational domain (11 $\times$ 100 for the step, 61 $\times$ 61 for the closed square), and at each grid point the velocity and the stream function were calculated from the positions and strengths of the vortex elements. This was done every 10 time steps, and the results were written onto tape. Note that this grid plays no role in the numerical simulations; it just provides discrete temporal and spatial sampling of the flow. The data tape was then transported to Thinking Machines Corporation, where it was loaded to disk for analysis using the graphics environment.

7.1 FLOW OVER A BACKWARD-FACING STEP

A detailed convergence study of flow over a backward-facing step for a wide range of flow conditions was previously performed (Sethian and Ghoniem, 1988). The code used for this calculation was a modified version of an earlier one to study step flow (Ghoniem, Chorin, and Oppenheim, 1982). The velocity field and stream function data on a fixed grid calculated by Sethian and Ghoniem (1988) served as the input to the graphics environment described here. The domain geometry consists of a narrow inlet on the left through which flow enters. A sudden step expansion occurs, and the flow exits 20 step heights downstream on the right. The behavior of the flow depends critically on the Reynolds number, and the results of numerous laboratory experiments (Armaly et al., 1983; Denham and Patrick, 1974; Honji, 1975; Periaux, Pironneau, and Thomasset, 1976) have been described by Sethian and Ghonein (1988), and characterized by laminar, transitional, and turbulent phases.

In the laminar regime ($R < 250$), the flow is smooth, with a single, steady, and stable recirculation zone in the lower-left-hand corner. Fluid mixing is lim-

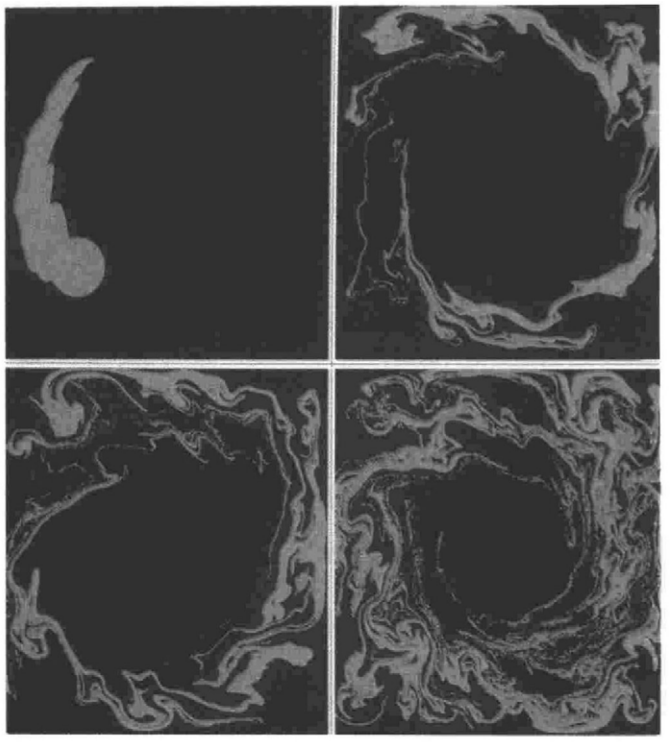

Fig. 6 Single-color dye injection: turbulent square flow (data from Baden and Puckett, 1989).

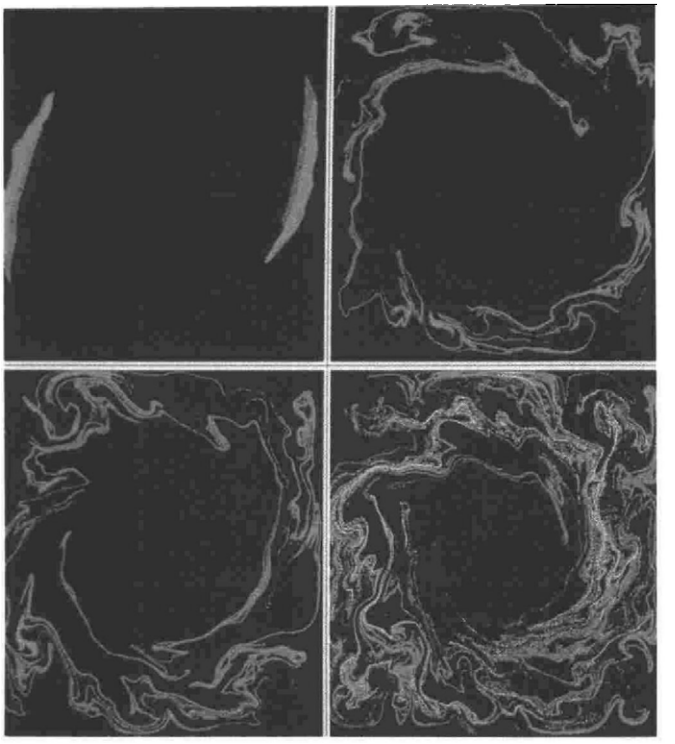

Fig. 7 Two-color dye injection: turbulent square flow (data from Baden and Puckett, 1989).

ited: smoke injection near the inlet stays confined to a narrow path through the chamber, and the flow takes the expected parabolic profile as it nears the downstream exit. Throughout the laminar regime, the length of the recirculation eddy scales linearly with increasing R. Within the next phase, the transitional regime ($R \approx$ 500–2,000), the downstream edge of the recirculation bubble tears off and moves downstream as a distinct eddy structure. When the eddy tears off, the shortened recirculation bubble begins to entrain more fluid, growing in size until another downstream eddy is shed; this mechanism repeats itself over and over. Significant mixing occurs: photographs of smoke injected into experiments with flow in the transition regime show convoluted corkscrew-type paths with fairly wide dispersal. Finally, as the Reynolds number is increased still further, experiments show that the flow enters the turbulent phase for R greater than a few thousand. Here, eddies continue to shed from the recirculation bubble near the step, but boundary layer flow along the top wall separates and rolls up into counterrotating eddies. These counterrotating eddies intersperse with those shed from the step, trapping a narrow band of rapidly moving fluid. Fluid mixing is most pronounced in the turbulent regime.

As mentioned above, these various flow portraits were previously computed using the random vortex method (Sethian and Ghoniem, 1988). The following images of the flow using our graphics environment have been presented elsewhere (Sethian, Salem, and Ghoniem, 1988). In Figure 1, we show the time evolution of multicolored smoke particles injected into laminar flow using smoke/dye injection. We periodically inject yellow particles in the inlet and purple particles along the bottom wall. The moving particles trace out the flow in good agreement with the experimental results described earlier. The parabolic profile is formed by the advected yellow particles. At the same time, the purple particles are caught up in the recirculation zone, and clearly outline the elliptical bubble near the base of the step.

In the transitional regime, the large recirculation eddy continually breaks off and sheds eddies which move downstream. In Figure 2, we show the time evolu-

"Note that the eddies seem to form in pairs: one can clearly see two eddies merge as they start to form a single eddy that moves downstream. This paired merging is difficult to discern from raw data and only becomes evident through the animation of the flow."

tion of the stream function Ψ with color map corresponding to values of Ψ. The entrance to the step has been deleted, and only the main body of the channel is shown. We have truncated values so only those values of Ψ between 0.0 and 0.35 are displayed. This isolates the rolled-up eddies as they move downstream. Note that the eddies seem to form in pairs: one can clearly see two eddies merge as they start to form a single eddy that moves downstream. This paired merging is difficult to discern from raw data and only becomes evident through the animation of the flow. Other examples of phenomena revealed through graphics have been discussed by Winkler et al. (1987). To study mixing in the transitional regime, in Figure 3, we show the time evolution of particles injected into the transitional flow using smoke/dye injection. We inject purple particles near the inlet, and blue particles near the base of the bottom wall. One can easily see the injected particles caught up in the large, clockwise rotating eddies as they move downstream. The displayed results clearly resemble smoke injection in a laboratory apparatus (see Honji, 1975).

In the turbulent flow regime, eddies are shed off the top wall as well as off the step. In Figure 4, we show the time evolution of two sets of smoke particles injected into the flow. Blue particles are injected near the bottom toward the corner, and yellow particles are injected near the middle of the top wall. The blue particles near the step are first dragged backward by the recirculation zone, and then outline a large, clockwise rotating eddy that moves downstream. Conversely, the yellow particles are sent into a large eddy that rotates in the opposite direction as it moves downstream. The two streams are caught up in different eddy structures as they move downstream and intertwine. In Figure 5, we use the bubble wire technique along a vertical line just past the inlet. This shows the development of the velocity profile of the evolving flow. We inject 200 particles in repeating sequence of red, blue, and yellow, illustrating the evolution of various lines and velocity gradients in the flow. The rotating and counterrotating vortex structures are clearly delineated by the pulsed hydrogen bubbles.

7.2 FLOW IN A CLOSED SQUARE

Two-dimensional, viscous, incompressible high Reynolds number flow in a closed square was previously modeled

using the random vortex method (Sethian, 1984). The domain was a closed rectangle. A large central vortex was placed at the center, whose strength remained constant in time. The potential flow solution to this is a steady flow in which only the normal no-flow boundary condition is satisfied. At $t = 0$, the tangential no-slip condition was instantaneously turned on, causing additional vorticity to form along the solid walls and diffuse into the interior. This problem is similar to the standard driven cavity problem in which the top wall is moved at a constant speed, as well as traditional "spin down" problems. An earlier calculation with a hybrid vortex–finite difference method was performed (Shestakov, 1979). The calculations in Sethian (1984) showed the development, growth, and diffusion of counterrotating eddies in the corners, and their effect on the large central vortex and underlying flow; in addition, the interaction between flow dynamics and flame propagation in this geometry was analyzed.

The algorithm in both this calculation and that for the backward-facing step compute the velocity field induced by the vortex elements through a direct summation technique. By this, we mean that the velocity at each of N vortex elements is computed by considering the effect of all the other vortex elements. This is, as stated, an $O(N^2)$ algorithm. Various techniques have been proposed to approximate the solution to the associated elliptic problem through fast summation techniques. One such fast summation algorithm was partitioned onto a multiprocessor with significant speedup by Baden (1987). In that work, considerable attention was given to the accompanying load-balancing issues. That multiprocessor fast summation technique was later used to repeat the above calculation of flow in a closed square (Baden and Puckett, 1989). The velocity field and stream function data on a fixed grid computed by Baden and Puckett (1989) served as the input to the graphics environment described here.

In Figure 6, we show the evolution of a large circular patch of blue dye continuously injected into a counterclockwise circulating flow. In the upper left, we see the injected fluid shortly after the no-slip condition is turned on. In the upper right, we see the swirling fluid developing rolls and oscillations as it spins around the large central vortex. In the lower left, the fluid has

"The algorithm in both this calculation and that for the backward-facing step compute the velocity field induced by the vortex elements through a direct summation technique. By this, we mean that the velocity at each of N vortex elements is computed by considering the effect of all the other vortex elements."

made several rotations around the vessel. Flow in the corners is caught up in the counterrotating eddies and outlines the large vortex structures. In the last figure (lower right), the injected fluid has stretched out considerably, concentrated in some regions and stretched into thin strands in others.

In Figure 7 we study the mixing that results from this flow in more detail by injecting two different colored dyes at opposite sides of the box. The two patches of blue and purple dye are carried by the flow and the resulting vortex structures, which grow in corners and are then shed into the flow. The fluid becomes highly mixed, as shown by the mixing of the purple and blue dyes.

8. CONCLUSION

We have built a graphics environment for analyzing the results of numerical simulations of fluid mechanics. The purpose of this environment is to provide visualization tools that mimic those in physical experiments. For velocity fields, tools include color-coded smoke/dye injection in user-defined intervals, locations, and configurations, and "hydrogen" bubble-wire tracers for determining velocity profiles. For scalar quantities, color contour maps within user-selected ranges are available. The tools were used to analyze data from vortex calculations of two-dimensional viscous flow over a backward-facing step and in a closed square, three-dimensional flow over a flat plate, and a spectral calculation of three-dimensional channel flow.

This project represents a step toward the goal of using interactive visualization as a real tool in designing numerical flow experiments. Given a precomputed set of flow data on a discrete grid, the current environment allows the user to interactively execute computationally intensive calculations as the flow visualization unfolds. Thus, the user is able to perform dozens of numerical experiments with various injection frequencies, locations, and configurations in rapid succession.

The next development might be a true interactive environment, in which the equations of motion are solved and visualized in as close to real time as possible. The immediate goal would not be the most accurate and refined calculation, but rather a coarse enough simula-

tion to proceed in close to real time while observing significant fluid mechanics. Optimally, one could experiment interactively with different design parameters, such as confinement geometries and inlet parameters. If a rough picture of the flow dynamics could be obtained, the user could quickly compare a large number of possible configurations, and select some subset for more refined, detailed calculation. Preliminary work on such an environment has begun, and will be reported elsewhere.

ACKNOWLEDGMENT

S. Baden and E. Puckett generously provided the data for the square flow calculation presented there. We also acknowledge the work of A. F. Ghoniem in developing the original step flow code. In addition, we wish to acknowledge the contributions of Jill P. Mesirov and Washington Taylor IV. The velocity field used for display was computed at the Lawrence Berkeley Laboratory, University of California, Berkeley. J. A. Sethian is partially supported by the Sloan Foundation and the Applied Mathematics Subprogram of the Office of Energy Research under contract DE-AC03-76SF00098. The images were computed at Thinking Machines Corporation, Cambridge, Massachusetts.

BIOGRAPHIES

J. A. Sethian received a Ph.D. in applied mathematics from the University of California at Berkeley in 1982. He was an NSF postdoctoral fellow at the Courant Institute of Mathematical Sciences, and is currently an associate professor of mathematics at the University of California at Berkeley.

James B. Salem received a B.S.E.E. in 1984 from the Massachusetts Institute of Technology, where he worked in the MIT Media Lab. Since then he has designed and led language and applications software projects at Thinking Machines Corporation in Cambridge. Currently, he is the director of scientific visualization software at Thinking Machines.

SUBJECT AREA EDITOR

Steven E. Follin

REFERENCES

Armaly, B. F., Durst, F., Pereira, J. C. F., and Schonung, B. 1983. *J. Fluid Mech.* 127:473.

Baden, S. B. 1987. Runtime partitioning of scientific continuum calculations running on multiprocessors. Ph.D. thesis, University of California, Berkeley.

Baden, S. B., and Puckett, E. G. 1989. A fast vortex code for computing 2-D flow in a box. *J. Comput. Phys.*, in press.

Billet, M. L., Kim, J. H., and Heidrick, T. R., eds. 1985. *International symposium on physical and numerical flow visualization.* New York: ASME, Fluid Engineering Division.

Chorin, A. J. 1973. *J. Fluid Mech.* 57:785.

Chorin, A. J. 1978. *J. Comput. Phys.* 27:428.

Chorin, A. J. 1988. Spectrum, dimension and polymer analogies in fluid turbulence. *Phys. Rev. Lett.* 60:1947–1949.

Chorin, A. J. 1989. Hairpin removal in vortex interactions. *J. Fluid Mech.*, in press.

Corke, T. C. 1984. *AIAA J.* 22:1124.

Denham, M. K., and Patrick, M. A. 1974. *Trans. Instn. Chem. Engrs.* 52:361.

Foley, J. D., and Van Dam, A. 1984. *Fundamentals of interactive computer graphics.* Reading, Mass.: Addison-Wesley.

Gad-el-Hak, M. 1987. Review of flow visualization techniques for unsteady flow. In *Flow Visualization IV, Proc. of the Fourth Internat. Symp. on Flow Visualization,* edited by C. Veret. New York: Hemisphere Publishing Co., Harper and Row.

Ghoniem, A. F., Chorin, A. J., and Oppenheim, A. K. 1982. *Philos. Trans. Roy. Soc. London Ser. A* 304:303–325.

Hillis, D. 1985. *The Connection Machine.* Cambridge: MIT Press.

Honji, H. 1975. *J. Fluid Mech.* 69:229.

Johnston, W. E., Hall, D., Huang, J., Rible, M., and Robertston, D. 1988. Distributed scientific video movie making. In *Proc. of Supercomputing '88.* New York: IEEE, Computer Society Press.

Kim, J., Moin, P., and Moser, R. 1987. *J. Fluid Mech.* 177:133.

Lorenson, W., and Cline, H. 1987. *Comput. Graphics* 21:163.

McCormick, B., DeFanti, T., and Brown, M. 1987. Visualization in scientific computing. Report to the National Science Foundation by the Panel on Graphics, Image Processing, and Workstations.

Merzkirch, W. 1987. *Flow visualization.* 2nd ed. New York: Academic Press.

Periaux, J., Pironneau, O., and Thomasset, F. 1976. *Fifth Internat. GAMM Conf. on Numerical Methods in Fluid Mechanics.* New York: Springer-Verlag, vol. 9, p. 75.

Reed, H. L. 1987. *Phys. Fluids* 30:2597.

Rogers, S. E., Buning, P. G., and Merritt, F. J. 1987. Distributed interactive graphics applications in computational fluid dynamics. *Internat. J. Supercomput. Appl.* 1(4):96–105.

Salem, J. B. 1988. *RENDER: a data parallel approach to polygon rendering. Thinking Machines Corp. Technical Report VZ88-2.

Sethian, J. A. 1984. *J. Comput. Phys.* 55:425.

Sethian, J. A. 1987. Visualization of data from nu-

merical simulations: moving surfaces and fluid flow. Videotape, Lawrence Berkeley Laboratory.

Sethian, J. A. 1989. Identifying and tracking coherent structures in turbulent flow. Technical Report. Lawrence Berkeley Laboratory.

Sethian, J. A., and Ghoniem, A. F. 1988. *J. Comput. Phys.* 74:283.

Sethian, J. A., Salem, J. B., and Ghoniem, A. F. 1988. Interactive scientific visualization and parallel display techniques. In *Proc. of supercomputer '88.* New York: IEEE, Computer Society Press.

Shestakov, A. I. 1979. *J. Comput. Phys.* 31:313.

Smarr, L. 1987. The computational science revolution: technology, methodology and sociology. In *High-speed computing: scientific applications and algorithm design,* edited by R. B. Wilhelmson. Urbana: University of Illinois Press.

Thinking Machines Corp. 1987. Connection Machine Model CM-2 technical summary. Thinking Machines Corp. Technical Report HA87-4.

Upson, C., and Keeler, M. 1988. *Comput. Graphics* 22:59.

Van Dyke, M. 1982. *An album of fluid motion.* Stanford, Calif: Parabolic Press.

Veret, C., ed. 1987. *Flow visualization IV, proc. of the fourth internat. symp. on flow visualization.* New York: Hemisphere Publishing Co., Harper and Row.

Werle, H. 1973. Hydrodynamic flow visualization. In *Annual review of fluid mechanics,* edited by M. Van Dyke, W. G. Vincenti, and J. V. Wehausen. Palo Alto, Calif.: Annual Reviews, vol. 5, p. 361.

Winkler, K. H. A., Chalmers, J. W., Hodson, S. W., Woodward, P. R., and Zabusky, N. J. 1987. *Phys. Today,* October.

Zabusky, N. J. 1987. *Phys. Today,* October.