3D Shock Wave Visualization on Unstructured Grids

Kwan-Liu Ma [*]
ICASE

John Van Rosendale [†]
ICASE

Willem Vermeer [‡]
Delft University of Technology

Abstract

A critical issue in understanding high speed flows is the study of shock waves. This paper summarizes our research on techniques for the detection and visualization of shock waves occuring in simulations of three-dimensional flows on unstructured grids. Detection algorithms based on Mach number, density gradient and directional derivatives are compared using a data set from calculations of a transonic flow with a weak double shock around an airfoil. Both surface and volume rendering techniques are used to display the shocks.

The issues in this research area are very much like those occurring in medical imaging. Since the data themselves (in this case the results of the fluid dynamics simulation) are intrinsically low resolution and noisy, properly extracting and visualizing the shock is very difficult. In this environment blurry, low-resolution techniques, like the splatting volume rendering, seem to do rather well. More complex schemes, using sophisticated numerical shock detectors coupled with polygon rendering, produce visually sharper shocks, but also introduce "graphics artifacts," which complicate understanding of the flow physics. On the other hand, visualization results produced with techniques like splatting are, in effect, relying more on the human visual system to compensate for limited resolution in the simulation.

1 Introduction

As the speed of an airplane increases in the subsonic range, the zone of disturbance expands vertically, and at a certain point the speed of sound will be exceeded in a local region, usually on the upper surface where faster flow is needed to provide lift. Still further increases of speed then lead to a shock wave, which terminates the region of supersonic flow. Such a shock wave can cause the boundary layer to separate, the drag to increase abruptly, and the flow to become unsteady. Given these consequences, a considerable effort, both experimental and numerical, has been devoted to the study of mixed subsonic-supersonic flow with embedded shocks. In this paper, we describe techniques for detecting and visualizing shock waves in computed three-dimensional flows on unstructured grids.

Determining the exact location and structure of shock waves in computed flow solutions is surprisingly difficult. Though physical shocks are very sharp, numerically computed shocks are ordinarily smeared over several grid cells, due to errors in the numerical approximation of the fluid dynamics equations. Moreover, the data is available only at the vertices of the grid, which rarely coincide with the exact shock location, so interpolation issues arise. Also, once the shock is found, there is the issue of appropriately displaying the shock in three dimensions. The use of unstructured grids, now standard in many kinds of flow calculations, introduces additional complications in both the detection and visualization of shocks.

In this paper, we investigate shock detection methods based on Mach number, density gradient, and directional derivatives. A data set from aerodynamic calculations of transonic flow over an ONERA-M6 wing is used for comparing these detection methods as well as for comparing the representations and rendering of the detected shocks. In general, the quality of the visualization results depends on both the detection and visualization method, as well as the accuracy of the interpolation and differencing schemes used. Due to limited space, this paper only provides a discussion of the detection algorithms and corresponding visualization techniques and results. The numerical formulations and implementation details of the detection and rendering algorithms for unstructured data are provided in [10].

2 Related Work

A great deal of research has been conducted on the problems of capturing and fitting shocks in two-dimensional flow simulations [7, 8, 4, 9]. However, locating shocks in three dimensions is much more complicated than it is in two dimensions. The methods used in two dimensions do not always extend well to three dimensions. In

[*]Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, Virginia 23681-0001, kma@icase.edu

[†]Visiting Scientist, ICASE; also Division of Advanced Scientific Computing, National Science Foundation, Arlington, VA 22230, jvr@nsf.gov

[‡]Department of Applied Mathematics and Informatics, Delft University of Technology, Julianalaan 132 2628 BL Delft, The Netherlands, vermeer@math.tudelft.nl

0-7803-3708-5/96 ..$4.00 © 1996 IEEE

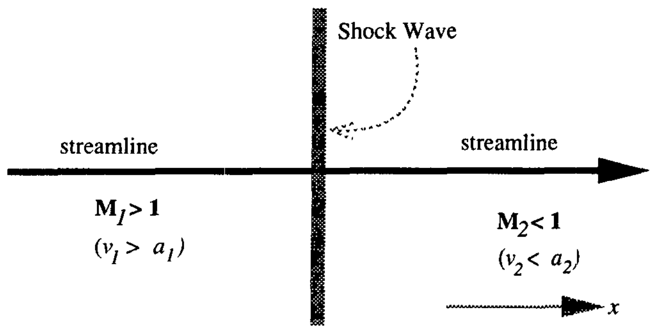
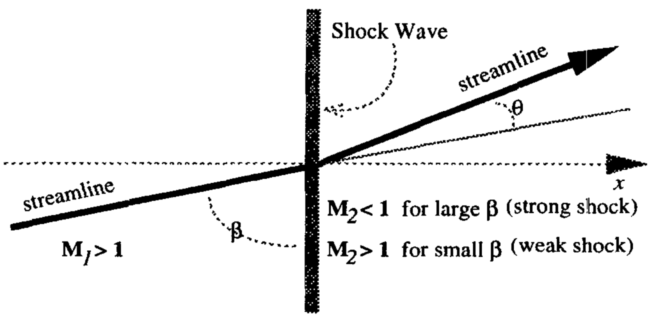


Figure 1: A one-dimensional normal shock.



Shock Wave
streamline
θ
x
streamline
β
$M_1 > 1$
$M_2 < 1$ for large β (strong shock)
$M_2 > 1$ for small β (weak shock)

Figure 2: A one-dimensional oblique shock.

particular, the grids used for three dimensional problems are necessarily coarser (and the geometries more complex), so obtaining sufficient resolution in three dimensions is always a problem.

Visualization of shock waves in three-dimensions has been addressed by a number of researchers. One approach, embodied in the visualization package Visual3, is described in [1]. With this approach, iso-surfaces of the Mach-number normal to the shock are created, using both the density gradient and Mach number. In [6], shocks are determined for hypersonic flows on *structured grids* as the zero-level iso-surfaces of the second directional derivative of the density. The shocks are rendered as partially transparent surfaces, so that the underlying aircraft structure is visible. In [9], a two-dimensional shock-fitting algorithm is presented for unstructured grids. This idea relies on comparison of density gradients between grid nodes, and can be used in three dimensions to detect shocks. The next section describes three detection methods based on some of these ideas.

3 Shocks Detection

3.1 Normal Mach number

Shocks are abrupt changes in flow field quantities such as pressure, density and velocity. In particular, the velocity component normal to the shock wave jumps from supersonic (Mach number > 1) to subsonic (Mach number < 1) as flow passes through the shock, as shown in Figure 1. The Mach number M is the ratio between flow velocity (v) and sonic velocity (a).

Consequently, the first and simplest idea that comes to mind for detecting shocks, is to connect all points in the flow where the Mach-number equals one, because by definition a shock wave marks the transition of the flow velocity from supersonic to subsonic. Unfortunately, this "sonic-surface" does not, in general, represent a shock. The normal shock wave is actually only a special case of the broader class of flow discontinuities called oblique shock waves which are found in most supersonic flows. Figure 2 shows an oblique shock. Flow passing through an oblique shock wave can remain supersonic, and conversely, there are many cases where the sonic line is not a shock.

The next idea would then be to create a surface of all points where the normal Mach number equals one. This idea is correct, but raises a difficult chicken-and-egg problem: in order to find the shock-surface, the normal direction to this surface is needed to check whether the normal Mach-number jumps from $M > 1$ to $M < 1$; however, this normal direction remains unknown until the surface has been found. We can resolve this problem by approximating the direction normal to the shock with the density gradient $\nabla\rho$, which should be normal to the shock. This idea, which is also used by PLOT3D, the CFD-plotting program developed at NASA Ames [5], is the basis for the first detection algorithm.

The algorithm works as follows. At all points in the flow domain, it computes the Mach-number M in the direction of the density gradient, defined by $M = \frac{\|\hat{v}\|}{a}$, where the vector $\|\hat{v}\|$ is the velocity v in the direction of the density gradient, or, in other word, the projection of v onto the density gradient. Now, a shock-surface can be found by connecting the points in the flow where this expression for M equals one.

3.2 Directional derivatives thresholding

In [6], density gradients in the direction of local flow field velocity are used to determine shock locations. Suppose some scalar function f, such as density, has been approximated by a numerical solution at a few grid points. A discontinuity in f generally does not coincide with one of these grid points. The discontinuity is smeared out across a few grid points, and frequently under- and overshoots appear. The location of the discontinuity can be approximated by taking the position of the steepest gradient of the curve through the data points. This can be found, in this one-dimensional example, by taking numerically the second spatial derivative of the scalar f: $\frac{d^2f}{dx^2}$ The points where the second derivative of f is zero and the first derivative $\frac{df}{dx}$ is non-zero are taken to be the position of the discontinuity.

To apply this method in three-dimensions, we can compute the derivative of the density ρ in the direction of the local velocity vector v, by taking the inner product of the density gradient $\nabla\rho$ and the normalized velocity:

$$\delta_1\rho = \frac{v}{||v||} \cdot \nabla\rho$$

That is, $\delta_1\rho$ denotes the first derivative of the density of the gas in the direction of the local velocity.

This quantity $\delta_1\rho$ supplies much information regarding the flow. First of all, for extracting shock waves, we are mostly interested in the extrema of this quantity. We will describe a method to determine these extrema shortly. Secondly, the sign of $\delta_1\rho$ indicates whether a flow is compressing or expanding. When $\delta_1\rho > 0$, the velocity and the density gradient are oriented in the same direction. This means that the gas is compressing in this region. Conversely, if $\delta_1\rho < 0$, the density increase is in the opposite direction to the velocity, which means that the gas is expanding. Since only those parts of the flow field where the flow is compressing indicate the possible presence of a shock wave, we are mainly interested in positive values of $\delta_1\rho$. Instead of looking for extrema of $\delta_1\rho$, we can restrict ourselves to the (local) maxima. This is the basic idea behind this detection algorithm.

The shock can now be located by searching for extrema of the quantity $\delta_1\rho$. This can be accomplished by determining zero-values of its directional derivative $\delta_2\rho$, defined as:

$$\delta_2\rho = \frac{v}{||v||} . \nabla\delta_1\rho = \frac{v}{||v||} . \nabla\left(\frac{v}{||v||} . \nabla\rho\right)$$

This is just the second directional derivative of the density ρ. We have seen from the one-dimensional example explained above that a discontinuity in density can be found by looking for zero values of the second derivative. In other words, we can now find shocks by constructing zero-level iso-surfaces of this second directional derivative $\delta_2\rho$.

There is, however, a difficulty with this approach. Although it is true that the second directional derivative $\delta_2\rho$ is zero at the shock, the converse is, in general, not true. Unfortunately, the quantity $\delta_2\rho$ also vanishes in smooth regions with few disturbances, causing erroneous shock detection. Therefore, we have to somehow select those parts of the $\delta_2\rho =$ zero-surface which coincide with a shock and discard those parts where no actual shock is present. In [6] it is shown that this can be done by thresholding the first derivative $\delta_1\rho$ by some constant $\epsilon > 0$, thus filtering those regions where the flow is relatively undisturbed. The procedure to find the shock thus becomes:

1. compute both $\delta_1\rho$ and $\delta_2\rho$ at all points in the flow
2. construct a zero-level iso-surface of the quantity $\delta_2\rho$
3. discard those parts of this surface where $\delta_1\rho < \epsilon$

This approach can be problematic, since the value of $\epsilon > 0$ critically influences the quality of the results. Setting ϵ too large creates numerous holes in the shock surface, since $\delta_1\rho$ can be fairly small at many intersection triangles on the shock. Conversely, setting ϵ too small causes spurious shocks to be found in the smooth regions of the flow field. Setting a value for ϵ is therefore a problematic trial-and-error procedure.

Hence, we have used another criterion to decide which parts of the $\delta_2\rho =$ *zero surface* coincide with a shock. Instead of looking at values of $\delta_1\rho$, we used the normal Mach number to decide which parts of the surface should be discarded. We have seen previously that the normal Mach-number (the component of the Mach-number in the direction of the density gradient) should cross one across a shock. We can use this to improve step 3 in the above algorithm. For numerical reasons, we cannot expect the normal Mach number to be exactly one in a shock, so again we have to look for values in a small neighborhood of one. The procedure to find a shock thus now becomes:

1. perform the first two steps of the above algorithm
2. discard those parts of the surface where the *normal* Mach-number is not close enough to one

Again, there is the issue of how to choose this neighborhood of one. However, when this neighborhood has been chosen properly, this algorithm produces, in some cases, better results than the algorithm which uses $\delta_1\rho$ to decide which parts of the surface coincide with the shock. This will be demonstrated in Section 4, where we compare the behavior of these algorithms.

3.3 Weighted density gradient

In [9], an adaptive method combining the advantages of shock-capturing and shock-fitting is described. The shock detector using density gradients at the grid points warps the computational grid to yield shock-fitted accuracy. A grid point is in the vicinity of a shock if its density gradient exceeds a weighted average of density gradients of its neighboring grid points. All such grid points are then transformed closer to the shock using an attraction force, which depends on the local grid size and the density in neighboring points. Next, the solution is recomputed using the new grid and in this way the grid is iteratively adapted to the shocks. Using grid-control functions to prevent the generation of very thin grid cells, the grid is ultimately fitted to the shock. This technique has been successfully applied in two dimensions for the adaptation of grids to shock locations while the flow simulation is marching to steady state.

Extending the same algorithm to three dimensions would imply the implementation of a three-dimensional flow solver. Since the goal of this research is to find shock waves in given, precomputed data sets, we use the core of the algorithm, which is easily extended to three dimensions, as the basis for a shock detection algorithm.

The crucial part of the algorithm described in [9] is the part where it is determined which points are in the vicinity of a shock. This is done with the following algorithm, which is performed at each grid point n:

1. compute a weighted average of the norm of the density gradients at the points neighboring n.
2. compare this weighted average with the norm of the density gradient $||\nabla\rho_n||$ at n itself. If $||\nabla\rho_n||$ exceeds the weighted average of the neighboring points, point n is assumed to be in the vicinity of the shock.

This idea can be used in three dimensions as well. We will now discuss this method in more detail.

The weighted average at a grid point n is computed using the following weight function:

$$w_a = |(\vec{a} - \vec{n}) \cdot \nabla\rho_n|$$

where $\vec{n}$ is the location vector of point n, and $\vec{a}$ is the location vector of a neighboring grid point a. This quantity w_a is thus the weight of neighbor a at node n. The motivation to this weighting is the assumption that the density gradient will be normal to the shock surface; therefore, neighboring nodes located in the direction of the density gradient are assigned relatively large weights. The weighted average c_n is now defined as:

$$c_n = \frac{\sum_{neighbors} w_a ||\nabla\rho_a||}{\sum_{neighbors} w_a}$$

Those points n where $||\nabla\rho_n||$ exceeds c_n are assumed to be in the vicinity of a shock. At those points, the difference $||\nabla\rho_n|| - c_n$ is computed. This difference can now be used to search for shocks. Unlike the detection algorithms discussed in the previous sections, this algorithm does not produce a surface which can be visualized using surface-rendering techniques. A direct volume rendering seems more appropriate, as we will demonstrate later.

The disadvantage of this algorithm is that it is not able to distinguish shocks from other regions where high density gradients occur, for instance at the nose or tip of the wing. In order to improve this algorithm and achieve accurate detection, this scheme would need to use additional flow-field characteristics to properly locate the shocks.

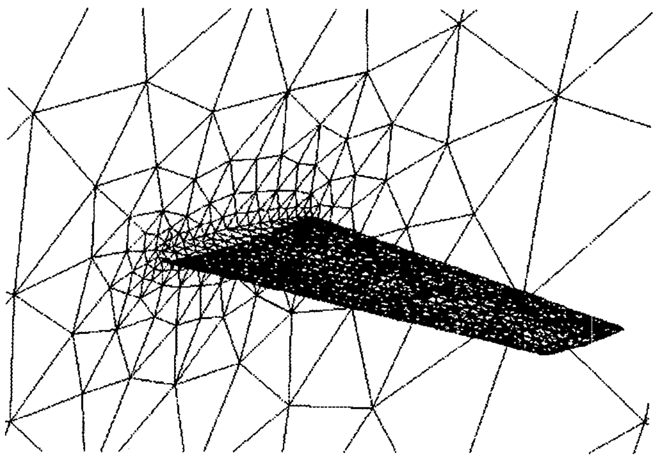


Figure 3: Boundary grids of the ONERA-M6 wing.

4 Visualization Results

For comparison, we apply the detection algorithms described in the previous section to the same data set. Although it is physically impossible to have discontinuities in fluid properties, the normal shock is nearly discontinuous. The thickness of a shock is in the order of 10^{-5}in., roughly four times the *mean free path* of the gas molecules. The detected shocks are thus represented as three-dimensional triangular meshes which can be efficiently rendered with a high-performance graphics workstation, like an SGI Indigo 2. A hardware splatting technique [2] is also used to derive shock surfaces as semi-transparent cloud, which may very well be a better approximation to the physical nature of a shock.

The data set was generated from a multigrid computation on a transonic flow over an ONERA-M6 wing with free-stream Mach number 0.86 and 3.06 degrees angle of attack. This is a well-known test-case in aerospace science. The multigrid solver is described in detail in [3]. The data set stores density, momentum and energy at the grid nodes. The grid consists of 53961 nodes and 287962 tetrahedral elements. Figure 3 depicts the grid on the wing and part of the symmetry surface of the problem domain. Due to the high grid density on the wing (over 20000 boundary elements), the wing shows up almost as a black solid shape.

On top of the wing, one expects to see a double shock arising in a V-shaped pattern. In the color pictures to be presented, the wing shows up as a grey body and the shock-wave surfaces are shaded according to Mach number. Depending on the detection algorithm used, scalar quantities representing some aspect of a shock, such as shock strength, can also be used to shade the surface. To facilitate the comparison, all visualization results are shown from the same viewpoint with the same lighting settings.

Figure 4 (Color Plate 1) shows an iso-surface of the

Mach-number in the direction of the density gradient with Mach-number mapped onto it. The iso-level is one (sonic level). The surfaces can be constructed by using a "marching tetrahedra" algorithm, which linearly interpolates vertex-values on all six edges of a tetrahedron to find possible iso-level intersection points. A tetrahedron can either have three or four such intersection points. In the latter case, the corresponding intersection quadrilateral is split into two triangles to allow for easier rendering.

This detection algorithm gives a good indication of the shock location, though it misses a part of the first shock, which in reality extends all the way to the left-hand side of the wing. In addition, the region on the right-hand side of the wing is quite unclear. The two shocks seem to blend together over some distance and it is not clear in this region where the shocks actually meet. However, taking the relative simplicity of the detection algorithm used into account, this picture is a very reasonable result.

Figure 5 (Color Plate 2) displays the result of the algorithm described in Section 3.2. The zero-level iso-surface of the second directional derivative is thresholded against the first directional derivative $\delta_1\rho$. the value chosen for the thresholding is $\delta_1\rho > 1$ on a scale of $-34.8 < \delta_1\rho < 17.6$. this value was chosen by a trial and error procedure. When a smaller value for the thresholding is selected, the shock which appears green in this figure, starts to "grow" in vertical direction, making it appear larger than it really is. Also, in other regions, surfaces start to appear which do not correspond to shock waves. There are the regions in the flow where both second and first directional derivatives are almost zero. As we know from the one-dimensional example described in Section 3.2, these regions do not necessarily correspond to extrema of the density gradient. Conversely, when a larger value for the thresholding is used, the surfaces develop holes or are excluded altogether.

Note that this algorithm erroneously detects a shock, color blue, in front of the wing. This is a region of strong compression, but not a shock. On the other hand, the first shock on top of the wing now extends all the way across the wing. While this algorithm does a better job of defining the actual shocks, it also detects erroneous shocks.

Figure 6 (Color Plate 3) was created using the same algorithm as Figure 5, but with different thresholding. In this case, the normal Mach-number was thresholded for values larger than 0.95, meaning that triangles with normal Mach-number smaller than 0.95 were excluded. Compared with the previous Figure 4 and 5, fewer erroneous surfaces are detected, but the first shock again does not extend to the left-edge of the wing.

In Figure 4, 5, and 6, the detected surfaces have a jagged look about their edges. Especially the surfaces in Figure 5 and 6, which are thresholded iso-surfaces, are not very smooth at all. This is a result of the linear interpolation used to construct the surfaces. The algorithms do not really construct surfaces; each tetrahedron is tested separately for an intersection with the iso-surface. If this intersection polygon meets the threshold criterion, the polygon is included in the surface. Neighboring intersection triangles that meet the threshold criterion jointly form the surface. Therefore, the border of a surface is formed more or less arbitrarily, depending on the shape of the triangles at the edges of the surface. Improving the smoothness of the surface border will be a difficult procedure because one would first have to find out which triangles of the surface are boundary triangles. If this is known, a smoothing procedure could be applied to those boundary triangles. One would have to be careful to ensure, however, that the resulting surface is still a valid representation of the shock, not just of a visualization trick.

Finally, Figure 7 (Color Plate 4) is the result of the splatting technique. The opacity of the splats is set according to positive values of the first directional density derivative $\delta_1\rho$ using a linear transfer function. As was explained in Section 3.2, $\delta_1\rho$ is an important indicator of possible shock locations. Positive values of $\delta_1\rho$ indicate a compression in the flow, while negative values correspond to an expansion. Therefore, if we discard those regions in the flow where $\delta_1\rho < 0$ and only use the positive values of $\delta_1\rho$ for the opacity mapping, splats will only be drawn in compression regions.

The color of the splats is set according to the Mach number, in order to compare to the previous three images. This seems to be the image that best displays the shocks, although they may be somewhat too fuzzy. If the opacity of the splats is increased, other regions in the flow where $\delta_1\rho > 0$ start to appear, notably in the region where Figure 5 finds the blue surface. This is unavoidable since the compression occurring in this region results in a positive value of $\delta_1\rho$.

5 An Interactive Shock Analysis System

From the above discussion, it is clear that there is no single best shock detection (and visualization) algorithm allowing us to implement a generally accurate, automated computational technique for locating (and visualizing) three-dimensional shock waves. Therefore, we have implemented an interactive shock analysis environment which incorporates multiple detection and visualization schemes. An interactive setting allows us to experiment with different detection and visualization methods in a more efficient manner and to derive useful visualization results more quickly for analysis.

This interactive visualization system is implemented in C++, Motif and GL for running on an SGI workstation. At present, it reads data in PLOT3D file format. Important considerations for implementing such system include:

- simple and friendly user interface
- interactivity
- flexibility

The graphical user interface is greatly simplified by keeping on screen only those interface objects that are currently needed. Interactivity is achieved by making use of the SGI's excellent hardware rendering support and by intelligently reducing the number of graphics primitives that must be processed.

Flexibility means the ability to use different detection schemes and add new ones. The current system allows the user to experiment with different detection schemes based on predefined scalars. That is, a particular detection scheme is normally not hard-coded. Instead, a detection process is completed by selecting multiple individual scalar variables for mapping to appropriate graphics primitives and transfer functions. Therefore, a user of such system is usually the scientist who performs the numerical flow simulation and is knowledgeable in shock detection.

In addition to the five variables read from the solution data file, which are:

- density
- velocity x component
- velocity y component
- velocity z component
- energy

there are a number of predefined scalars that can be computed using the flow solution variables. These scalars are:

- Mach number
- speed of sound
- first directional derivative
- second directional derivative
- normal Mach number
- pressure

The modularity of the C++ source program makes the insertion of new scalar calculations straightforward. Besides, several thresholding capabilities are provided for filtering out certain data items.

As to the visualization capabilities, the current system provides hardware surface rendering and splatting so it is capable of displaying cutting planes, isosurfaces and semi-transparent volumes. The interactivity of the system is limited by the processing power and memory capacity of the SGI workstation used. There is no support for handling exceptionally large data sets. Thus, for example, interactive visualization of a data set with millions of grid cells would be impossible now on an SGI Indigo 2 with 128 megabytes of memory.

5.1 Calculations of derived quantities

For shock analysis, it is necessary to compute derived quantities such as Mach number, pressure or gradients. In particular, computing gradients accurately on unstructured grids is less straightforward than it is on structured grids. One effective and frequently used approach is to compute a divergence theorem surface integral at a point P, with the polyhedron formed by its neighboring points as a control-volume V:

$$\frac{\partial\rho}{\partial x} \approx \frac{1}{V}\int\int_S \rho \cdot n_x dS$$

where S denotes the surface of V and n_x the x-component of the outward normal to S. The components $\frac{\partial\rho}{\partial y}$ and $\frac{\partial\rho}{\partial x}$ are found analogously to jointly form

$$\nabla\rho = (\frac{\partial\rho}{\partial x}, \frac{\partial\rho}{\partial y}, \frac{\partial\rho}{\partial z}).$$

This approximation was used because it is exact for linear functions.

Let $T_1, ..., T_n$ be the set of terahedra forming the polyhedral control-volume V for some point P. The outer surface of V is composed of the outer faces S_i of each tetrahedron T_i. These faces S_i are defined by their three vertices v_{i1}, v_{i2}, v_{i3}. Using this notation, the surface integral can be approximated by:

$$\frac{\sum_{j=1}^{n} ||S_i|| \rho S_i n_x}{\sum_{i=1}^{n} ||T_i||}$$

where $||T_i||$ denotes the volume of tetrahedron T_i and $||S_i||$ the surface area of S_i. The density on the surface S_i is approximated by taking the average of the density in the three vertices of the surface:

$$\rho S_i = \frac{1}{3}(\rho_{v_{i1}} + \rho_{v_{i2}} + \rho_{v_{i3}}).$$

The outward normal $n = (n_x, n_y, n_z)$ can be computed from the coordinates of the points, with the additional constraint that it should point away from P.

The actual computation of this expression is still quite complicated. One major problem is that we have to know which points neighbor point V, information usually not directly available. Moreover, when this computation is performed at all points in the field, the volume of the tetrahedra and the surface areas will be computed several times at each point. It is clear that this approach would lead to a very slow algorithm.

This approach can be much improved by looping over all tetrahedra instead of over all points P. At each tetrahedron we compute the contribution of each vertex of this cell to the integral and store this at the vertices. Also we compute the volume of the cell once and store this at its vertices. When this has been done for all tetrahedra, we loop once more over all vertices and add up the contributions to the integral of all tetrahedra at each point V and divide this by the sum of the volumes of the tetrahedra. Special care has to be taken for boundary-cells, in which case the additional contribution of the boundary face should be added to the vertices. Note that the current implementation handles vertex-based data only. For cell-centered data, data at vertices can be calculated by interpolation, though this introduces additional interpolation errors.

5.2 Splatting for unstructured-grid data

Splatting was first introduced by Westover [11] and has been used as a fast approximation technique for rendering data on uniformly-spaced rectilinear grids. An image is formed by determining the screen space contribution of each grid point–a footprint–and compositing the footprints on top of each other in the visibility order. For parallel projection, a single footprint table can be pre-calculated and shared by all the voxels.

Applying splatting to unstructured-grid data allows us to ignore the type of computational cells we are dealing with. However, because of the unstructured nature of the grid, a separate footprint must be constructed for each grid point. Using parallel projection, further approximation has been taken by always representing a footprint with a circle. So each footprint is now defined by the scalar value (e.g. density or pressure) and coordinates of the corresponding grid point, and a radius value which is the average distance from the point to all other immediately neighboring points. In this way, we can approximate each footprint, for example, as an octagon, with a set of hardware Gouraud-shaded triangles as described in [2]. Compositing is done with the hardware blending support.

The multiple levels of approximation taken certainly degrade the quality and accuracy of visualization results. The goal is to have a quick view of the data and the detected shock. Although our hardware splatting approach provides a crude approximation of the actual physical phenomena, it gives the viewer a pretty good impression about the size, shape, location and corresponding scalar quantities of the shock. After using splatting to preselect viewing and rendering parameters, we can always apply more accurate rendering methods such as a ray-casting or cell-projection algorithm to generate high-quality visualization results for further analysis of the extracted data.

6 Conclusions

Summarizing our previous discussion, if the direction of the density gradient is sufficiently accurate, the first two algorithms described in Section 3.1 and 3.2 give a good indication of the shock location. Using directional derivatives to find shocks yields satisfactory results, when used in combination with a proper thresholding mechanism.

Note that the performance of the algorithms relying on the direction of the density gradient is affected by the accuracy of the gradient computation. When the direction is too much off, it becomes too unreliable to use as the basis of a detection scheme. Finally, the weighted average scheme is useful for locating regions with high density gradients, but when interpreting the results of this algorithm one has to keep in mind that not every region with high density gradients corresponds to a shock.

Splatting is a natural way of rendering shock surfaces. Using the positive first directional derivative to set the opacity of the splats will show the shocks more clearly than using, for instance, the norm of the density gradient, because discarding the negative values of the directional derivative ensures that regions where the flow expands do not contribute to the opacity of the splats.

It can be hard to grasp the three-dimensional set-up of a visualization result by looking at a single picture. Although the three-dimensional perception can be improved by lighting the surfaces or combining the surfaces with, for example, cutting planes, the best way to get a good idea of the three-dimensional structure of the flow field and its features such as shocks is to interactively examine the flow field by doing real-time rotations, translations and magnifications. In this research, a shock analysis system has been implemented and different approaches for shock detection and visualization can be experimented with in this interactive, visual environment. Details and features of this system can be found in [10].

References

[1] Haimes, R., and Darmofal, D. Visualization in Computational Fluid Dynamics: A Case Study.

In *Proceedings of the Visualization '91 Conference* (1991), pp. 392–397.

[2] LAUR, D., AND HANRAHAN, P. Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering. In *Proceedings of SIGGRAPH '91 (Las Vegas, Nevada, July 29-August 2, 1991)*, pp. 285–288. In Computer Graphics 25, 4 (July 1991).

[3] MAVRIPLIS, D. Three Dimensional Unstructured Multigrid For The Euler Equations. Tech. Rep. ICASE Report No. 91-41, Institute for Computer Applications in Science and Engineering, 1991.

[4] MORTON, K., AND RUDGYARD, M. *Lecture notes in Physics.* Springer Verlag, 1989, ch. Shock Recovery and the Cell Vertex Scheme for the Steady Euler Equations, pp. 424–428.

[5] NASA NUMERICAL AERODYNAMICS SIMULATION DIVISION. *PLOT3D User's Manual*, 1990. Version 3.6.

[6] PAGENDARM, H.-G., AND SEITZ, B. *Scientific Visualization - Advanced Software Techniques.* Ellis Hortwood, New York, 1993, ch. An Algorithm for Detection and Visualization of Discontinuities in Scientific Data Fields Applied to Flow Data with Shock Waves, pp. 161–177.

[7] SALAS, M. D. Shock-Fitting Method for Complicated Two-Dimensional Supersonic Flows. *AIAA Journal 14*, 5 (August 1976), 583–588.

[8] VAN LEER, B. The Computation of Steady Solutions to the Euler Equations: A perspective. Tech. rep., University of Michigan, 1986. Invited lecture at GAMM workshop, June 10-13 1986, France.

[9] VAN ROSENDALE, J. Floating Shock Fitting via Lagrangian Adaptive Meshes. Tech. Rep. ICASE Report No. 94-89, Institute for Computer Applications in Science and Engineering, 1994.

[10] VERMEER, W. Three-dimensional Visualization of Shock Wave Computations on Unstructured Grids. Master's thesis, Delft University of Technology, October 1995.

[11] WESTOVER, L. Footprint Evaluation for Volume Rendering. In *Proceedings of SIGGRAPH '90 (Dallas, Texas, August 6-10, 1990)*, pp. 267–276. In Computer Graphics 24, 4 (July 1990).

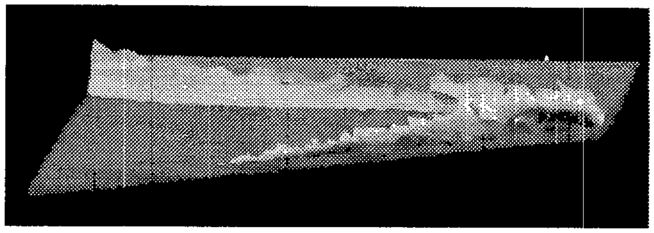


Figure 4: Normal Mach number using surface rendering.

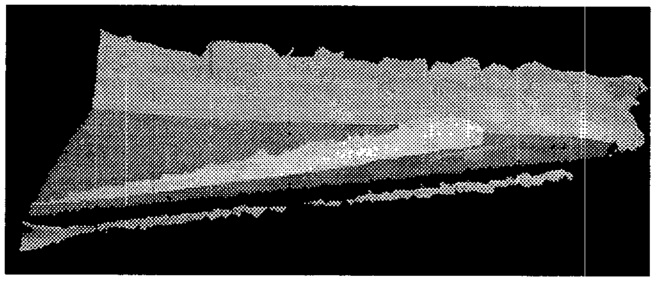


Figure 5: $\delta_2\rho$, thresholded by $\delta_1\rho$

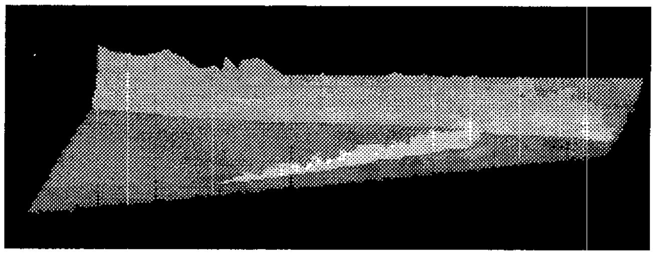


Figure 6: $\delta_2\rho$, thresholded by normal Mach number

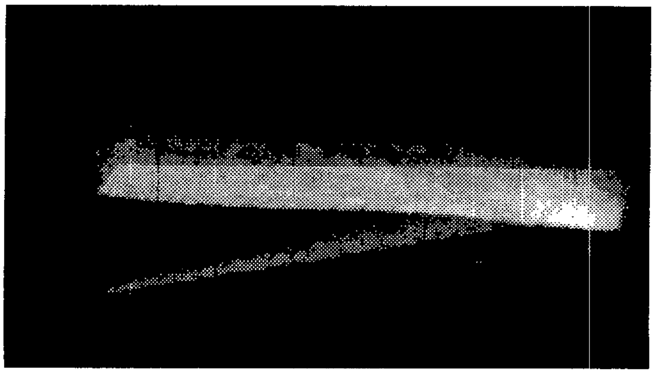
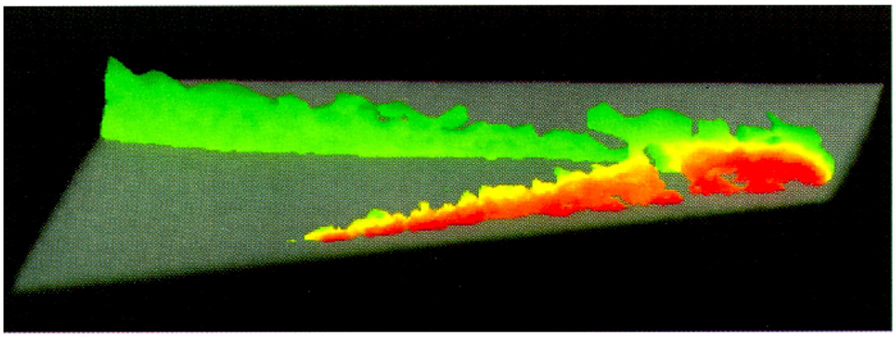
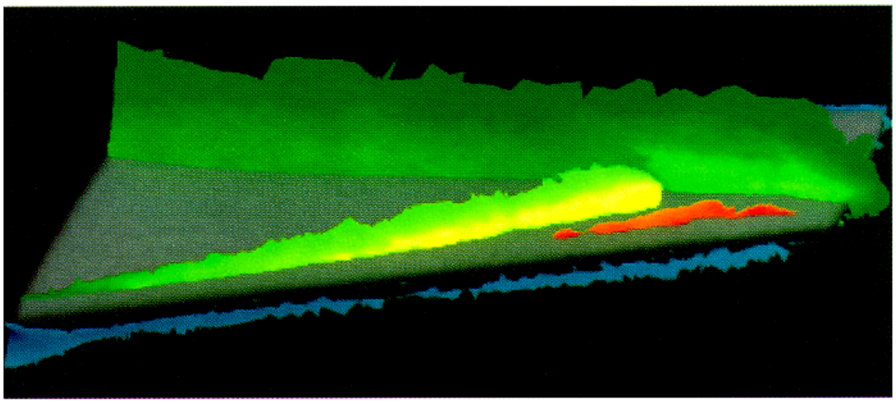
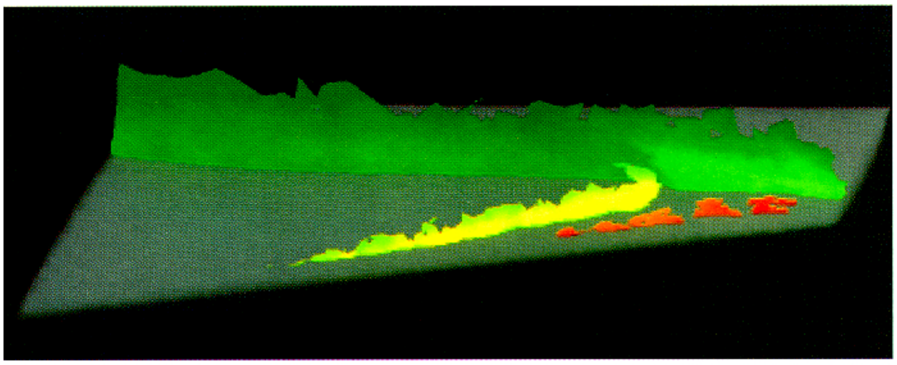
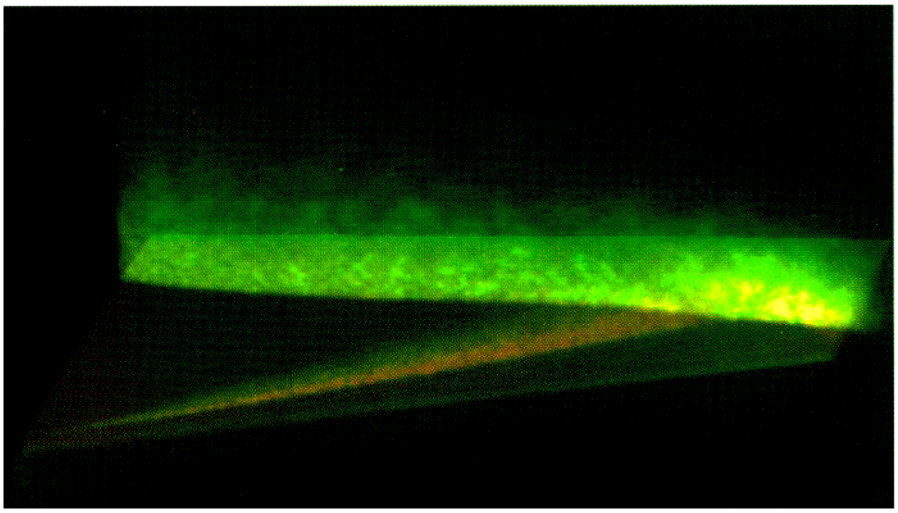


Figure 7: $\delta_1\rho$ and Mach number using splatting


Color Plate 1:
Normal Mach number
using surface rendering.


Color Plate 2:
2nd directional derivative
thresholded by 1st
direction derivative.


Color Plate 3:
2nd directional derivative
thresholded by normal
Mach number.


Color Plate 4:
1st directional derivative
(opacity) and Mach
number (color) using
splatting.