A Computer Visualization System for Multiple Submerged Buoyant Jets from Ocean Outfalls

Cheung King Bong, Sebastian

A dissertation submitted
in partial fulfilment of the requirements for
the degree of Master of Philosophy
at the University of Hong Kong

August 2000

A Computer Visualization System for Multiple Submerged Buoyant Jets from Ocean Outfalls

Cheung King Bong, Sebastian

A dissertation submitted
in partial fulfilment of the requirements for
the degree of Master of Philosophy
at the University of Hong Kong

August 2000

Abstract of thesis entitled
"A Computer Visualization System for Multiple Submerged Buoyant Jets from Ocean Outfalls"

Submitted by Cheung King Bong, Sebastian

for the degree of Master of Philosophy
at The University of Hong Kong in August 2000

Sewage and industrial effluents from coastal cities are often discharged into the adjacent sea after some land-based treatment. In modern design, the wastewater is often discharged in buoyant jet groups from risers mounted on a submarine outfall on the seabed to achieve rapid mixing of effluents with tidal flow. In tradition, the submerged ocean outfall is built in scaled models to see the effect of its design. However, building real life models is time-consuming and frequent changes of design cannot be performed. There is a need for an interactive program to visualize the design and allow quick response to changes.

In this thesis, a system for visualizing the ocean sewage discharge based on a mathematical model for buoyant jets in currents will be presented. We will discuss the design, features and implementation of the system in detail. One of the contributions of this thesis is the computer visualization system developed, called VISJET. This system also illustrates how computer graphics and visualization techniques can be used to help solving real life problems. This helps the engineers to visualize data and understand the interaction among multiple buoyant jets.

Moreover, in this thesis, an in-depth discussion on transparency rendering will be presented. When multiple jets are displayed, their trajectories tend to appear in a cluster and may occlude each other. The mutual occlusion problem can be eased if some of the jets are rendered in a semi-transparent manner. Transparency is an important feature in rendering surfaces as we can see through transparent or semi-transparent objects. An idea employing both octree and binary space partitioning (BSP) tree, which are used separately in tradition, has been implemented and the performance will also be described.

Contents

Chapter 1

Introduction

1.1 Background

Sewage and industrial effluents from coastal cities are often discharged into the adjacent sea after some land-based treatment. As shown in figure 1.1, sewage and industrial effluents are produced and carried by the underground pipes. Afterward, the effluents are processed in the land-based treatment plant before discharging into the adjacent sea.

In modern designs, the wastewater is often discharged in the form of buoyant jet groups from risers mounted on a submarine outfall on the seabed. A buoyant jet is a continuous and steady discharge with an initial momentum. Rapid mixing of the effluents with tidal flow can be achieved by jet group design so that the water quality can be controlled to within acceptable levels. The hydraulics of submerged buoyant jets is intimately related to the design of effective wastewater disposal systems.

In tradition, the submerged ocean outfall was built in scaled models, with plastic or concrete, to see the effect of its design. However, building real life models was time-consuming and frequent changes of design could not be performed. There is a need for an interactive and low-cost software system to visualize the design and allow quick response to changes.

In order to cater for this need, a visualization system called VISJET is developed. VISJET [4] is a visualization system for visualizing ocean sewage discharge

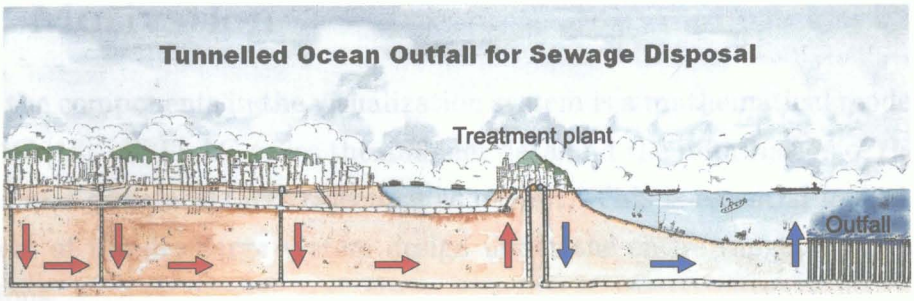

Figure 1.1: Sewage and industrial effluents (red arrow) are produced and carried by the underground pipes. The effluents are processed in the land-based treatment plant. The processed effluents (blue arrow) discharging into the adjacent sea.

based on a mathematical model for buoyant jets in currents. It is a visual tool to portray the evolution and interaction of multiple buoyant jets discharged at different angles to ambient tidal current.

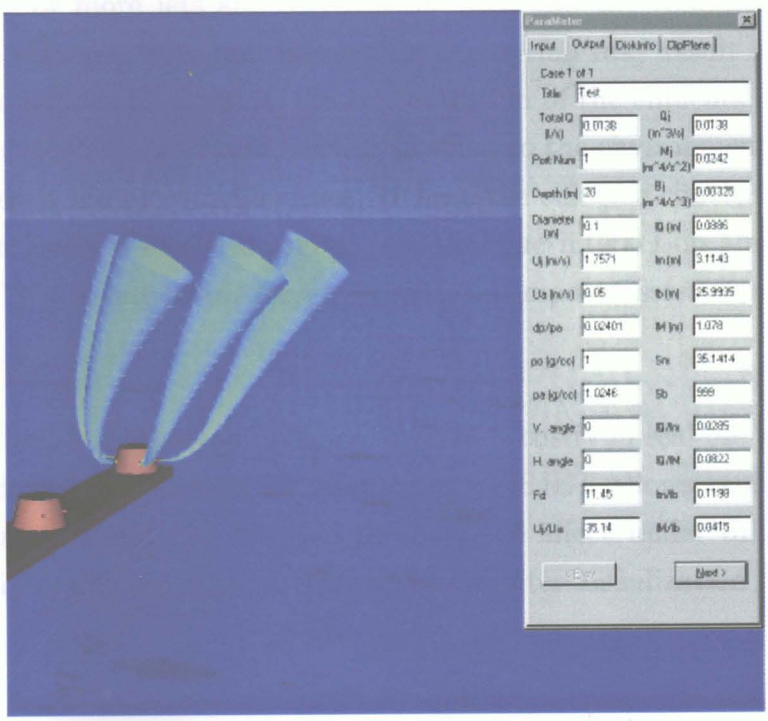

Figure 1.2: A snapshot from VISET.

VISJET is the resulting system to be discussed in the following chapters and forms the basis of this thesis.

1.2 Motivation

One of the components in the visualization system is a mathematical model called JETLAG [16]. JETLAG takes the ambient data and the information of the buoyant jets and gives a 3D jet trajectory as output. This is essential for predicting the effect of the discharge system design under the entire range of the ambient conditions.

Traditionally, the data generated by JETLAG was displayed in numbers or in two-dimensional graphs. However, the engineers cannot see the effect easily by just looking at the numerical data produced by the JETLAG model. VISJET can be used as a visual tool to depict clearly the evolution and interaction of multiple buoyant jets discharged at different angles to ambient tidal current.

Secondly, JETLAG model does not produce the result of interaction among multiple jets and VISJET provides a feature which computes the overlapping region of two or more jets and displays the area and the concentration of the region. The effluents from multiple jets in the outfall may result in mixing together in the region. As a result, the concentration of the effluents may rise to a dangerous level which can cause hazardous effect to the environment and the living organisms in the surrounding area. To keep the effluents concentration below some certain levels is essential and to visualize the interaction among multiple is helpful. This is important for environmental impact assessment and outfall design.

Thirdly, building a sewage discharge system is costly in real life and changing the design parameters cannot be interactive. VISJET allows rapid changes of design parameters and can be used to show the effect under different design parameters interactively. Also, setting up a computer visualization system is less costly.

1.3 Contribution

The main contributions of this thesis are:

- VISJET, the computer visualization system.

- To show how computer graphics and visualization techniques can be used in engineering and solving real life problems.
- To discuss how to develop a computer visualization system for visualizing the submerged buoyant jets. This includes the design of a computer visualization system, the features in the system, the problems encountered when developing the system and the approaches we used to solve these problems.
- To discuss the issues on transparency rendering and computation of overlapping region.

1.4 Outline of the thesis

In this chapter, we have introduced the computer visualization system to be discussed. The background and motivation to develop the system have also been presented. In the following chapters, we will organize the contents as follows:

- Chapter 2 will discuss the previous work in scientific visualization. We will discuss other systems related to visualizing ocean data and discuss the differences between those systems and ours.
- Chapter 3 will describe the overview of our computer visualization system. This will include the main components in our system.
- Chapter 4 will present the detailed discussion of the features in the visualization system.
- Chapter 5 and 6 will present two problems – transparency rendering and computation of overlapping region – encountered when developing our system and our approach to solve these problems will be discussed.
- Chapter 7 will present the issues which can be leaded to further research. These issues include the use of color in representing the concentration of the effluents and how to increase the realism of movement of fluids.
- In Chapter 8, the conclusion for the thesis will be presented.

Chapter 2

Previous Work

2.1 Scientific Visualization

Scientific visualization has long been one of the most significant research topics in computer graphics. The basic idea of scientific visualization is to use computer-generated pictures to display information and perform data analysis. Computer-aided design, finite element analysis, simulation, and computational fluid dynamics analysis are just few examples to which visualization can be beneficial.

2.2 Visualization Systems on Ocean Data

Recently, visualization on environmental data has become a hot topic. Xiao et al. [31] discussed the challenges in visualization of environmental data. Several systems have been developed for visualizing environmental data. A system for visualizing rainfall events has been developed in [27] and information on the mesoscale of rainstorms is obtained. Nations et al. [19] developed an interactive system of ocean circulation models. The traditional software for ocean modeling was a batch job and the data generated would be written to a file. The data generation and the visualization of the data generated were two separate software system and had to do separately. Their work is to provide an interactive tool to visualize the model as soon as it is generated. Moreover, the system developed by Nations et al. provides user the capability to change the parameters and visualize the results interactively.

Rona et al. [21] has used acoustic imaging and visualization methods to visualize and analyze thermal plumes discharging from hot springs on the seabed. Their focus is to compare the data from observations and the predictions from the plume theory. Their system and ours are both motivated by visualizing the plumes discharge. However, their data are the thermal plumes discharging from submarine hot springs while we try to visualize the effluents discharge from the ocean outfalls.

The visualization system of water quality data developed by Forgang et al. [7] is similar to our system in several ways. Both systems are visualizing environmental data and are used in the analysis of water pollution problems. However, the usage of their system is to compare simulated and measured water quality data, while our main focus is to assist the engineers to design a sewage discharge system.

Chapter 3

System Overview

The purpose of the computer visualization system is to provide a tool for engineers to visualize the effect of having multiple buoyant jets discharged from ocean outfalls. An ocean outfall is shown in figure 3.1.

Figure 3.1: Ocean outfall with vertical diffusers: The volcano-like structures are the sewage outfall diffusers for discharging effluents.

The basic requirement of developing such a system is to show the output data

of the mathematical model, JETLAG, in three-dimensional graphics. Having this system developed can help the engineers to design the sewage discharge system which consists of multiple buoyant jets discharging. This system has to produce three-dimensional visualization for the user to investigate the properties of jets with direct and intuitive visual aids. Moreover, the interaction among multiple jets has to be visualized in the system. This is essential to the study of outfall design and environmental impact assessment.

In the following sections, we will discuss the two main components of the system: JETLAG computational model and computer visualization system.

3.1 JETLAG Model

In the study of outfall design and environmental impact assessment, it is desirable to take into account the effect of an ambient current on the initial mixing of buoyant wastewater discharges. The prediction of the concentration of a pollutant or passive scalar along the unknown jet trajectory of a buoyant effluent discharge is a complicated fluid mechanics problem which is not fully resolved. JETLAG is a Lagrangian model which has given reasonably good predictions for this problem.

JETLAG is a mathematical model for buoyant jets in currents, based on the Lagrangian model. It allows the user to find the trajectory of fluid coming out from a submarine jet, under the effect of ambient water currents. JETLAG is usually applied to the field of sewage discharge in order to predict the flow of pollutants. However, JETLAG model does not produce the result for the interaction among multiple jets.

JETLAG model uses the initial jet orientation, the jet discharge parameters, the ambient tidal data and the density profile of ambient water as input parameters. The model will gives a three-dimensional jet trajectory as output. The trajectory is represented by a circular disk structure. In figure 3.2, the disk-like structure is shown and one disk is shown in every five steps in the movement of the effluents. Each disk represents one step in the calculation. In every step, the concentration of the effluents is assumed to be constant and this assumption is used in finding the concentration in the overlapping region in section 6.

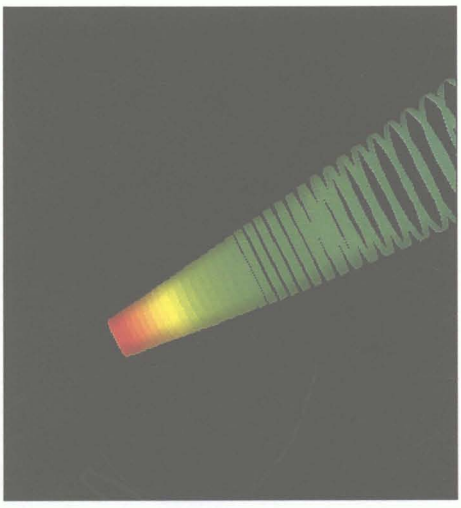

Figure 3.2: An illustration of the output data of JETLAG. In this example, one disk is rendered for every five steps in the movement of the effluents.

A jet is a continuous and steady discharge with an initial momentum. An example of jet is shown in figure 3.3.

Figure 3.3: An example of jet.

3.2 Visualization System

For the graphics part of the system, a few design issues have to be addressed. The design issues include:

- The system should be able to visualize flow data with a sense of high realism. Ambient factors and objects in the surrounding environment should be displayed to provide a proper context for the user to visualize the flow data. This allows the users to have a better sense of realism and the relationship between virtual environment and reality.

- The users can interactively control the viewpoints and take a close-up picture. This allows the users to navigate in the virtual environment.

- The system should be able to allow the users to make frequent changes to the design parameters and the effect should be shown interactively.

- The evolution of jets can be visualized using animation. Animation can be used to display the evolution of jets and to show the effect of the interaction among multiple jets. This can provide the users with better understanding of the flow data shown.

Chapter 4

Detailed System Features

In this chapter, the features supported in the system will be discussed in details. Also, the reasons of having these features and how these features can be used to help the users will be presented.

4.1 Input Interface

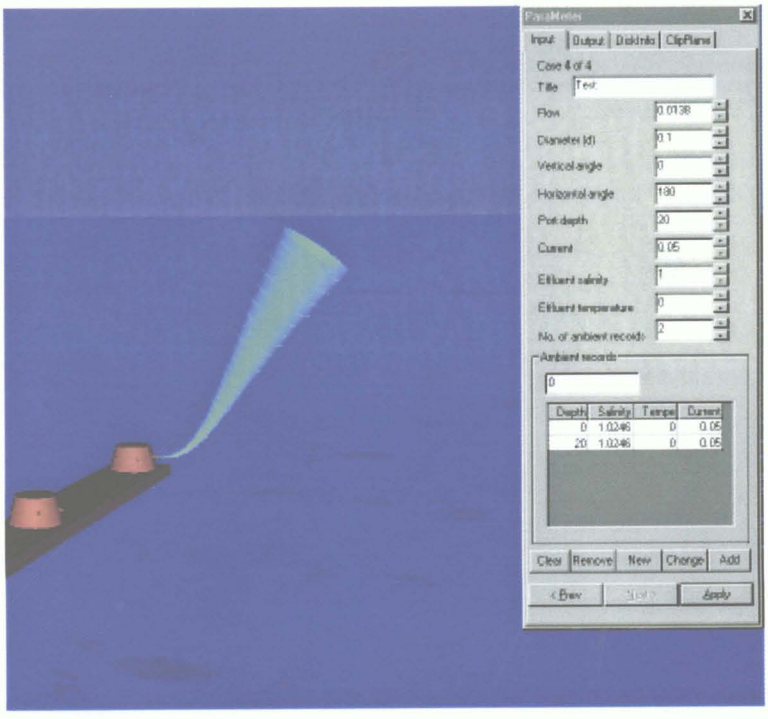

Figure 4.1: Input interface for changing design parameters.

The input interface, as shown in figure 4.1, is used to let the user input the

design parameters. The design parameters include the initial jet orientation, the jet discharge parameters, such as the initial velocity and initial effluent concentration, the ambient tidal current data and the density profile of the ambient water.

4.2 Visualization

In this section, we will discuss the features provided in the system to visualize the flow data.

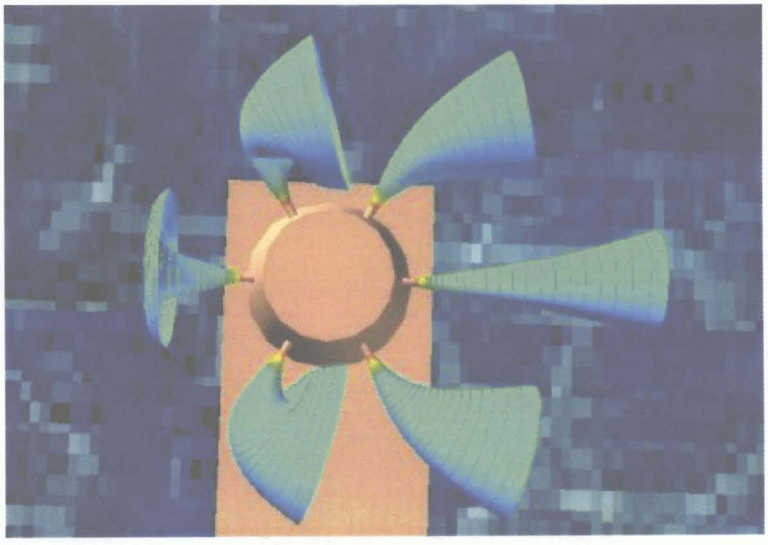

Figure 4.2: Rosette-shaped buoyant jets (overhead view): The effluents are being discharged in six jets.

4.2.1 3D display

The spatial structure of the jets is displayed using 3D color graphics. The users can move the virtual viewpoints to get a different view of the jets. The new view will be displayed instantly to give the user real-time visual feedback when the viewpoint is adjusted. The system allows the users to take a close-up view to look at the details of the jets. Examples are shown in figures 4.2 and 4.3.

4.2.2 Representing the flow data

Fluid flows visualization is one of the common areas in scientific visualization. However, there is no obvious solution developed for representing flows. Many

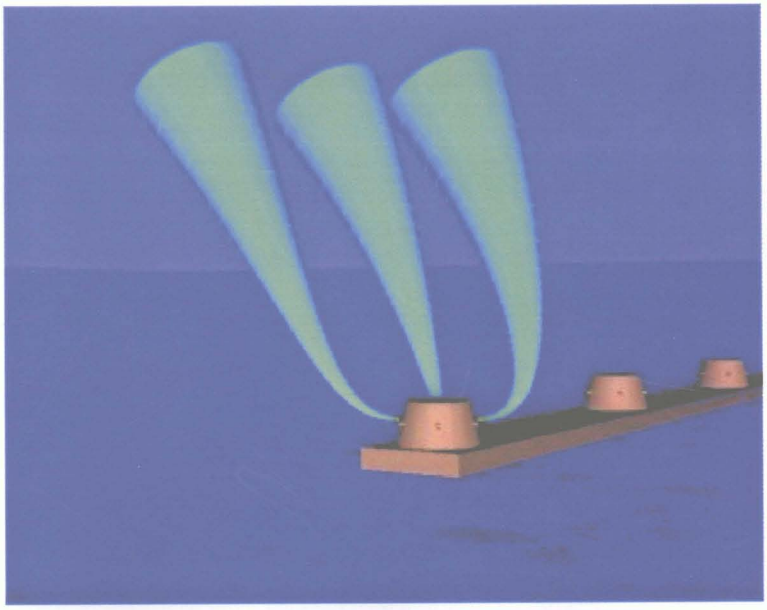

Figure 4.3: Rosette-shaped buoyant jets (perspective view).

Figure 4.4: A snapshot from the system. This shows the jets are confined within a certain layer in the water, which is the result of different input parameters.

different approaches have been developed. Common approaches in presenting

data of a fluid or flow in scientific visualization include the use of particles [20], streamlines [14, 15] or surfaces [11]. A study by van Wijk [28] shows that each of them has its advantages and shortcomings.

Particles techniques have been used widely in visualizing flows. A particle is inserted into the flow and the motion of the particle shows the flow. This is the simplest way to do the job. Particle system has been used in animation, such as fireworks.

Another important technique in visualizing vector field is the use of streamlines. A streamline is the path of a particle in the flow. Streamlines have been used in several applications, such as turbulence. One form of streamlines, vector arrows, is used in depicting the flow in the studies by Kenwright [14]. UFAT [15] uses so-called streaklines, which are similar to the streamlines for visualizing time-dependent flow fields.

Stream surface has also been used in displaying flows. Instead of tracing particles in the case of streamlines, stream surfaces are the results of tracing lines or curves. Hultquist [11] has developed an algorithm to construct stream surfaces.

Brill et al. [1] introduced a technique called streamball for visualizing flows. The technique allows visualizing the flows with convergence or divergence.

Another approach in representing the flow data is to display the data in volumes with textures mapped onto the volume to represent the vector fields. Forssell [8] used the technique called Line Integral Convolution (LIC) for showing the vector fields. The use of LIC can provide a sense that the vector fields are in motion.

In our system, the results produced by the JETLAG are shown in the form of surface models. We choose to use surface models because surfaces can be rendered easily and having a good quality even with limited resources as in the case in most personal computers. The contribution of our work is to provide a tool for changing the design parameters of ocean outfalls and visualizing the effect of having multiple buoyant jets interactively. Moreover, our main concern is to show the volume confined by the boundary and the interaction between

these volumes. The ease of rendering and handling of surfaces proves to be an advantage over the other means of presentation of data.

4.2.3 Animation

Animating the evolution of jets can enhance the understanding of the data displayed. This provides the user with a sense how the fluid discharges from the ocean outfalls and how multiple jets interact with each other. Some of the frames showing the evolution of a jet is shown in figure 4.5.

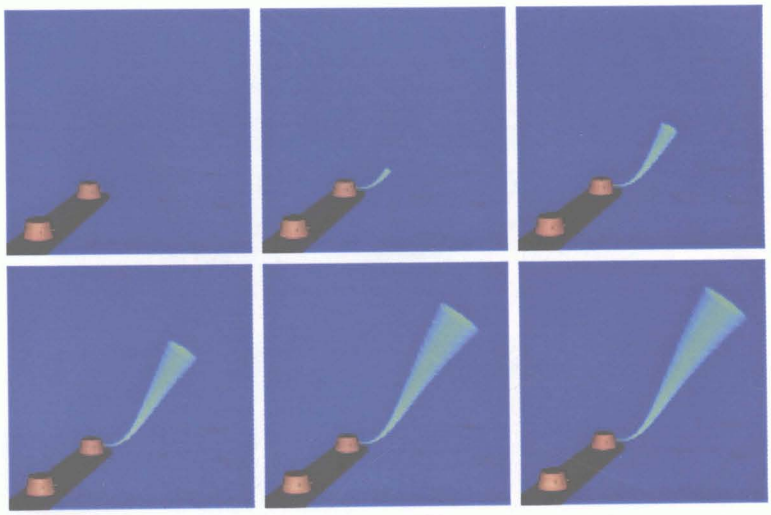

Figure 4.5: Animation of a jet (left to right, top to bottom).

4.2.4 Realism of ambience

In order to provide a high sense of realism, we display some reference objects. This can help the users to understand the orientation in the virtual environment better. Also, we display some ambient factors, such as the direction of ambient tidal current, to provide a proper context for the data to be visualized. Displaying the ambient factors can help in understanding the relationship between the ambient factors and the evolution of the jets. In figure 4.6, the risers mounted on the outfall are shown while the direction of the ambient current is shown as an arrow. Moreover, images (figure 4.7) are texture-mapped to provide a sense of being in the underwater environment.

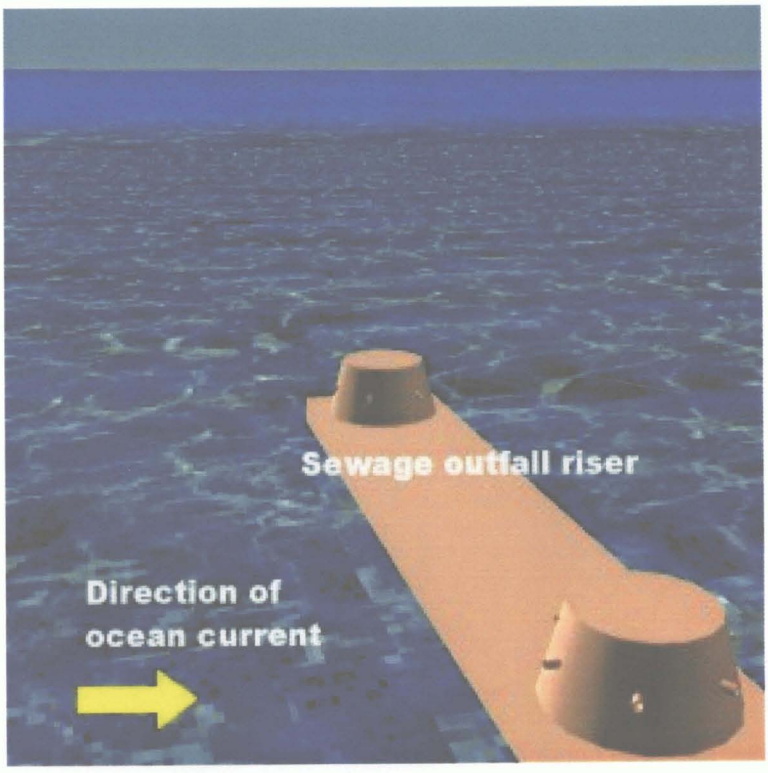

Figure 4.6: Sewage outfall riser.

4.2.5 Color coding

Color is assigned to different points of jets according to the concentration of the effluents. This feature depicts the data visually to the users. In the current implementation, a scale similar to the visible spectrum is used, with lower concentration is shown with color closer to blue and color closer to red is used to represent higher concentration. The changes of the effluents concentrations at most points are not significant and so, the color of the jets in most figures shown appears blue. In figure 4.8, the beginning part of a jet is shown in which there are colors other than blue.

4.2.6 Transparency

When more than one jet are displayed, they tend to appear in a cluster and some jets cannot be viewed clearly from certain viewpoints. The mutual occlusion problem can be eased if some of the jets are rendered in a semi-transparent manner. Figure 4.9 shows the result when one of the jets are highlighted by displaying it in colors while the others are rendered in semi-transparent manner.

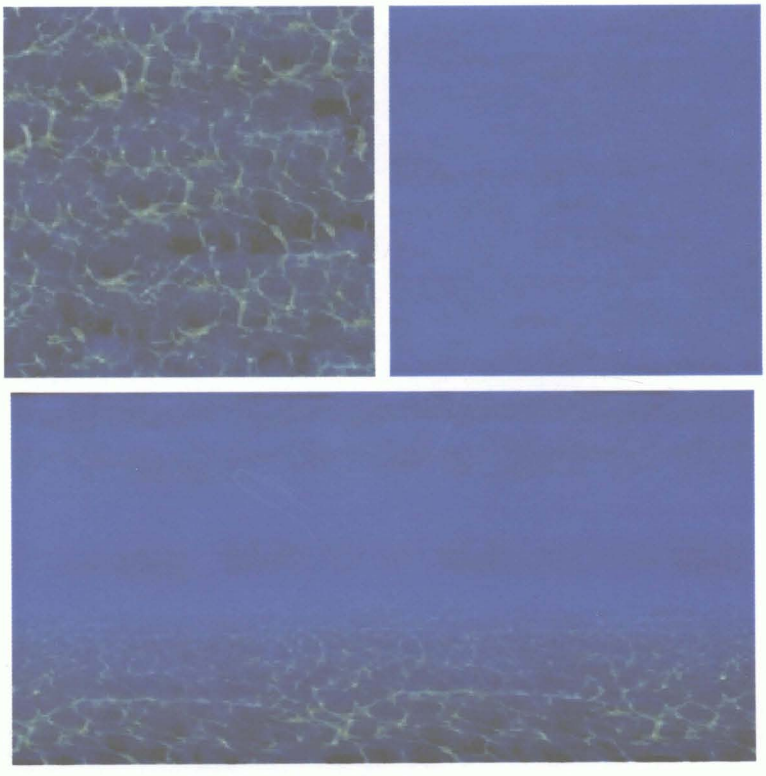

Figure 4.7: Textures used in the virtual environment for providing a sense of realism: (top left) Texture for seabed, (top right) Texture for water surface, (bottom) Texture for blending the seabed and the surface.

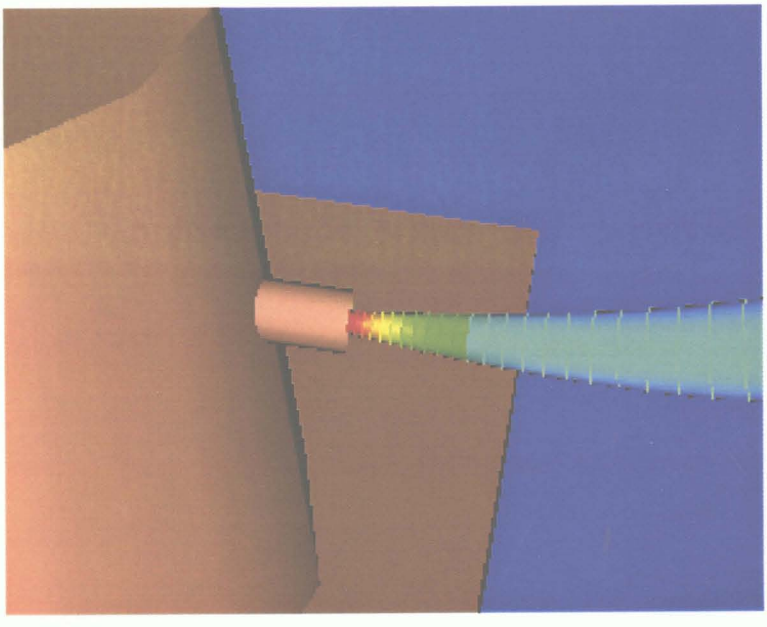

Figure 4.8: The beginning part of a jet.

However, to render the jets with transparency correctly is not simple. This requires a correct rendering order and this order is not easy to be found. An in-depth discussion on transparency rendering will be presented in section 5.

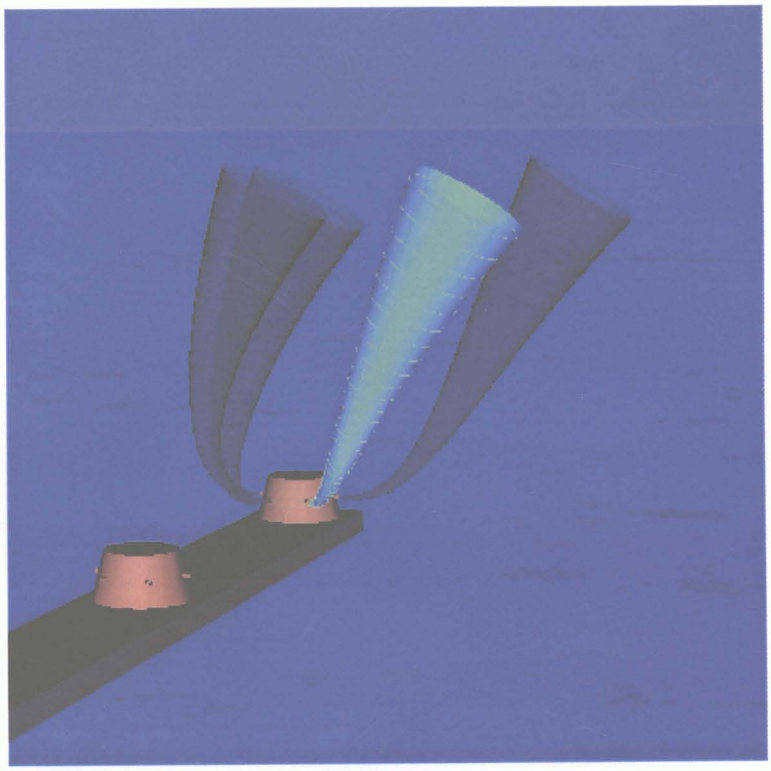

Figure 4.9: A jet is displayed in colors while the others are made semi-transparent. This can highlight a particular jet.

4.2.7 Color animation

In order to provide the users with a sense that the fluid is flowing in the virtual environment, color animation is employed to show the effect. Color bands on the surface models are moving in the direction of jet evolution. We have used grey bands in our system. Originally, the colors on the jets are from the color coding scheme. When color animation is enabled, the original color is replaced by grey bands momentarily. In next frame, the color is reverted to the color-coded one and the grey bands are advanced to the subsequent part of the jet. The process is illustrated in figure 4.10.

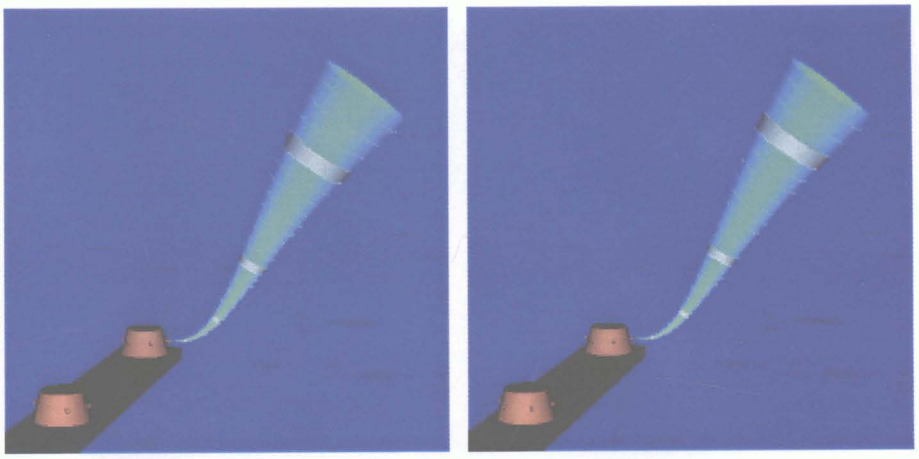

Figure 4.10: In first frame, original color is replaced by grey bands(left). In next frame, the bands are advanced (right).

4.3 Data Analysis

4.3.1 Data interrogation

This feature allows the user to interactively retrieve the data values defined at a point. The user can locate the point of interest by clicking the mouse and the data values, such as velocity and the effluent concentration at the point, will be displayed in the dialog box. An example is shown in figure 4.12.

Moreover, the user can select a particular jet and retrieve the data values or design parameters of the jet by using a mouse. Design parameters associated with the selected jet, such as the initial velocity, the initial effluent concentration and diameter of the outlet, will be shown in the dialog as in figure 4.11.

4.3.2 Jet inspection by intersection

Horizontal planes at different altitudes are used to intersect one of the jets or all of them. By observing the resulting section, the user can have better understanding in how each jet evolves and more important, the interaction among multiple jets. This is helpful in understanding the effect of mixing the jets in different water depths.

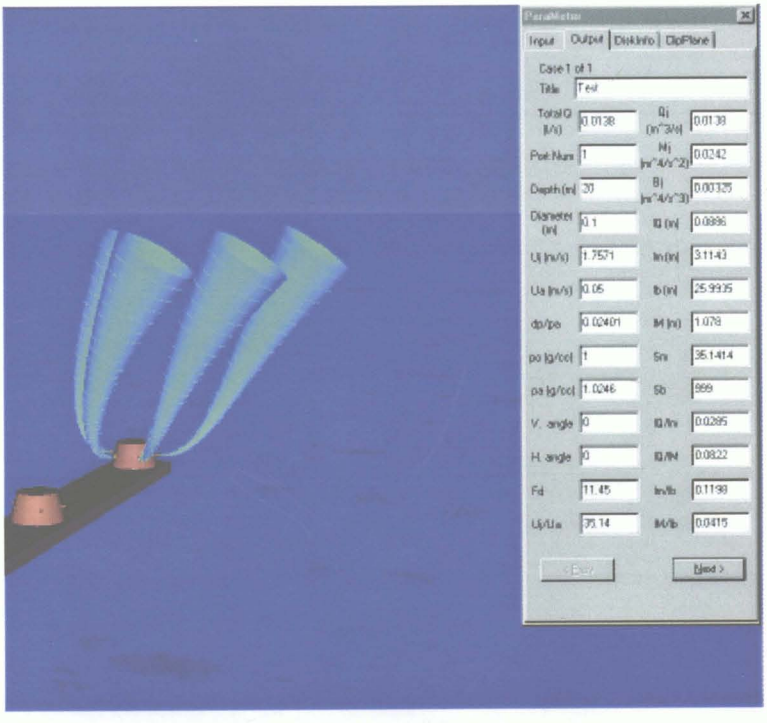

Figure 4.11: When the user selects one of the jets, the parameters about that jet will be displayed in the dialog.

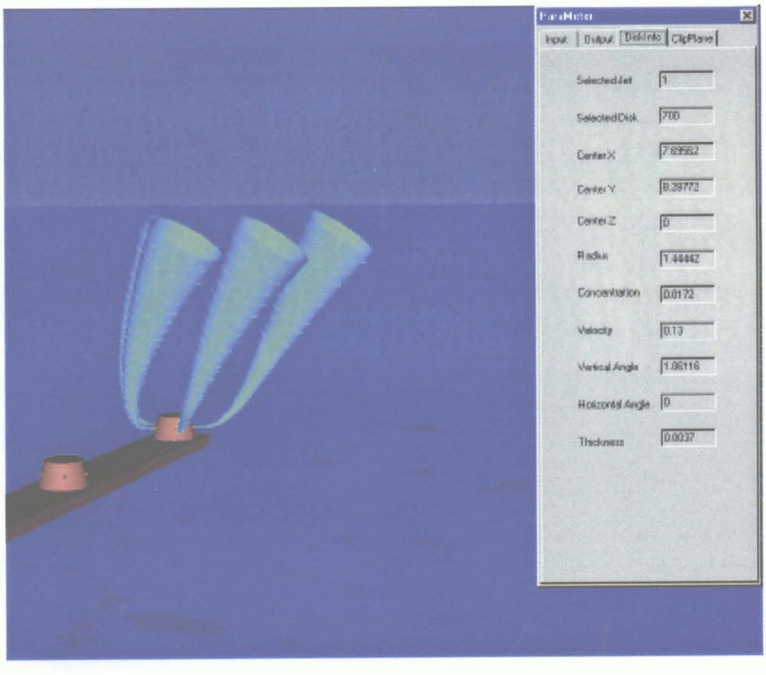

Figure 4.12: The data values defined at a point will be displayed when the user clicks on the point of interest.

4.3.3 Computing overlap among multiple jets

The engineers are interested in the situation when 2 or more jets may overlap with each other. This may signal that the concentration of pollutants in the overlapping region exceeds some threshold or reaches a dangerous level. This feature computes and displays the area of and the concentration in an overlapping region. The user can specify a cutting plane and the information of the intersections of jets at that particular plane will be shown, as well as the distribution of the concentrations of the effluents. An example is illustrated in figure 4.13.

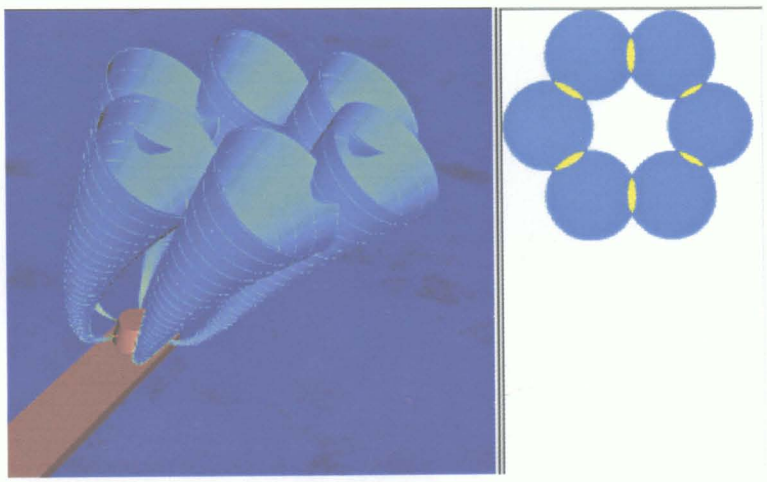

Figure 4.13: Results of computation of overlapping among multiple jets.

To obtain global information about the evolution of jets and how the jets interact, the user can use a series of parallel probing planes. This helps in revealing the distribution of the pollutant concentrations over time and at different places. The details of the computation will be discussed in chapter 6.

Chapter 5

Transparency Rendering

5.1 Transparency

Transparency is an important feature in rendering surfaces. Specular and diffuse reflections can be present in surface rendering, but the feature which allows light to pass through cannot be ignored. We should be able to see through transparent or semi-transparent objects.

There are two main approaches in modeling transparency. The simpler one is called nonrefractive transparency, which ignores refraction in rendering transparency. This approach is simple and easy to implement but this is not realistic enough. The other approach is refractive transparency. On one hand, the result generated is more realistic. On the other hand, it is much more difficult to model than in nonrefractive counterparts. After considering the trade-off between efficiency and realistism, we decided that the emphasis will be put on nonrefractive transparency and when the word transparency is used, it means nonrefractive transparency hereafter.

5.2 Background

When multiple jets are displayed, their trajectories tend to appear in a cluster and may occlude each other. This situation is illustrated in figure 5.1. This can be avoided by highlighting one of them and making others visually less conspicuous. One approach is to display all or most jets in a semi-transparent fashion.

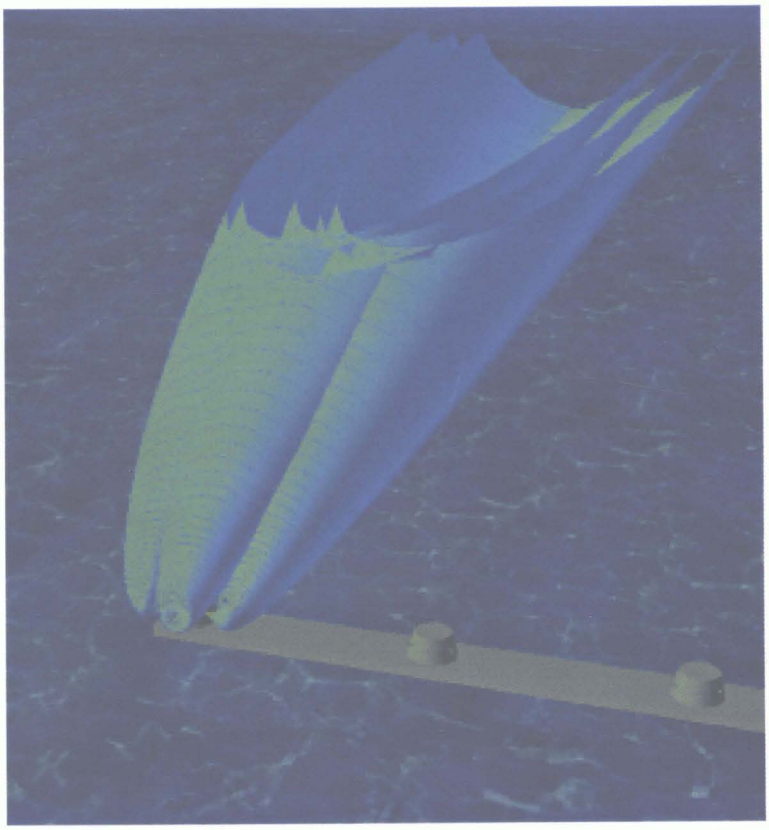

Figure 5.1: In some cases, when multiple jets are displayed, their trajectories may appear in a cluster and occlude each other.

In our system, the jets may be rendered in an arbitrary order. However, standard z-buffer technique in computer graphics area may not be able to render the transparent jets properly and we have to find a new solution to display them correctly.

5.3 Problem Statement

Our goal is to find a method for rendering transparency correctly and efficiently. In our system, the data are visualized in the form of polygons and surfaces. Thus, in searching for a method to tackle this problem, we have a restriction that it must be applicable to polygons and surfaces.

5.4 Previous Work

There are two main approaches in rendering transparency. One of them is to employ the techniques in visible surface determination [6]. The other approach is to use the results in solving the problem of aliasing. In particular, two data structures, octree and binary space partitioning tree, used in visible surface determination will also be presented.

5.4.1 Using visible surface determination

Visible surface determination is one of the fundamental tasks in computer graphics. The goal is to determine which lines or surfaces of the objects are visible. One way of determination is to find a visibility order corresponding to a viewpoint such that a correct picture results if the objects are rendered in that order. A visibility order is an order of rendering such that if an object A occludes another object B, the order of rendering will render object B before object A. Visibility ordering can be used in transparency rendering as the color can be correctly blended if the objects are rendered according to the order.

There are many algorithms for solving the problem of visibility determination. In the following, we will discuss some works on this problem.

The use of occlusion graphs [26] resolves the visibility problem based on the pairwise occlusion relation. The occlusion relation defines as follows: an object A occludes another object B means that there exists a ray from the eyepoint intersects A before B. A directed graph, called the occlusion graph, is formed by adding all pairwise occlusion relations from the objects in the scene to the graph. The visibility order can be found correctly using this approach. However, the existence of cycles needs extra care and the cycles have to be treated in a special way.

The problem of cycle can be illustrated as follows: Let A, B and C be planar surfaces and the sequence of rendering based on a pairwise occlusion relation is that C should be rendered before B, B before A and A before C. It should be noticed that there is a cycle in the order. One possible situation of cycle is shown in figure 5.2.

Whenever there exists a cycle, the easiest way to break the cycle is to split

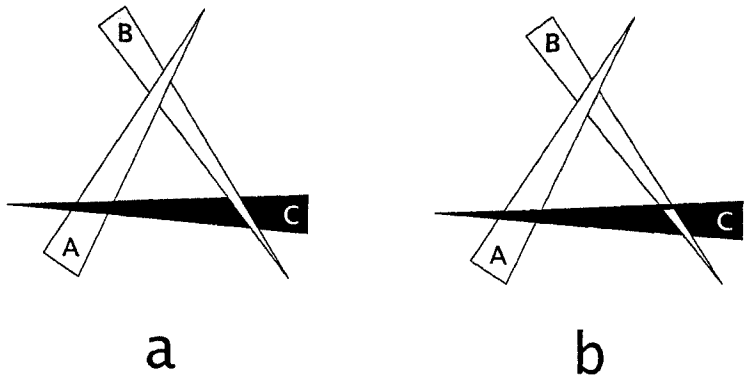

Figure 5.2: (a) A cycle exists. (b) By splitting polygon B, the cycle is broken.

one or some of the conflicting polygons. However, if the number of cycles is large, splitting will cause the problem of increasing the number of polygons significantly. This will degrade the performance of the algorithm.

Another approach uses meshed polyhedra [29, 25] to obtain a total order of cells. The total order is also based on the occlusion relation mentioned above. However, the method does not perform well if the cells are non-convex polyhedra and they are intersecting. Also, the algorithms can only detect the existence of cycles but cannot provide a solution if there is one. The users have to tackle the problem of having cycles themselves.

5.4.2 Using antialiasing algorithms

Some of the algorithms used in transparency rendering are based on the solutions for the problem of aliasing. Aliasing is the result of information loss due to sampling. To solve this, increasing the number of samples per unit area is the only means and having more than one sample for each pixel is called supersampling. However, the drawback of supersampling is the requirement of large memory.

Some algorithms have been proposed to solve the problem. A-buffer [3], a descendent of Z-buffer, is a hidden surface technique which can solve the problems such as antialiasing and transparency. The algorithm can render transparency correctly if there are no interpenetrating semi-transparent surfaces. One drawback is that the algorithm requires the visible transparent polygons to be sorted in Z for correct rendering of transparency. Also, sorting needs the information of the polygons which cannot be discarded until the rendering process is finished.

Thus, the amount of memory required is enormous. EXACT [24] is one of the modification of A-buffer. Similar to A-buffer, polygons are maintained.

Z^3 [12] is an algorithm which provides antialiasing and order-independent transparency. In most cases of having interpenetrating transparent surfaces, it correctly solves the aliasing problem while this cannot be done correctly in the original A-buffer algorithm.

Kelley et al. [13] proposed another modified version of A-buffer in which the algorithms are based on a hardware architecture which performs front-to-back Z-sorted shading. The algorithm used to render transparency requires sorting and the polygons may be required in multiple passes. It uses a small and fixed amount of memory per pixel. However, when there are many transparent objects, the performance will be low. The algorithm suffers from the problem of multi-pass rendering.

The accumulation buffer [9] can maintain the quality produced by supersampling with less memory, but its multi-pass characteristics degrades the performance. The methods in a modified version of A-buffer [30] also requires multiple passes of the objects.

5.4.3 Octree

Octree [6, 22] is a spatial partitioning representation. The idea of the octree is the divide-and-conquer approach. A octree is built by subdividing the 3D scene in all three dimensions into 8 octants and we can label them from 0 to 7. Let the cell at lower-left corner be 0 and that at the upper-right corner be 7. An example of space represented using an octree is shown in figure 5.3a.

Using octree in visible surface determination bases on the property of the non-intersecting structure of the octants. Octree is spatially sorted for correctly displaying the 3D scene. The visibility order can be found by traversing the octree. A correct visibility order can be obtained by rendering the farthest octant first, then its three neighbors, then the three neighbors of the closest octant and finally the closest octant. If the octant contains children, the same traversal order will be applied recursively. For example, in figure 5.3b, a possible order of

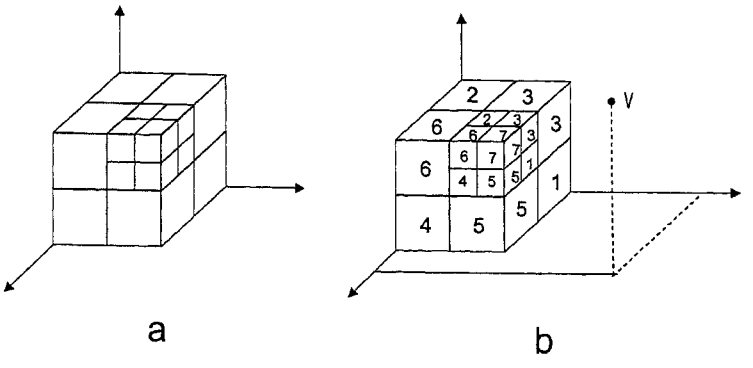

Figure 5.3: Octree: (a) An example of octree. (b) From viewpoint V, a possible order of rendering is 0, 1, 2, 4, 3, 5, 6, 7. (Cell 0 is at the lower-left corner). (Based on [6])

rendering with respect to viewpoint V is to render cell 0 first, followed by cells 1, 2, 4, 3, 5, 6 and finally cell 7 and their children can be rendered in the same order.

5.4.4 Binary space partitioning tree

Binary space partitioning (BSP) tree [6] is a binary tree for partitioning space by polygons. It is a method in finding the visibility order among polygons in 3D space. The idea of BSP tree is that if there is a polygon R and the plane containing R is used partition the polygons into two groups, front and back, relative to the surface normal of R and if the viewpoint is in front of R, the polygons in back group cannot occlude those polygons in front group while polygons in front can occlude those in back.

The algorithm of building a BSP tree is as follows:

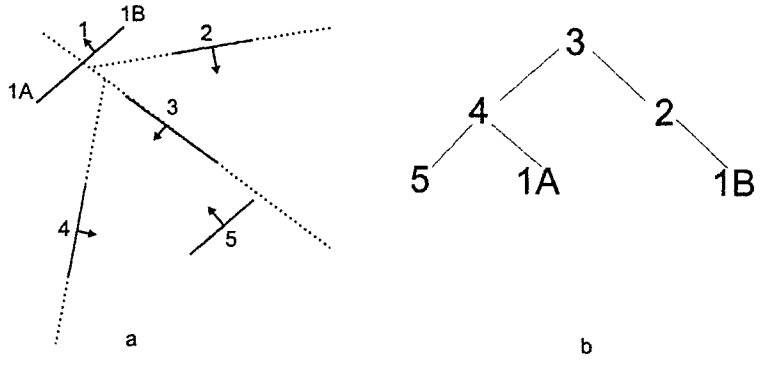

Figure 5.4: BSP tree: (a) 2D view of a scene. (b) Resulting BSP tree with polygon 3 as root.

1. Choose a polygon R as the first partitioning plane and this is the root of the BSP tree.
2. The polygons in the 3D scene will be partitioned into 2 groups, front and back, relative to the surface normal of R.
3. If there is a polygon A which lies on both sides of the plane containing R,
   a. A will be split into 2 parts
   b. Front and back parts will be put into the corresponding group.
4. The front and back groups will be the child nodes of R
5. This process will be recursively performed by the children until every node contains only one polygon.

Once the BSP tree is constructed, the BSP tree can be traversed in linear time for displaying the scene in a correct visibility order, even when the viewpoint is changed.

The algorithm for displaying the polygons in the BSP tree is as follows:

1. If the viewpoint is in front of the root polygon,
   a. Display the polygons in back child.
   b. Display the root polygon.
   c. Display the polygons in front child.
2. Else
   a. Display the polygons in front child.
   b. Display the root polygon.
   c. Display the polygons in back child.

If the viewpoint is in front of the root polygon and if we hope to get a correct visibility order, all the polygons in back child should be rendered before the root polygon and the root polygon should be rendered before those in front child. Similarly, this is true for the viewpoint being in back child. If the viewpoint lies on the root polygon's plane, either way will get a correct result.

5.5 Our Approach

In current implementation, we use a combination of octree and BSP tree, which are usually used separately in tradition. First, we partition the whole space using an octree. The partitioning process will go on until the number of polygons in one cell is smaller than a specified value M. Then, within each leaf in the global octree, a BSP tree is used locally to partition the cell. The idea is illustrated in figure 5.5. In figure 5.5a, there are four cells of the octree. When the number of the polygons in the cell is smaller than M, a BSP tree is built for the cell at top right corner, as shown in figure 5.5b. Figure 5.6 shows the results of rendering transparency using our approach. In the figure, effluents are beging discharged in 2 jets. By varying the value of M, different octrees will be produced.

This tree-building procedure is performed in a pre-processing fashion. During the execution of the system, what we have to do is to find the correct order of rendering based on the data structure if the design parameters are not changed. However, the tree has to be re-built when changes are made to the design parameters.

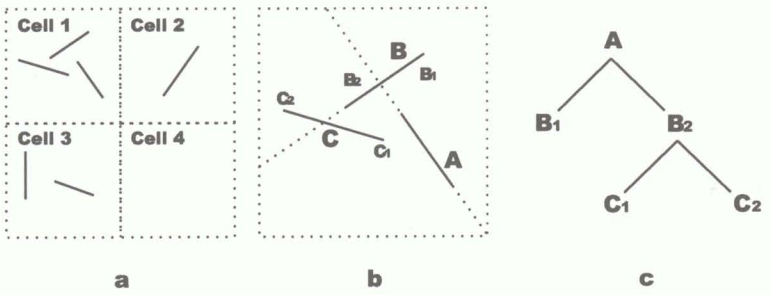

Figure 5.5: (a) Four cells of the octree. (b) Zoom-in view of cell 1 (c) The local BSP tree for cell 1.

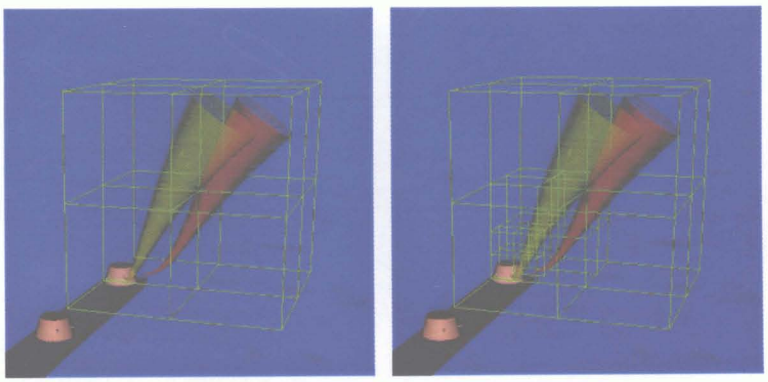

Figure 5.6: Green lines show the octree. Pre-defined maximum number of faces in one octree cell(M): (left) 2048, (right) 1024.

5.6 Results

In order to evaluate the performance of our proposed method, in particular, finding the best value of M, we have conducted a few experiments. The experiments compare the performance of using BSP tree alone and using our hybrid approach with M = 8, 16, 32, 64, 128, 512, 1024 and 2048.

The tables and the graphs for number of polygons and average frame rate are shown in tables 5.1 and 5.2 and figures 5.7 and 5.8 respectively.

For our approach, the number of polygons will become smaller when M decreases up to M = 64 or 128. If M is further decreased, the number of polygons will increase again. From the result for number of polygons rendered, most cases attain the minimum when M is 64 or 128. For the frame rates, we can observe that the higher the frame rate, the smaller is the number of polygons. Also, in

Number of Polygons									
Original (O)	Using BSP Tree only	Combining octree and BSP Tree (M)							
		8	16	32	64	128	256	512	1024	2048
1472	18803	11743	8460	6566	5587	6719	8121	10184	13133	18803
2976	36111	24499	16179	12919	11763	10893	12306	18459	23198	28466
4480	43390	41285	26741	20948	18722	17837	20859	30737	33037	33832
5984	70842	46269	31263	25246	22658	22992	24943	34195	37879	50952
7488	82967	59744	40257	32101	28649	30155	33046	43412	46470	59244

Table 5.1: Experiment results: The number of polygons rendered using different approaches.

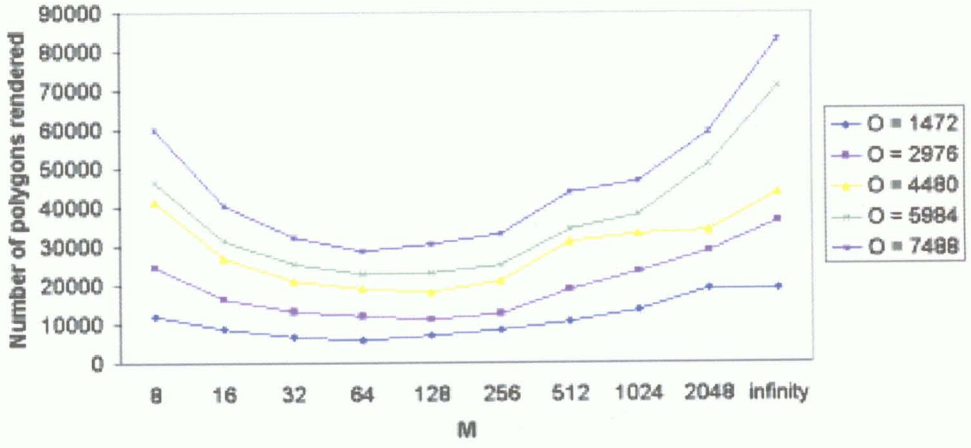

Figure 5.7: Number of polygons rendered against M.

Average Frame Rates (frame per second)									
Original number of polyons (O)	Using BSP Tree only	Combining octree and BSP Tree (M)							
		8	16	32	64	128	256	512	1024	2048
1472	3.05	4.43	6.11	8.59	9.87	8.59	6.76	5.58	4.43	3.05
2976	1.6	2.17	3.37	4.27	4.75	5.14	4.58	3.13	2.46	2.03
4480	1.33	1.27	2.04	2.62	2.99	3.13	2.73	1.86	1.73	1.71
5984	0.81	1.13	1.73	2.21	2.51	2.42	2.29	1.66	1.52	1.12
7488	0.7	0.74	1.36	1.76	1.97	1.91	1.73	1.33	1.24	0.97

Table 5.2: Experiment results: Average frame rates using different approaches.

all cases, the numbers of polygons rendered using BSP tree alone are highest and the frame rates are the lowest.

Results show that the use of BSP tree alone will significantly increase the number of polygons because polygons will be split when building the BSP tree. The number of polygons in the resulting tree is approximately ten times the

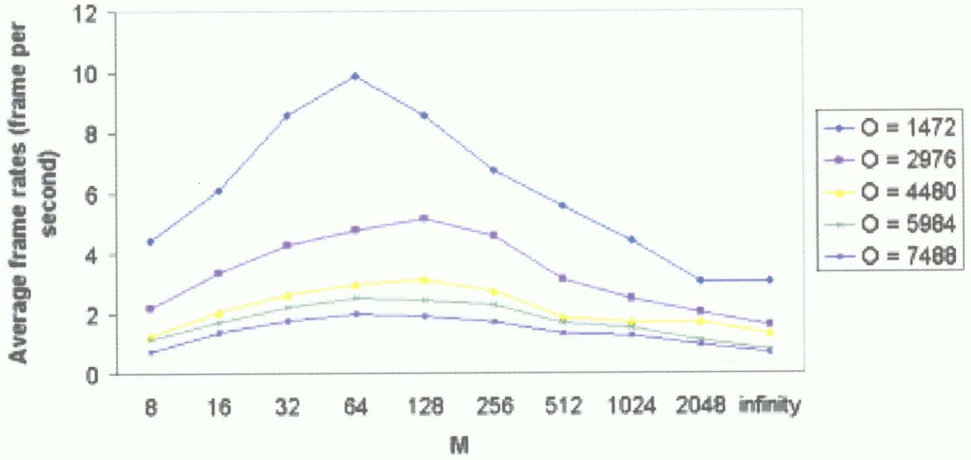

Figure 5.8: Average frame rate against M.

original. From the results of the proposed approach, using a global octree and local BSP tree in an octree cell, fewer polygons are obtained. When compared to using BSP only, the reason for having fewer polygons in the proposed approach is that in a localized BSP tree, fewer polygons are used to build the BSP tree and the number of splitting can be smaller.

5.7 Discussions

Either octree or BSP tree has the advantage of providing a correct visibility order. Using a combination of octree and BSP tree can get benefit from their property. In our algorithm, there is no need to handle the problem of having cycles because BSP tree breaks the cycle by splitting the conflicting polygons (Figure 5.2). Also, this allows intersecting polygons to be rendered correctly. This hybrid approach can be used for rendering scenes with transparency.

Another advantage of our algorithm over algorithms using BSP tree solely is that the number of splitting can be made smaller. In our approach, a localized BSP tree is used. Fewer polygons are used to build the BSP tree. As a result, the number of polygon splits and the number of polygons can be smaller.

However, in this approach, both octree and BSP tree entail polygon splitting, and this normally increases the number of polygons. The major drawbacks are the high requirement of memory. For the octree algorithm, the polygons or the

surfaces may not be aligned properly to the cell boundaries in the octree. This requires splitting the misaligned polygons.

For the BSP tree algorithm, when jets intersect with each other, the underlying polygons will also intersect. In this situation, before we can find a correct order for rendering, we must split the polygons. The situation will become much more complicated when there are a lot of intersections among the jets. The requirement of memory is high because the number of cells in the resulting octree and the BSP tree will increase significantly.

From the experimental results, we found out that most splittings come from building of the BSP tree. Using an octree with smaller M before building the BSP tree may produce less polygons. To further reduce the total number of polygons, we can limit the splitting process when building BSP tree if the area of the polygons is too small. However, this may produce tiny holes on the surface.

When M becomes larger, the splitting of polygons are mostly happened when building the local BSP tree. The octree may have only one or two levels in these cases and many polygons are used to build the BSP tree. On the other hand, when the number of polygons allowed in one octree cell is small, a significant number of polygon splits occurs in building the octree. If the polygons are closely packed, this will be more significant. Either small M or large M will result in a large number of polygon splits. The best value of M lies in the range between 64 and 128. From the experimental results, we found out that the numbers of polygons are at the minimum when there are no more than 64 polygons in one octree cell in most cases.

One more thing is that if the changes to the design parameters are made too frequently, the tree-building process will need much time.

Chapter 6

Computation of overlapping region

6.1 Background

The interaction among multiple jets is interesting. Two or more jets may overlap with each other and the effluents may mix together. The mixing may cause the concentration of the pollutants to rise above some threshold values or reach a dangerous level. Keeping the concentrations within acceptable level is important and studying the interaction among multiple jets is essential in the environmental impact assessment.

One motivation of developing the VISJET system is to allow the users to view the overlapping region of two or more jets and to find the distribution of the concentrations in the region.

In VISJET, the user specifies a cutting plane to view the mixing of the pollutants and the concentrations at that particular plane. The system will then calculate the concentrations and display onto the screen.

The main problem we have to face is to find and display the interactions at the overlapping region.

6.2 Our Approach

To compute the overlapping region involves finding the intersection of two or more jets which are of irregular volumetric structures. This is one of the well-known problems in computer graphics. As our intention is to use a plane to view how the pollutants mix and the concentrations at that plane, we simplify the computation. Instead of finding the intersecting volume, we reduce the problem to finding the overlap in two-dimensional space.

There are two main steps in our approach to solve this problem. First, we have to find the intersections between the jets and the plane. Then, we need to compute the concentrations of the effluents at each point in the overlapping region.

6.2.1 Finding intersections

The jets are represented by surfaces which are made up of a number of polygons. Finding the intersections is reduced to the solution of finding the intersections of the plane and the underlying polygons. In figure 6.1, a plane P is used to intersect with a jet and the intersection is displayed on P'.

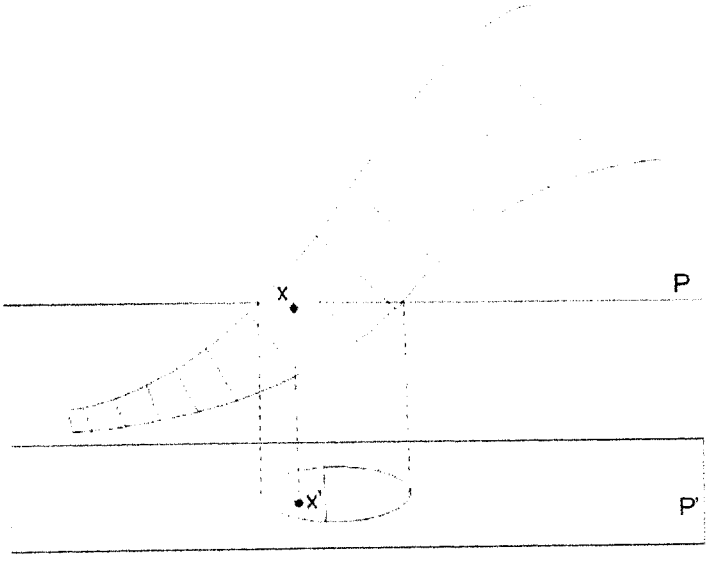

Figure 6.1: A plane P is used to cut the jet and the intersection is displayed on P'. At point X', the concentration is the same as that at X.

6.2.2 Calculating the concentrations of the effluents

The total concentration of effluents at each point may be influenced by more than one jet. We have to find the total concentration at a certain point due to multiple jets. Our idea is similar to the traditional scanline algorithm used in computer graphics area.

Our algorithm for finding the total concentration is as follows:

1. The concentrations at all points in the overlapping region are initialized to zero.
2. For each jet,
   a. Scan the region
   b. Accumulate the concentration at each point due to the jet.
3. After considering all the jets, the total concentrations at all points can be found.

An example is shown in figure 6.1. At point X', the concentration will be the same as the concentration at X.

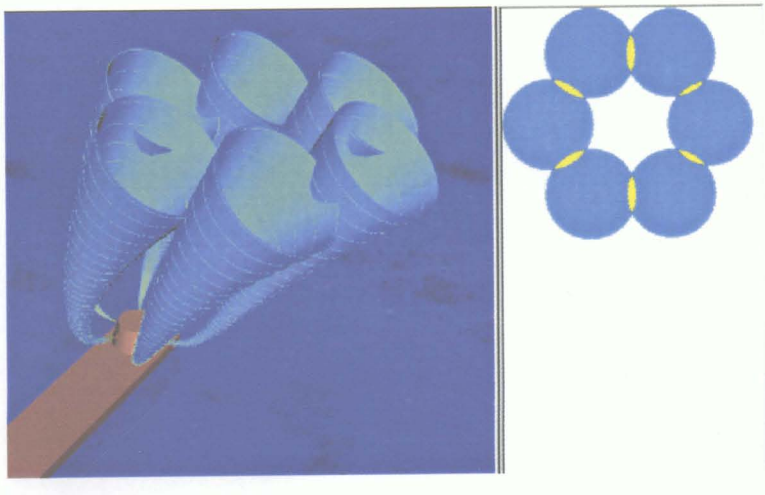

Figure 6.2: The right side shows the intersection of a plane with the jets. Different colors mean different effluent concentrations. In this example, yellow color shows concentration is higher in the overlapping region.

6.3 Discussions

Our intention is to find the total effluents concentration in the overlapping region. One idea is to sum up the values of concentration for individual jet. Our approach is based on this idea by considering one jet at a time and accumulating the concentration at each point. In the current implementation, the distribution of the effluents concentration within one step is assumed to be uniform. This approach may be oversimplified. In real life, the distribution of effluent concentrations may not be uniform and it is more concentrated near the center than the boundary. However, even with non-uniform distribution, our scanline algorithm can still find the correct concentrations at all points.

Chapter 7

Further Research

7.1 Color Coding

Using colors is one of the essential part in computer graphics. MacDonald [17] has given a review on color selection. In a study by Murch [18], the physiological principles of using color were discussed. Color coding is an important and common technique used in visualization. This gives the user a visual sense in describing the underlying data values enciphered in the coloring scheme. From their discussions, no natural color coding scheme exists although color coding in spectral order is common, as in our system. The use of gray scale may be difficult for color-deficient observers to view, according to the study by Murch.

In our system, color is assigned to the jets to represent the concentrations of the effluents at that point. However, the changes of the effluents concentrations at most points are not significant. This makes the color in most part of the jets appear to be the same, blue in our case. A better coloring scheme, such as those with logarithmic distribution, can be considered in the future.

For enhancing visual effects, different coloring schemes can also be employed.

7.2 Fluid Animation

In our system, an important feature is to show how the jets evolve and interact among themselves. Animation is used to show the evolution. The evolution of the jets can be visualized but the effect of movement of the fluids is not easily

depicted. We can enhance the understanding of how the jets evolve using techniques in fluid animation. Color animation can also be used to show the fluid in movement.

Other techniques, such as line integral convolution(LIC) [2, 8], have been developed for showing movement of fluids. Further researches can investigate the possibility on applying these techniques to the VISJET system.

Chapter 8

Conclusion

The focus of this thesis is to show how computer graphics and visualization techniques can be used to help engineers in visualizing ocean outfall data.

We have presented a visualization system of an ocean outfall. This work enables the engineers to view and understand the buoyant jets of a submerged ocean outfall system in a three dimensional way. The features includes the following:

- The system allows the user to visualize the data with a sense of high realism.
- The users can interactively control the viewpoints and navigate in the environment.
- Changes to design parameters are allowed and the results are shown interactively.
- The evolution of jets are animated and visualized.
- Interaction among multiple jets, which cannot be easily obtained by reading the data, is shown.

Also, in this thesis, several design issues in visualization system and the difficulties encountered when designing and implementing such a system have been discussed. In particular, issues on transparency rendering and computation of the overlapping region have been discussed, and the approaches we used to solve these problems have been presented.

Bibliography

[1] Manfred Brill, Hans Hagen, Hans-Christian Rodrian, Wladimir Djatschin, Stanislav V. Klimenko. Streamball Techniques for Flow Visualization. *IEEE Visualization '94*, pp. 225–231, 1994.

[2] Brian Cabral, Leith Leedom. Imaging Vector Fields Using Line Integral Convolution. *Computer Graphics, (SIGGRAPH '93 Proceedings)*, pp. 263–270, 1993.

[3] Loren Carpenter. The A-buffer, an Antialiased Hidden Surface Method. *Computer Graphics, (SIGGRAPH '84 Proceedings)*, pp. 103–108, 1984.

[4] S.K.B. Cheung, D.Y.L. Leung, W. Wang, J.H.W. Lee, V. Cheung. VISJET – A Computer Ocean Outfall Modelling System. *IEEE Computer Graphics International 2000*, pp. 75–80, 2000.

[5] A. Clematis, M. De Martino. Using GIS to Visualise Environmental Information — A Case Study Based on Digital Terrain Models. *IEEE Computer Graphics International '98*, pp. 123–127, 1998.

[6] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes. *Computer graphics: Principles and Practice*, Addison-Wesley, 1996.

[7] Adam B. Forgang, Bernd Hamann. Visualization of Water Quality Data for the Chesapeake Bay. *IEEE Visualization '96*, pp. 417–420, 1996.

[8] Lisa K. Forssell. Visualizing Flow Over Curvilinear Grid Surfaces Using Line Integral Convolution. *IEEE Visualization '94*, pp. 240–247, 1994.

[9] Paul E. Haeberli, Kurt Akeley. The Accumulation Buffer: Hardware Support for High-Quality Rendering. *Computer Graphics, (SIGGRAPH '90 Proceedings)*, pp. 309–318, 1990.

[10] Lambertus Hesselink, Frits H. Post, Jarke J. van Wijk. Research Issues in Vector and Tensor Field Visualization. *IEEE Computer Graphics and Applications*, pp. 76–79, March 1994.

[11] J.P. Hultquist. Constructing Stream Surfaces in Steady 3D Vector Fields. *IEEE Visualization '92*, pp. 171–178, 1992.

[12] Norman P. Jouppi, Chun-Fa Chang. Z^3: An Economical Hardware Technique for High-Quality Antialiasing and Transparency. *1999 Eurographics/SIGGRAPH Workshop on Graphics Hardware*, pp. 85–93, 1999.

[13] Michael Kelley, Kirk Gould, Brent Pease, Stephanie Winner, Alex Yen. Hardware Accelerated Rendering of CSG and Transparency. *Computer Graphics, (SIGGRAPH '94 Proceedings)*, pp. 177–184, 1994.

[14] David N. Kenwright. Automatic Detection of Open and Closed Separation and Attachment Lines. *IEEE Visualization '98*, pp. 151–156, 1998.

[15] David A. Lane. UFAT – A Particle Tracer for Time-Dependent Flow Fields. *IEEE Visualization '94*, pp. 257–264, 1994.

[16] Joseph H.W. Lee, Valiant Cheung. Generalized Lagrangian Model for Buoyant Jets in Current. *ASCE Journal of Environmental Engineering*, pp. 1085–1106, November/December 1990.

[17] Lindsay W. MacDonald. Using Color Effectively in Computer Graphics. *IEEE Computer Graphics and Applications*, pp. 20–35, 1999.

[18] Gerald M. Murch. Physiological Principles for the Effective Use of Color. *IEEE Computer Graphics and Applications*, pp. 49–54, 1984.

[19] Scott Nations, Robert Moorhead, Kelly Gaither, Steve Aukstakalnis, Rhonda Vickery, Warren C. Couvillion Jr., Daniel N. Fox, Peter Flynn, Alan Wallcraft, Patrick Hogan, Ole Martin Smedstad. Interactive Visualization Of Ocean Circulation Models. *IEEE Visualization '96*, pp. 429–432, 1996.

[20] Gregory M. Nielson, Hans Hagen, Heinrich Müller. Scientific visualization: overview, methodologies, and techniques. *IEEE Computer Society*, 1997.

[21] P. Rona, K. Bemis, D. Kenchammana-Hosekote, D. Silver. Acoustic Imaging and Visualization of Plumes Discharging from Black Smoker Vents on the Deep Seafloor. *IEEE Visualization '98*, pp. 475–478, 1998.

[22] Hanan Samet. *Applications of spatial data structures: computer graphics, image processing, and GIS*, Addison-Wesley, 1990.

[23] Andreas Schilling. A New Simple and Efficient Antialiasing with Subpixel Masks. *Computer Graphics, (SIGGRAPH '91)*, pp. 133–141, 1991.

[24] Andreas Schilling, Wolfgang Straßer. EXACT: Algorithm and Hardware Architecture for an Improved A-Buffer. *Computer Graphics, (SIGGRAPH '93)*, pp. 85–91, 1993.

[25] Cláudio T. Silva, Josehph S.B. Mitchell, Peter L. Williams. An Exact Interactive Time Visibility Ordering Algorithm for Polyhedral Cell Complexes. *IEEE Symposium on Volume Visualization '98*, pp. 87–94, 1998.

[26] John Snyder, Jed Lengyel. Visibility Sorting and Compositing without Splitting for Image Layer Decomposition. *Computer Graphics, (SIGGRAPH '98 Proceedings)*, pp. 219–230, 1998.

[27] Lloyd A. Treinish. Severe Rainfall Events in Northwestern Peru: Visualization of Scattered Meterological Data. *IEEE Visualization '94*, pp. 350–354, 1994.

[28] Jarke J. van Wijk. Flow Visualization with Surface Particles. *IEEE Computer Graphics and Applications*, pp. 18–24, July 1993.

[29] Peter L. Williams. Visibility Ordering Meshed Polyhedra. *ACM Transactions on Graphics*, pp. 103–126, April 1992.

[30] Stephanie Winner, Michael Kelley, Kirk Gould, Brent Pease, Bill Rivard, Alex Yen. Hardware Accelerated Rendering of Antialiasing Using A Modified A-buffer Algorithm. *Computer Graphics, (SIGGRAPH '97 Proceedings)*, pp. 307–316, 1997.

[31] Yingcai Xiao, John P. Ziebarth, Chuck Woodbury, Eric Bayer, Bruce Rundell, Jeroen van der Zijp. The Challenges of Visualizing And Modeling Environmental Data. *IEEE Visualization '96*, pp. 413–416, 1996.