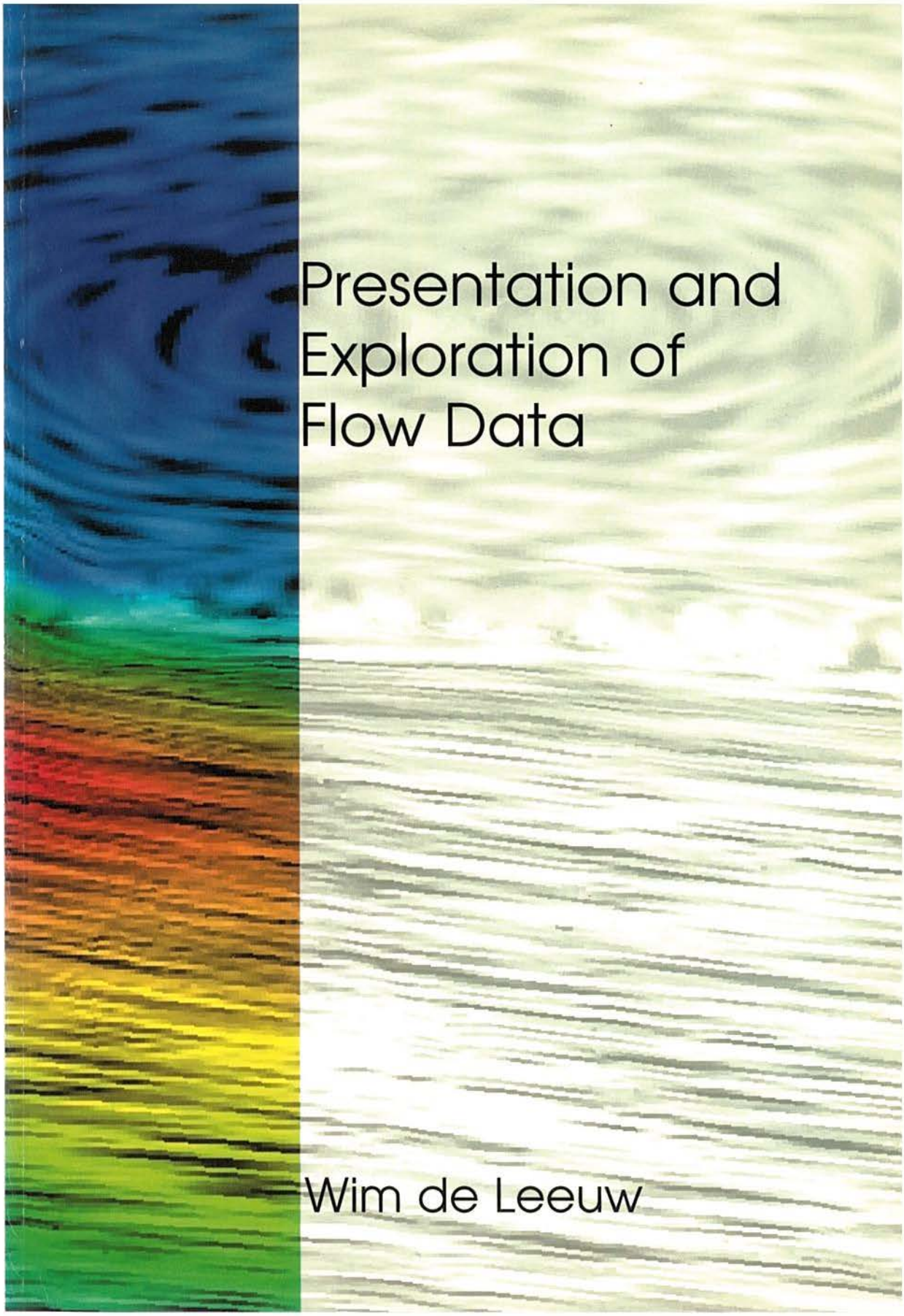


Presentation and Exploration of Flow Data

Wim de Leeuw

Presentation and Exploration of Flow Data

About the cover

Images on the back cover

- Top: the tensor probe (Chapter 3)
- Middle: statistical probe (Chapter 4)
- Bottom: Spot noise emulation of oil-flow image of a fin wedge configuration (Chapter 6)

Cover design: Basia Knobloch

Presentation and Exploration of Flow Data

Presentatie en Exploratie van Stromingsgegevens

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft
op gezag van de Rector Magnificus Prof.dr.ir. J. Blauwendraad
in het openbaar te verdedigen ten overstaan van een commissie,
door het College van Dekanen aangewezen,
op maandag 27 januari 1997 om 13:30 uur
door

Willem Cornelis DE LEEUW

informatica ingenieur
geboren te Leerbroek

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. F.W. Jansen

Toegevoegd promotor:
Ir. F.H. Post

Samenstelling promotiecommissie:
Rector Magnificus, voorzitter
Prof.dr.ir. F.W. Jansen, Technische Universiteit Delft, promotor
Ir. F.H. Post, Technische Universiteit Delft, toegevoegd promotor
Prof.dr. I.T. Young, Technische Universiteit Delft
Prof.dr.ir H.J. Sips, Universiteit van Amsterdam
Prof.dr. A.E.P. Veldman, Rijksuniversiteit Groningen
Dr.ir. J.J. van Wijk, Energieonderzoek Centrum Nederland (ECN)
Dr.-Ing. H.-G. Pagendarm, German Aerospace Research Establishment (DLR)

Preface

The research described in this thesis was carried out at the Computer Graphics Group of the Faculty of Technical Mathematics and Informatics of the Delft University of Technology. Visualization is one of the subjects studied by this group. Previously Andrea Hin [Hin 1994] obtained her doctorate on the visualization of turbulent flow. Theo van Walsum [van Walsum 1995] obtained his doctorate on selective visualization on curvilinear grids. Currently Ari Sadarjoen is working on the extraction and visualization of geometries in fluid flows. Freek Reinders is working on feature extraction and interactive exploration. The project was supported by the Netherlands Computer Science Research Foundation (SION), with financial support of the Netherlands Organization for Scientific Research (NWO).

This work is the result of four years of work on a variety of flow visualization techniques. All the described techniques are somehow related to presentation and exploration. Chapter 2 presents an introduction into flow visualization. The chapters 3, 4 and 5 each cover a different visualization technique and can be read independently.

Many people have contributed to the work reported in this thesis, in different ways. Without their support this thesis would have been impossible to achieve. Some people I would like to thank in particular.

I am very grateful to first supervisor Frits Post. His knowledge of and enthusiasm for visualization were a crucial factor in the success of the research. Working with him was a pleasure. His continuous support of my work was invaluable. I appreciated guidance of and cooperation with Jack van Wijk, my other supervisor, very much. During my stay at the Netherlands Energy Foundation (ECN) he gave my project a very good start. Later he was a continuous source of insightful ideas which form the basis of large parts of the thesis.

An important contribution to this work was made by Hans-Georg Pagendarm of the German Aerospace Establishment (DLR) in Göttingen. He came up with the idea of using spot noise to emulate wall friction images and helped me very much with the development of the idea. Furthermore I thank Martin Rumpf of the Institute for Applied Mathematics in Freiburg[1] for providing the case study of the probe used in a two-stroke engine. Useful contributions were also made by the master's student Remko Vaatstra. He extended the spot noise program to make use of rectilinear grids and to visualize turbulent flow.

[1] Now at IAM, Bonn university

I further thank my current employer the Center for Mathematics and Computer Science (CWI) for the support in the final stage of the project. Especially Jan and Jos of the publishing department did a great job in preparing the material for printing.

My colleagues during the project were Arjan, Andrea, Theo, Winfried, Maurice, Klaas Jan, Erik, Ari en Freek. Their good humour served to create a pleasant working atmosphere. I really enjoyed working with them. The system maintainers Peter, Piter, Aadjan and Kees did their best to keep the computers running as much as possible.

The necessary diversion from working matters was provided by the people from Pechland[2] especially my house mates Frank and Kamil. They provided me with a nice living environment and saved me from working too much.

Finally, I thank Basia for her care and support during the project and the design of the cover of the thesis.

Wim de Leeuw

[2]A part of Rotterdam also known as *de Bospolder*

Contents

Chapter 1

Introduction

Computers have caused a revolution in many research fields. The ability to do calculations several orders of magnitude faster compared to human calculators has enabled researchers to employ large-scale numerical models in addition to the use of physical experiments and analytic methods. Numerical methods can be applied to problems where an analytical solution is impossible. When part of the work of a researcher is taken over by a computer, the problem of in- and output arises. The problem has to be communicated to the computer and the solution has to be communicated back to the user. The output of a numerical simulation usually consists of a very large amount of data. Effective communication of this data is of prime importance to the useful application of computers.

1.1 Scientific visualization

Visualization is an aid to extract information from data. It is a tool to look at data. Just like a microscope reveals things which are too small to be perceived with the unaided eye, visualization offers a view into the world of numerical data. This is achieved by mapping the data to visual parameters, such as shape, colour and texture. Although visualization itself is not new, the use of computer-generated images opens a new world of possibilities for the presentation of data.

Scientific visualization is visualization used for scientific and engineering research. The data usually originates from numerical simulations of physical processes. In this thesis, visualization is used as shorthand for scientific visualization. The aim of scientific visualization is to convey information. Contrary to data, information is subjective. Information for one person may not be information to somebody else.

Simulations often do not produce the desired information directly. The amount of output generated by simulations usually does not permit interpretation by inspection of the numerical data themself. Other ways for interpretation of the results are necessary. Aspects of the data set, such as relations between data values, can be uncovered by translating the data into images. By doing this the perceptual abilities of the human brain are exploited.

An engineer wants to know whether the car which he has designed has a low air resistance, or whether the driver will be injured due to a crash of the car. Finite element analysis can be used to simulate the air flowing around a car or the forces acting on the structural elements of the car. These simulations produce data about the numerical air speed at many locations around the car or the forces and deformation acting on small elements, into which the car has to be divided in order to carry out the simulation. The answer to the questions of the engineer are in the data but cannot easily be found. Visualization can be a tool to present the answer of the questions using the output of the simulation. The calculated deformation of the elements can be presented by showing a rendered image of the shape of the car as the result of the simulated crash.

Usually the user has a question in mind which should be answered by a simulation or measurement. Many different questions might be asked about the same data. It is not known which information can be extracted from a certain data set. Regardless of the question visualization should provide the answer if it can be found in the data. Due to the variety of questions it is not possible to generate a single image which will satisfy all people. Instead, visualization should be an interactive process. The person using visualization to obtain information is an active participant in this process. Apart from presenting data in pictorial form, a visualization system should also offer the user possibilities to actively explore the data. In this thesis several ways to answer user-questions about the data are presented. If the user is interested in a certain detail in a flow field then the local flow probe can be used. If the aggregate properties of the data over a certain area are of interest, exploration of statistical properties is useful. If the user wants to know the global characteristics, spot noise may be used.

1.2 Research objectives

In the visualization process two complementary aspects can be distinguished. First, *presentation* deals with the transformation of the raw data into a picture. Second, *exploration* concerns the interactive inspection and interrogation of data by the user to obtain an impression of their main characteristics. In this thesis both aspects will be investigated in the field of flow visualization. Due to the large amount of work done in the field of fluid dynamics, and because the visualization of the results provides a challenge, visualization of flow fields can serve as an example for other visualization problems. The objective of this research is to investigate different ways to gain insight into the data, by a clear way of presenting the data and possibilities to explore the data.

1.3 Outline of the thesis

This thesis is structured as follows. In Chapter 2 an introduction into visualization will be given, which serves as a basis for the later chapters. In this chapter also the research related to this work is described. Chapters 3, 4 and 5 each explore a specific topic of flow visualization. All these chapters have the same structure: first the technique is described,

then some results are presented and then some conclusions are given. In Chapter 3 an interactive probe is described which presents information on a small region of a flow data set. Chapter 4 discusses the use of statistics to reduce the amount of data presented to the user. The synthesis of texture for the visualization of flow is the subject of Chapter 5 where enhancements to spot noise are described. Chapter 6 presents some cases in which spot noise was used for visualization. Finally, in Chapter 7 conclusions are made and suggestions for future research are done.

Chapter 2

Dimensions in Visualization

An often cited motivation of visualization is given by Richard Hamming: "The purpose of computing is insight, not numbers" [Hamming 1962]. Analysis of large data sets produced by measurements or computer simulations directly from the numerical output is very difficult if not impossible. Visualization serves as a tool to provide this insight in large amounts of data by visual means. Depending on the point of view, different aspects of visualization can be recognized and different possibilities and limitations become apparent. In this chapter we will look at visualization from different viewpoints. This treatment of visualization is by no means complete, we only want to show some aspects of visualization as a background to the chapters that follow.

From a distant perspective, we see visualization in its environment. From this viewpoint visualization is an interface between a user and raw data. Both the user and the data must be understood to create this interface. A brief overview on aspects of users and data important for visualization will be given in section 2.1. If we zoom in we can look at the details of the visualization process itself. In section 2.2 visualization as a process is considered. A number of steps are needed to transform the raw data into images. This sequence of steps is usually called the visualization pipeline. Visualization can also be seen as the problem of transformation of one data representation into another. The numbers are translated into graphical primitives which can be described formally. This is described in section 2.3. Yet another view of visualization is the way it is used. In many applications visualization is used in a cycle together with data generation. Data are visualized, which lead to understanding, which leads to further questions, which lead to new data generation, and so on. In section 2.4 this aspect is explored.

2.1 Interface between user and data

An interface forms the boundary between two agents. Through this interface data is exchanged. Both sides of the interface must be understood to build an interface. In case of visualization we have data from a simulation or measurement at one side and the user at the other side (see Figure 2.1). In this section we will examine the aspects of data and users important for visualization.

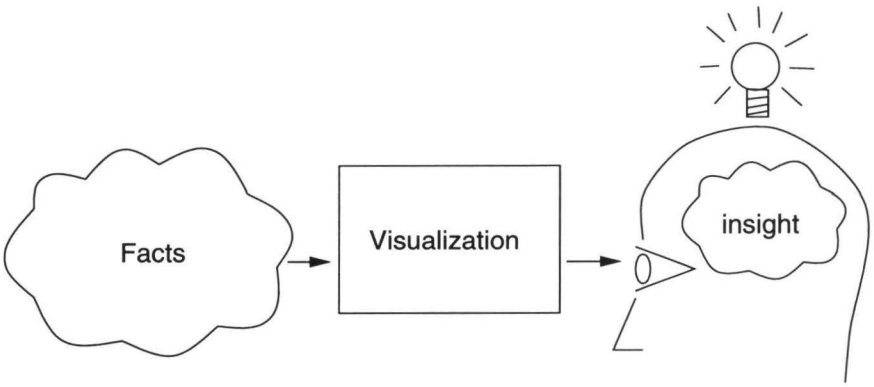


Figure 2.1: Visualization, the interface between a user and the data

2.1.1 Data

In general data are descriptions of facts. Here we restrict the word to mean a *numerical* description of facts. A data set is a collection of numbers together with information about the meaning and structure of the data. Without this information or meta data, the data set is meaningless. Information which must be included are the structure of the data and the quantity represented. Other important information is: the units used or assumptions with regard to the interpolation used in the simulation, the methods used for measuring or computing the data.

Data types

Two important aspects of a data set are the dimensions and data types of the set. For example a temperature is measured at regular time intervals for a certain time. The result is a collection of time samples, each with an associated temperature. The independent variable, or domain, of this data set is time. Temperature is the dependent variable, the range, of the data. Both variables can be represented as real numbers. This data set could be effectively visualized using a two dimensional graph with an x and a y-axis.

In practice data sets often have a much more complex structure. First there may be several dependent variables defined for the same domain, e.g. not only temperature but also pressure may be measured. Data with more than one dependent variable is called multivariate. Also the domain may consist of more than one variable; for example, the temperature may be measured at various elevations as well as at various times. A flow may be described by a three dimensional vector field. In that case the domain consists of the three spatial dimensions. At each point in the domain a vector consisting of three values is defined.

Mathematically, data sets can be characterized by their domain and range dimensions [Inselberg et al. 1994, Schumann, López de Chávez & Graw 1996]. A data set denoted by L_m^n has a domain dimension of m and a range dimension of n. For example a three dimensional vector field is denoted as L_3^3.

Typically, the values in a dataset can be considered as samples of an underlying continuous function. If the sample positions are unrelated with respect to each other it is called scattered data. Many numerical simulations are carried out on a grid. A grid is an imaginary structure which subdivides a part of space into volumetric elements, called cells. Depending on the method used for solving the model, certain constraints have to be satisfied for the grid. At certain points of these cells, often at the vertices, data values are defined. Several types of grids can be distinguished based on the type and regularity of the polyhedra used. Figure 2.2 shows the characteristics of the most common grid types.

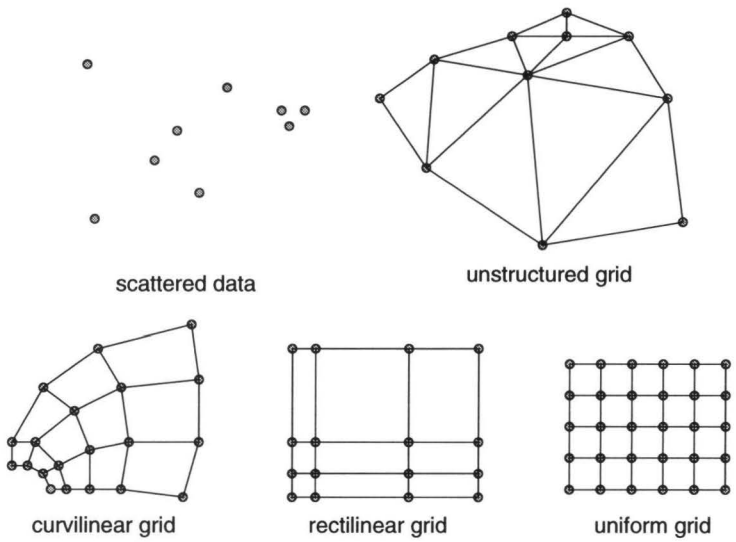


Figure 2.2: Grid types, from top to bottom and from left to right more restrictions are placed on the grid structure

Data interpolation

As we have seen, values in a dataset are only given for a finite number of discrete positions. However, for visualization purposes data values between these positions are needed because many visualization techniques require values at arbitrary positions in the data field. Operations such as stream line integration require that the data is continuous. The value at intermediate positions can be derived by interpolation from the

available samples. There are many methods available to do this. The method to use depends on the structure of the data and the model used to generate the data.

If a data set is represented as a collection of unrelated samples, the problem is referred to as scattered data interpolation [Franke & Nielson 1980, Nielson 1987]. In a one dimensional domain the samples can be sorted and values can be estimated by linear interpolation between the neighbours of the desired function value. In higher dimensional spaces sorting is not possible and other techniques, such as triangulation, of the data must be applied.

In data defined on a grid, interpolation algorithms usually exploit the grid structure to derive intermediate values. The values at the vertices of the cell in which the point to be determined lies are used as input to the interpolation algorithm [Sadarjoen et al. 1996]. Trilinear interpolation is an often used technique in hexahedral cells.

Another approach is the decomposition of the cells into tetrahedra [van Walsum 1995, Kenwright & Lane 1995]. The intermediate value is then calculated using linear interpolation from the four vertices of the tetrahedron in which the point lies.

A complication in interpolation schemes is that many simulations are performed on staggered grids: different variables are defined at different locations with respect to the grid structure. For example in a flow simulation pressure might be given at the centre of the cell while the velocity is given at the face centres of the cell. This implies that for different quantities different interpolation schemes have to be used.

For the remainder of this thesis we will assume that the data is a continuous function over a certain domain. For each position inside the domain of the dataset a value can be obtained, either by interpolation or by using a node value.

Flow data

In this thesis all examples originate from flow simulations. The dominant data type produced by flow simulations is a vector field, representing velocity. Such a field is the solution of a set of partial differential equations, for example the Navier-Stokes equations [Batchelor 1967]. If no simplifying assumptions are made a solution consists of the velocity field and two scalar fields, one representing pressure and the other temperature. In some simulations one or two of these variables are assumed to be constant to simplify calculation. In other simulations additional data is calculated such as turbulence intensity (Section 6.2).

Because analytical solution of the Navier-Stokes equations is impossible in all but the most simple cases, numerical approximation of the solution is needed [Roache 1985]. For numerical solutions the flow domain is discretized into a grid as described above. In the grids the quantities are assumed to vary linearly. The solution of the entire domain is subsequently possible using finite difference, finite element or finite volume methods. Due to the large amount of work done in the field of fluid dynamics and because the visualization of the result provides a challenge, visualization of flow fields has received much attention [Post & van Walsum 1993, Post & van Wijk 1994].

2.1.2 The user

Visualization of data is based on the human ability to visually interpret his environment. To develop effective tools for visualization, consideration must be given to the perceptual abilities, limitations and biases of the human operator. In this framework the computer screen can be considered a kind of incomplete visual environment, lacking for example appropriate primary depth cues. The challenge for developers is to understand what visual information is necessary and sufficient for unambiguous presentation of the relevant data [Kaiser & Profitt 1989]

The visual system consists of the eye and large parts of the brain which are used to analyse the image formed on the retina [Marr 1982]. From the image projected on the retina the brain is capable of reconstructing the three dimensional environment. Visualization can make use of these spatial abilities by presenting data in the form of a projected three dimensional scene. All techniques presented in this thesis use this projection of a three dimensional scene to present the information.

User is a term used for a large variety of people who for various reasons are interested in data. Insight into data might have a very different meaning for different users depending on the goals and questions the user has in mind with regard to the data. Also the background of a user has a profound influence on the understanding and interpretation of presented data. Therefore, if insight is the goal the characteristics and possible questions of the user must be taken into account.

Insight itself is a loose term, usually the user has a more or less specific question in mind which should be answered by a simulation or measurement supplemented by visualization techniques. Although the data might be the same, the questions asked might be very different. Visualization can be defined as the generation of a visual answer to the question of the user that is as clear as possible. If the user is interested in a certain detail in a flow field then the local flow probe described in chapter 3 can be used. If the aggregate properties of the data over a certain area are of interest, exploration of statistical properties is useful. A very specific question is described in section 6.2 where the differences and similarities between a physical experiment and a numerical simulation are of interest.

2.2 The visualization pipeline

If we look at the visualization process itself we can see a number of steps to take in order to present a picture from the raw data. These steps are described by the visualization pipeline [Haber & McNabb 1990]. The pipeline (see Figure 2.3) consists of the steps: data generation, preprocessing, mapping, rendering and display. These steps will be described in more detail in the following sections.

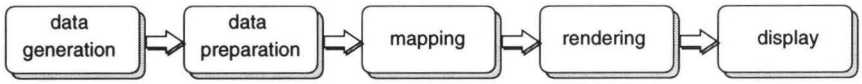


Figure 2.3: The visualization pipeline

Data generation

The origin of data can be measurements or numerical simulations. The generation step is generally not considered part of the visualization process itself. Visualization generally starts with the retrieval of the data set from a storing device such a tape or disc.

Data preparation

Common operations which are applied in this stage are filtering, interpolation, feature extraction, and selection. The goal is to make the data suitable for the visualization technique which the researcher wants to apply. Often this means reduction of the amount of data, for example by resampling the data at a lower resolution. More sophisticated techniques to reduce the amount of data take the physical meaning of the data into account. Content based selection [van Walsum 1993*a*, van Walsum 1995] and topology extraction [Helman & Hesselink 1991] are examples of such techniques.

Mapping

In the mapping step numerical data is translated to graphical primitives, geometry, colour and texture or time. There are many ways to map numerical information to attributes of graphical primitives. Often used techniques are volume rendering, isosurfaces, stream lines and vector plots. The choice of mapping depends on the type of data to be visualized, the available display algorithms and personal preferences of the user. In section 2.3 mapping will be described in more detail.

Rendering

In the rendering stage the geometric primitives are traversed and an image is generated. An image is a two dimensional raster of pixels each of which has a certain colour or intensity value. Often the mapping assumes a three dimensional space which is projected onto a plane, the viewing plane, using a viewpoint and lighting conditions as input. Computer graphics research has focussed on this problem for the last few decades and much progress has been made in this field [Foley et al. 1990]. The calculation of an image from a scene takes a large amount of computation. Workstations equipped with special-purpose hardware allow the rendering of complex scenes in only a fraction of a second. The speed and versatility of this hardware is still increasing rapidly, allowing

application of more computationally expensive visualization techniques in interactive environments.

Display

The rendered image has to be presented to the user by some display device, this is done in the display step. The most common device used for this purpose is a raster display (CRT). If the rendering of animation frames takes too much time to do it in real time, a sequence of images may be generated, stored and played back at the desired frame rate. Other media such as slides and video may be used for presentation purposes. Hard-copy of the visualization can also be achieved by devices such as printers or plotters. Image data needs a large amount of storage and communication capacity. Image compression is essential for storage and communication of images to fully exploit the available resources.

2.3 Classification of mappings

Data sets can be classified by the number of domain and range dimensions (see Section 2.1.1). In the same way a taxonomy of visualization techniques can be made by looking at the dimensions of the domain and range of the data that can be shown by the visual object. The number of different independent (domain) and dependent (range) variables that are communicated by a certain visual object are a basis for classification. Geometric objects have a certain dimension: a point is zero dimensional, a line has one dimension, a plane two, and so on. How these dimensions are divided between domain and range depends on the mapping used. Range data can be mapped to the graphical primitive in several ways, for example by deformation or by colouring of the visual primitive.

The same notation used for data sets can be used to describe visualization techniques:

$$M_d^r$$

where r is the range dimension and d is the domain dimension. Another way to look at this description is the minimum amount of information needed to specify the graphical primitive.

Some examples will be given to clarify the classification. A two dimensional graph has a one dimensional domain and a one dimensional range so it is denoted as M_1^1. L_1^1 data visualized using a graph can be fully recovered from the graph. A three dimensional *stream line* shows the velocity direction along a line. The domain is one dimensional and the range two dimensional; it is M_1^2. Volume rendering can be used to visualize scalar fields defined on a three dimensional domain and therefore is a M_3^1 type mapping. In Table 2.1 some more techniques are shown within this classification.

Note that although the table only lists dimensions up to three the table could be extended in both directions to four and more dimensions. However, there are few techniques which fall outside the table given. A notable exception are icons which show

		range dimension		
		1	2	3
domain dimension	0	number	icons	
	1	graph	3D stream lines	stream tubes
	2	height field	spot noise	-
	3	volume rendering	-	-

Table 2.1: Classification of visualization techniques based on dimensions

multi-variate data. Icons for mapping a zero dimensional domain to a theoretically very large number of range dimensions have been presented (see Subsection 2.3.1).

Time can be used for the visualization of an extra domain dimension. Animations of data can be very useful to show data sets with too many dimensions to show in a single image. This is especially true if the time is a dimension in the data itself and time in the data is mapped to time in the visualization. Other variables can also be mapped to time; for example, when a slice is moved through a data set in an animation, one spatial dimension is mapped to time in the visualization.

Some of the graphical primitives are three dimensional. On the screen these graphical primitives are projected on a two dimensional surface. This projection leads to loss of data. For example, if a three dimensional stream line is projected onto a plane, one range dimension is lost because at each point only one directional component is available. In some cases this loss of information can be prevented. Using cylinders in combination with rendering using lighting can provide information on the three dimensional shape of the visualization object. Also time can be used here by rotating the object on the screen the three dimensional position and shape can be reconstructed by the projections.

This classification of visualization techniques does not capture all possibilities of visualization because only the mapping is taken into account. This is a classification on a low level without taking into account additional meaning based on the phenomena modelled which may be present in the visualization. For example in visualization of the topology of a vector field [Helman & Hesselink 1991] critical points are connected with stream lines. The streamlines show the direction of the vector field. However these particular streamlines together with the critical points also present information on the global structure of the vector field. This particular use of streamlines usually cannot be extracted from the image itself, additional information is needed to know that the topological structure is being visualized.

Some visualization methods are based on the extraction of meaningful entities often referred to as features from the data before the actual mapping is carried out. The selection methods proposed by van Walsum [van Walsum 1995] are an example of this strategy.

One strategy in gaining insight is top-down exploration. First, the user investigates the global structure of a data set, then interesting phenomena at smaller scale are studied. This process can be supported by visualization. Tools must be supplied which present an overview of the data leaving out details but just showing the global structure of the data. If the global structure is understood it is possible to identify interesting locations and investigate such regions. This is a recursive process.

In the remainder of this section more examples will be given, mainly in the light of the dimension of the mapping but also with respect to the previous sections. Each subsection will deal with examples related to work described in later chapters in this thesis. The first one with the visualization of tensors (chapter 3), the second on with data reduction and statistics (chapter 4) and the last one with the use of texture (chapter 5).

2.3.1 Tensor visualization

Several researchers have devised symbolic representations of multivariate data. Chernoff [Chernoff 1973] used an icon resembling a human face in which data was linked to features of the face such as the size of eyes. Because 18 different facial features can be linked independently to data ranges, in our classification, it is denoted as M_0^{18} mapping. This method exploits the human ability to interpret facial expressions. This method can be very useful in case of non-physical data. For abstract data also Kleiner-Hartigan trees [Chambers et al. 1983] can be applied. These can be used for the representation of an arbitrary number of parameters by using a tree-shaped glyph in which the length of an angle between branches provides information on the data.

Although a single glyph is a M_0^N mapping many glyphs representing different points in a data space can be rendered. In this case the mapping can be used for domains with dimension larger than zero. However not all data is shown in this way, only samples at discrete locations. We denote it by an S behind the domain dimension to distinguish sampled visualization from continuous visualization. So Chernoff faces on a 2D domain is denoted by M_{2S}^{18}.

Rappoport et al. [Rappoport, Kosloff & Mayer 1992] mapped data items of wave functions from molecular dynamics simulations to polygons. They used triangles which colour showed the magnitude of the complex value and which size shows the size of the cell in which the value occurs. Many of such polygons distributed regularly over the domain give an overview of the data. A single polygon is M_0^2. A collection of these polygons is M_{3S}^2.

The *Stream Polygon* [Schroeder, Volpe & Lorensen 1991] is used to visualize flow direction and other data related to the local flow. It is a regular n-sided polygon perpendicular to the velocity vector. Data are mapped onto attributes of the polygon, such as the radius and the relative length of the edges. Deformation can be visualized by deforming the polygon accordingly. By sweeping the polygon along a streamline a stream tube results. Scalar values can be mapped on the surface of this tube using colour. Data are shown along a line so it is denoted as M_1^N. N depends on the number of parameters used

to define the cross section of the stream tube, the stream polygon, and whether colour is used to show additional data. Typical values for N range from 3 to 7.

In the visualization of tensors and tensor fields earlier work is done by Haber and Delmarcelle. Haber [Haber 1990] represents a symmetric second order stress tensor by an object consisting of a shaft and a disc. The shaft points in the direction of the major eigenvector of the tensor. The axes of the elliptical disc correspond to the other two eigenvectors. The length of the shaft and the radii of the ellipse are proportional to the magnitudes of the eigenvalues of the tensor. These tensor icons are mappings of the M_0^6 type, which can be used as a M_{3S}^6 type mapping.

Delmarcelle and Hesselink [Delmarcelle & Hesselink 1993] use hyperstreamlines to visualize the global structure of a symmetric tensor field. A hyperstreamline is an ellipse swept along the line everywhere tangent to the major principal direction of the tensor. The magnitude and direction of the other two eigenvectors at each point are reflected in the size of the swept ellipse. The resulting surface can be coloured according to the magnitude of the major eigenvector. If coloured hyperstreamlines show all information of a symmetric tensor along a line, as such they are a M_1^6 type mapping.

The local flow probe described in chapter 3 is another representation for a tensor based on a decomposition in meaningful components. It is a M_0^{12} type mapping used for interactive exploration of a flow field. The local flow probe is intended to provide detailed information on a small region in the flow, similar to a measuring device inserted in the flow in experimental fluid dynamics.

2.3.2 Data reduction

As shown by Table 2.1 the dimensionality of data which can be visualized is limited. In some data sets the dimensions of domain and range exceed the capacity of current visualization techniques. Instead of working on visualization techniques allowing more dimensions to be displayed at once, it is also possible to preprocess the data in a data reduction step reducing the dimensionality of the data. In this way not all original data is displayed in the visualization, however by careful reduction of the data the objective of understanding in relation with a specific question might be retained.

Data reduction may also be necessary if the data set is so rich in information that display of the entire set in a single image is impossible. A theoretical upper limit of the data which can be represented by a single image is the image size (width times the height) in pixels times the colour depth per pixel. A 512×512 image with a colour depth of 24 bits holds 768 KB of information, enough to hold a $64 \times 32 \times 32$ vector data set. If we keep in mind that this is a theoretical upper limit and that current simulation grids are often much larger than the one mentioned, it is clear that data reduction is unavoidable.

An obvious way to limit the dimensionality of the domain of the data is slicing. A two dimensional subdomain from the data is chosen and subsequently visualized. By varying the slice in time a complete overview of the data can be obtained. Slicing is only

effective if the domain dimension is small because in higher dimensions the number of possible slices becomes too large to be effective. Additional tools are possible, such as concurrent display of several orthogonal slices [van Wijk & van Liere 1993]. If the range dimension is too large projection of some of the dimensions can be considered.

Data reduction can also be achieved by representing the data at a lower sampling density. This is the domain of multi-resolution modelling. One approach for the representation of data at different levels is achieved by *Wavelets*. The central idea behind wavelets [Chui 1992, Daubechies 1992] is that a function is decomposed into orthogonal basis functions which are both local and have different supports. In this way the data can be reconstructed at different refinement levels as well as locally.

2.3.3 Textures for visualization

A texture is a pattern on the surface of an object. The perception of differences between material type such as stone, wood or cloth is mainly due to the texturing of the material. The human eye is sensitive to texture characteristics [Gibson 1950]. This sensitivity can be exploited for visualization purposes. Examples of texture synthesis techniques for vector field visualization are *texture splats* and *Line Integral Convolution* (LIC).

Texture splats [Crawfis & Max 1993] are small transparent textures used in volume rendering. Each splat represents the contribution to the opacity function of a node value in the data grid. They can be adapted to show vector fields by a slight disturbance of the reconstruction function used to generate the splat. This disturbance results in anisotropic splats; in this way the impression of a vector direction is created. Although texturing hardware can be used to speed up image generation, this technique should be considered as an iconic technique because individual splats are visible in the final image. The visualization is not continuous but sampled at the location of the splats.

LIC was developed by Cabral and Leedom [Cabral & Leedom 1993]. A LIC texture is generated by convolution of an input image, usually a random noise texture, with a one dimensional filter kernel, aligned with the local streamline direction through the kernel centre. It was extended by Forssell [Forssell & Cohen 1995] to be able to visualize curvilinear grids. Stalling and Hege [Stalling & Hege 1995] proposed a method for the efficient generation of LIC textures. A single LIC texture in its form as proposed by Cabral and Leedom is a M_2^1 type mapping, it shows the direction of a vector field on a surface. Note that direction on a two dimensional surface can be described by one parameter. By using variable length filter kernels or by animation of textures using varying filter kernels also the velocity magnitude is shown. Thus by using one of these extensions it is an M_2^2 type mapping.

Spot noise, described in Chapter 5, is also a technique for texture synthesis which can be used for flow visualization. In its basic form it is a M_2^2 mapping which shows the velocity direction and magnitude on a surface. Spot noise can be used to provide a global overview of flow fields in an intuitively clear way.

2.4 Presentation and exploration

So far we described visualization as the transformation of data into images. The data flow is from the source to the user. The user is seen as a passive receiver of data in pictorial form. However there are many choices to be made with regard to the visualization process. It is the user who is most capable of making these choices. So visualization should not be regarded as a one-way process with data flowing from the source to the user, but as an interactive process, where the user also sends data to the visualization process telling how the data should be processed. This input to the visualization system can be at any stage of the visualization pipeline.

Examples of these choices are the mapping used and the preprocessing applied. A keyword in the interactive analysis of data is exploration. The user interacts with a visualization system to explore a data set. Using the interaction techniques offered by the visualization system the user can actively search for the answers to the questions posed.

2.4.1 Presentation

As described in the previous sections, mapping is a very important aspect of visualization. In this stage the way the data is presented to the user is specified. There are many possibilities to carry out the mapping. Certain aspects of the data play a role in the possible mappings of the data. However with a certain type of data there are still many choices with respect to the mapping.

An important factor in the presentation of data is the background of the user. Computer visualization is often inspired by existing visual representations in a certain area. For example in fluid dynamics research traditionally windtunnels and scale models were used. For the visualization many techniques were developed such as the injection of smoke into the flow. The images which were produced in this way [van Dyke 1982] have served as an inspiration for scientific visualization [Becker, Lane & Max 1995].

Other fields have less tradition with respect to the visualization of data. The reason might be that the data type produced is more abstract and has no direct connection to physical phenomena.

The issue of presentation appears in many places in this thesis. An example is the mapping chosen for the presentation of an abstract quantity such as a velocity gradient tensor in Chapter 3 by visual primitives that are easy to understand. Also the presentation of the wall friction data similar to the results of windtunnel experiments (see Section 6.1) shows the importance of the presentation of the data.

2.4.2 Exploration

The visualization pipeline can be used as a conceptual model for interaction with the visualization process. The different stages in the process are points where the user can interact with the visualization system. Each stage has its own set of controls with which the

user can adjust the visualization process. Examples of systems applying this paradigm of visualization are AVS [Advanced Visual Systems Inc. 1992, Upson et al. 1989] and Iris Explorer [Silicon Graphics Inc. 1992] in which a user can build a network of processing units (modules), roughly equivalent to the steps in the pipeline, to specify a certain way of visualizing a data set.

Specifically for the mapping stage of visualization Ribarsky et al. [Ribarsky et al. 1994] developed a system called Glyphmaker. With this system data types can be interactively linked to attributes of geometrical icons. Position, size and colour are attributes which can be used for mapping of the data.

Interaction with three dimensional data through a CRT-screen, a mouse and a keyboard has its limitations because interaction between the user and the system is indirect. Normal ways of interacting with a three dimensional space, such as moving one's head to get a different viewpoint, have to be achieved in an indirect manner by the input devices. With virtual reality (VR) a less indirect interaction with an environment existing in the computer can be achieved. By presenting the user full viewing angle images and accurate tracking of the movements of the head a *virtual environment* can be created in which the user is immersed. By tracking hand movements the user can manipulate and navigate in the virtual reality as if it was a real world. The virtual windtunnel [Bryson & Levit 1991] is an example of the use of virtual environments for visualization. In the virtual windtunnel a user can move around in the data and release streamlines by movements of the hands.

If the data originates from a computer simulation, visualization can be used to monitor the simulation process. Instead of visualizing the end result of a simulation the simulation itself is visualized while it is running. If the user can interact with the simulation while it is running, it is called *computational steering* [van Wijk & van Liere 1996, Jern & Earnshaw 1995].

In this thesis interactive exploration plays an important role. All the techniques presented are implemented in interactive applications. Using efficient algorithms for the generation and rendering of geometry and texture, the user receives a quick response to his requests. The probes presented in chapter 3 and 4 offer the possibility of interactive placement while the information is updated in real time.

2.5 Conclusions

In this chapter a brief introduction into different aspects of visualization has been given. Several approaches and techniques to achieve insight into large numerical data sets using graphical methods were described. Due to the large variety in users as well as types of data a one and only solution to the visualization problem is not possible. The visualization problem can only be solved by a combination of accurate and varied intuitively clear visualization techniques combined with the possibility of interaction to change and customize the visualization. Visualization is an interactive process in which exploration and presentation are keywords.

The application field we focus on in this thesis is fluid dynamics. The visualization of the results of flow simulations provides a challenge because the three dimensional domain and the three or more dimensional range are too large to be directly visualized by existing techniques. In the remainder of this thesis we will describe three of techniques to present and explore flow data: a local flow probe, statistical probes, and spot noise. In each of these, both presentation and exploration play an important role.

Chapter 3

A local flow probe

In addition to global visualization, where an overview of the data in the whole domain is given, the user often wants to investigate the data in more detail at certain interesting locations. Therefore the local flow probe was developed. The probe is a complement to global visualization techniques such as stream lines. Using the probe, the user can investigate a small region in a flow field in detail. The probe shows the velocity and velocity gradients at a point in the flow.

In physical experiments, a probe is a measuring device to investigate a small region in a field. In this chapter we use it to refer to an interactive visualization method to show many variables in a point. A probe will be described which shows velocity and changes in velocity at a point in the flow. Velocity and its changes are described by 12 parameters, therefore, in the notation proposed in section 2.3, it is a M_0^{12} mapping. The user can interactively move this probe through the data set.

Arrows can be used to show the magnitude and direction of the flow velocity at a point. However, their projection leads to ambiguities. Also the amount of information presented by a single arrow icon is small, while the use of many arrows easily leads to cluttering. The probe presented here can be regarded as an extension and improvement to the arrow icon. The probe shows the local flow by showing how the velocity changes at a point. The velocity changes are described by a tensor. By decomposition of this tensor into meaningful components and their mapping using metaphors that are easy to understand, the tensor is presented in an intuitively clear way. The result is that the probe presents information on a small region in the flow. In addition to techniques that present a global overview of the data the probe can be used for the interactive study of details in the data.

In this chapter, we will first discuss the construction of the probe. Then some examples of the use of the probe are given. Section 3.3 is reserved for elaboration on an idea used in the probe, a plane everywhere perpendicular to the flow. It is shown that it is not possible to generate such a surface in an arbitrary 3D flow, and finally conclusions are drawn.

3.1 Construction of the probe

To visualize local flow via a probe, meaningful quantities must be derived from the raw data. This is done in four steps. First, we calculate the velocity gradient tensor; second, we transform this tensor to a local coordinate frame; third, we decompose it into meaningful components. Finally we map these components onto geometric primitives. Figure 3.1 shows the steps needed for the construction of the probe.

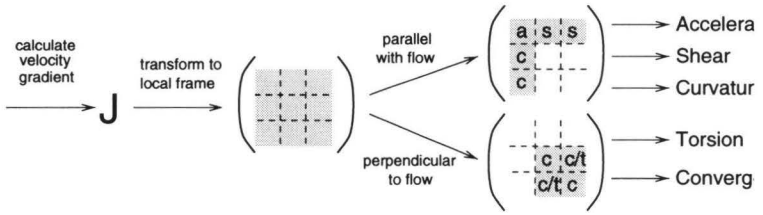


Figure 3.1: Decomposition of the probe

3.1.1 Velocity gradient tensor

In the probe the changes in velocity will be visualized. These changes are represented by the gradient of the velocity function. The velocity gradient is a second order real tensor. In a vector field the first order approximation of $\mathbf{u}(\mathbf{x})$ near a point $\mathbf{x}_0$ is:

$$\mathbf{u}(\mathbf{x}) = \mathbf{u}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x} - \mathbf{x}_0). \tag{3.1}$$

In three dimensions the velocity gradient tensor or Jacobian matrix $\mathbf{J}$ [Batchelor 1967] of a vector field $\mathbf{u} = (u, v, w)$ is given by:

$$\mathbf{J} = \nabla\mathbf{u} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix} \tag{3.2}$$

A subscript denotes a partial derivative, for example u_x is short for $\partial u/\partial x$. It is clear that to show local flow characteristics in a point, the velocity vector and the nine values from the gradient tensor must be visualized.

We calculate the velocity gradient tensor by differentiation, using finite differences. Centred differences using velocity values at half of the cell size are used. For example u_x at position $\mathbf{x}$ is calculated by the equation:

$$u_x = \frac{u(x + \frac{1}{2}\Delta x, y, z) - u(x - \frac{1}{2}\Delta x, y, z)}{\Delta x}, \tag{3.3}$$

where Δx is the size of the cell in the x-direction. The other derivatives are determined similarly. In uniform rectangular grids the cell-size does not vary over the data set and

the value can be precalculated once. In curvilinear grids the cell size has to be calculated from the grid nodes in which the point lies. Each of the three velocity components is differentiated with respect to three spatial directions, resulting in nine values. These values of a real tensor can be interpreted in terms of the distortion of an infinitesimal fluid element around the point where the tensor is defined.

3.1.2 Local coordinate frame

To get a clear representation, the calculated tensor is transformed to a local coordinate frame. In fluid flow a distinction can be made between velocity changes *parallel* with the local flow direction and velocity changes *perpendicular* to the flow direction. We use this distinction to decompose the tensor into two parts. Instead of a fluid element, we use a *fluid line* in the direction of the flow and a *fluid plane* perpendicular to the flow to represent the distortion. To represent distortion with such components, the velocity gradient tensor is transformed to a Frenet coordinate frame [do Carmo 1976] with respect to the streamline through the point of interest. The origin of this frame is the point where the tensor is calculated. The tangent vector $\mathbf{t}$ of this frame is aligned with the direction of the flow, the normal vector $\mathbf{n}$ points in the direction of the local curvature. The binormal $\mathbf{b}$ is perpendicular to the other two.

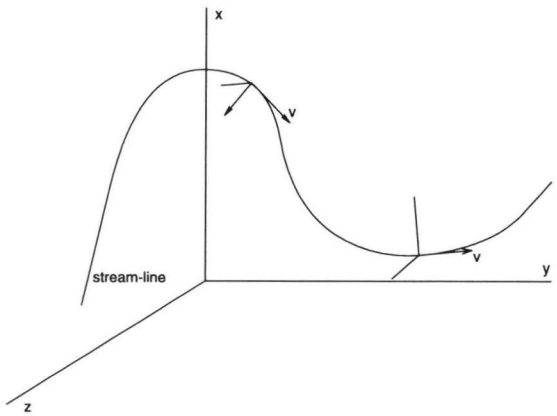


Figure 3.2: Two local coordinate frames

The curvature of the streamline through the origin has to be calculated, to define a Frenet coordinate frame. The curvature vector $\mathbf{c}$ of a parametric curve $\mathbf{p}(s)$ is given by:

$$\mathbf{c} = \frac{d^2\mathbf{p}}{ds^2} \tag{3.4}$$

where s is the parameter of arc length. On a stream line s is related to the velocity by:

$$s = \int_0^t \left|\frac{d\mathbf{p}}{dt}\right| dt = \int_0^t |\mathbf{u}(t)|dt \rightarrow \frac{ds}{dt} = |\mathbf{u}(t)| \tag{3.5}$$

Using the first order approximation of the velocity of an imaginary particle that moves along the streamline

$$\mathbf{u}(t) = \mathbf{u}_0 + \mathbf{J}\mathbf{u}_0 t; \tag{3.6}$$

the derivative of $\mathbf{p}$ with respect to s

$$\frac{d\mathbf{p}}{ds} = \frac{d\mathbf{p}}{dt}\frac{dt}{ds} = \frac{\mathbf{u}}{|\mathbf{u}|}; \tag{3.7}$$

and

$$\frac{d}{dt}|\mathbf{u}| = \frac{\mathbf{u}\cdot\mathbf{J}\mathbf{u}}{|\mathbf{u}|}; \tag{3.8}$$

the curvature vector at a point in the flow is given by:

$$\mathbf{c} = \frac{d}{ds}\frac{d\mathbf{p}}{ds} = \frac{d}{dt}\left(\frac{d\mathbf{p}}{ds}\right)\frac{dt}{ds} = \frac{d}{dt}\left(\frac{\mathbf{u}}{|\mathbf{u}|}\right)\frac{1}{|\mathbf{u}|}. \tag{3.9}$$

If we do the differentiation with respect to t and apply the quotient rule for differentiation we get the following expression for the curvature vector:

$$\mathbf{c} = \frac{\mathbf{J}\mathbf{u}(\mathbf{u}\cdot\mathbf{u}) - \mathbf{u}(\mathbf{u}\cdot\mathbf{J}\mathbf{u})}{|\mathbf{u}|^4}. \tag{3.10}$$

With this vector and the velocity vector the Frenet-frame can be constructed. The base of the frame consists of the normalized velocity vector, the normalized curvature vector and the binormal which is the cross product of these vectors:

$$M = \left(\frac{\mathbf{u}}{|\mathbf{u}|}, \frac{\mathbf{c}}{|\mathbf{c}|}, \frac{\mathbf{u}\times\mathbf{c}}{|\mathbf{u}\times\mathbf{c}|}\right). \tag{3.11}$$

If the curvature is zero the Frenet-frame is not defined. In this case any orthonormal frame with one axis aligned with the velocity can be used. For the probe, the curvature vector of a previously calculated frame projected onto the plane perpendicular to the velocity is used.

The inverse M^{-1} of the Frenet frame matrix expressed in global coordinates (Equation 3.11) is used to transform the velocity gradient tensor to a local coordinate frame

$$J_l = M^{-1}J. \tag{3.12}$$

In the remainder of this section we will use the local frame as our frame of reference. All references to coordinates and velocities are with respect to the local coordinate frame.

3.1.3 Decomposition

The transformation to a local coordinate frame allows us to decompose the tensor into two parts, one part that describes velocity changes in the direction of the velocity and a part containing changes perpendicular to the velocity. The components in the velocity direction are the derivatives with respect to x (u_x, v_x and w_x) and the derivatives of u (u_x, u_y and u_z). Note that u is equal to $\mathbf{u} = (u, 0, 0)$ because of the transformation to a local coordinate frame. The components in the perpendicular direction (v_y, v_z, w_y and w_z) form the 2×2 lower-right submatrix of the Jacobian matrix (see Figure 3.1).

To get meaningful primitives these two parts are further subdivided into components. The matrix describing velocity changes perpendicular to the velocity is split in a symmetric and anti-symmetric part. The symmetric part describes the convergence of the flow, the anti-symmetric part relates to the torsion in the flow. The velocity-parallel components are split in three parts. The top-left element of the matrix (u_x) gives the acceleration. The shear is represented by the remaining two elements in the top row (u_y and u_z). The lower two elements in the first column (v_x and w_x) give the curvature of the field. Because a Frenet frame is used, the curvature vector by definition points in the direction of the y-axis, therefore the value of w_x is always zero.

3.1.4 Mapping of the probe components

We now describe the mapping of the velocity and the components of the tensor to geometrical primitives of the probe. Figure 3.3 shows a graphical representation of the probe. The figure shows the geometric primitives onto which the components are mapped. On the left the probe shows the shape of the primitives if the associated value is zero i.e. constant flow. The right side shows the probe if the components are non-zero.

Velocity

Velocity is represented by an arrow in the form of a cylinder with a conic tip. Because of the use of a cylinder instead of a line, its direction can be perceived from the shading of the cylinder. The length and direction of the shaft of the arrow represent the magnitude and direction of flow velocity. In the local coordinate frame, velocity has only a component in the x-direction, therefore the magnitude of the velocity can be referred to as a scalar value u. The length of the velocity arrow is determined by the time scale Δt used. This also affects primitives related to the velocity (curvature, shear and acceleration). Thus, the length of the arrow shaft is given by:

$$l = u\Delta t. \tag{3.13}$$

Curvature

Curvature is visualized by the shaft of the velocity arrow of the probe; the shaft is bent such that it forms an arc of the osculating circle [do Carmo 1976] to the streamline (see

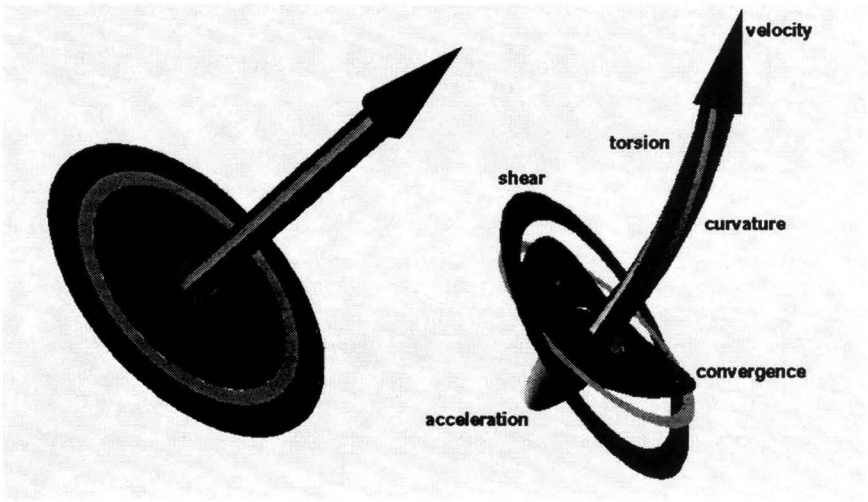


Figure 3.3: Components of the probe. Left: all gradient values are zero. Right all gradient values are non zero (see also Colour plate C.1 in the colour section)

Figure 3.4). The curved arrow thus shows a second order approximation of a streamline through the origin of the frame. The curvature of the streamline is given by equation 3.10. Because of our choice of the local coordinate frame the curvature vector points in the direction of the y-axis. The centre of the osculating circle of the streamline is given by:

$$\mathbf{q} = \begin{pmatrix} 0 \\ u/v_x \\ 0 \end{pmatrix}. \tag{3.14}$$

It is clear that the radius of the osculating circle becomes larger if the velocity increases and becomes smaller if the curvature increases. Because the arrow is bent we cannot apply equation 3.13 directly to the arrow but have to apply it to the arc length of the shaft instead. Therefore equation 3.13 has to be rewritten into an expression which gives an angle β over which the arc extends. It is given by:

$$\beta = \frac{u\Delta t}{|\mathbf{q}|} = \Delta t v_x. \tag{3.15}$$

Acceleration

The value of u_x determines the tangential acceleration of a particle released at the origin of the reference frame. This acceleration is visualized by a *membrane* perpendicular to the flow. The value of the acceleration is mapped to the displacement of the centre of the

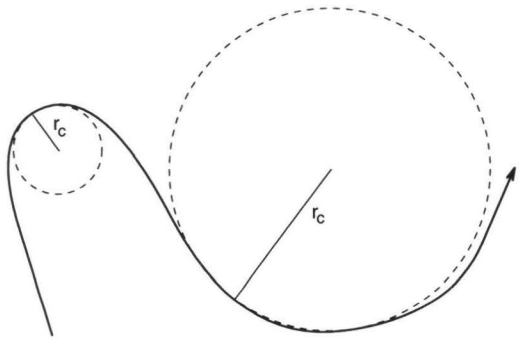


Figure 3.4: Visualization of curvature

membrane. Zero acceleration results in a flat membrane. If the acceleration is positive, a half-ellipsoid in the direction of the velocity is shown, if it is negative (deceleration), the centre of the membrane is stretched in the opposite direction. Thus, it shows the velocity difference between the probed point and the end of the velocity arrow. The displacement a of the centre of the membrane is given by:

$$a = u_x \Delta t^2 u. \tag{3.16}$$

The square in the time step Δt is needed because both the length of the arrow and the duration of the velocity change has to be taken into account.

Shear

Shear in the direction of the flow can be interpreted as the change of orientation of the plane perpendicular to the flow over time (see Figure 3.5). Only a ring-shaped part of this plane is shown, centred around the probed point. The equation of the plane of the ring is:

$$x = \Delta t(u_y y + u_z z). \tag{3.17}$$

Torsion

Torsion r is rotation of a fluid element around the axis oriented in the direction of the velocity. Using the values in the velocity gradient the torsion can be written as

$$r = w_y - v_x. \tag{3.18}$$

Candy stripes on the surface of the velocity arrow's shaft are used to represent torsion. The total torsion angle α around the axis, taking in account the length of the arrow, is

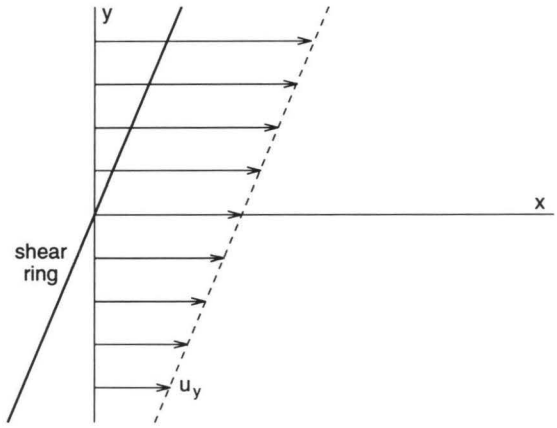


Figure 3.5: Visualization of shear in the probe

given by:

$$\alpha = r\Delta t. \tag{3.19}$$

Torsion is related to helicity of the flow; helicity is the product of torsion and velocity: $\mathbf{u} \cdot \nabla \times \mathbf{u}$, which is equal to $u(w_y - v_x)$ in the local coordinate frame. Therefore, the helicity of the flow can be easily deduced from the probe.

Convergence and divergence

Velocity changes in the plane perpendicular to the flow are related to convergence and divergence of the flow. They affect the size and orientation of a circle released perpendicular to the flow. For visualization, a surface that is everywhere perpendicular to the flow is used. An intuitively clear metaphor for such a surface is a lens that focuses the flow (see Figure 3.6). Here we use the osculating paraboloid [do Carmo 1976] to this surface to represent its curvature, just as we used an osculating circle to represent the curvature of a streamline. The distance $\delta(\eta, \xi)$ of an osculating paraboloid to the tangent plane with unit normal $\mathbf{N}$ parameterized by η and ξ is given by:

$$\delta(\eta, \xi) = \frac{1}{2}(Ld\eta^2 + 2Md\eta d\xi + Nd\xi^2), \tag{3.20}$$

where

$$\begin{array}{rcll} L & = & -(\mathbf{x}_\eta \cdot \mathbf{N}_\eta) & , \\ M & = & -\frac{1}{2}[(\mathbf{x}_\xi \cdot \mathbf{N}_\eta) + (\mathbf{x}_\eta \cdot \mathbf{N}_\xi)] & \text{and} \\ N & = & -(\mathbf{x}_\xi \cdot \mathbf{N}_\xi) & . \end{array}$$

For the osculating paraboloid to the surface perpendicular to velocity ($\mathbf{x}_\eta \cdot \mathbf{u} = \mathbf{x}_\xi \cdot \mathbf{u} = 0$), L is given by :

$$L = -\mathbf{x}_\eta \cdot \mathbf{N}_\eta = -\mathbf{x}_\eta \cdot \frac{\mathbf{u}_\eta}{|\mathbf{u}|} \tag{3.21}$$

M and N can be determined similarly. If y and z of the reference frame are chosen as the surface parameters η and ξ, then the equation for the surface perpendicular to the velocity reduces to:

$$x = -\frac{v_y}{u}y^2 - \frac{v_z + w_y}{u}yz - \frac{w_z}{u}z^2. \tag{3.22}$$

This equation is used to construct the surface.

Using a disc with a fixed radius poses problems when the differentials are large: the surface will become very large and the edge of the surface will be far from the probe. To prevent this, the edge of the surface rendered is the intersection of the surface defined by equation 3.22 and a sphere around the centre of the probe.

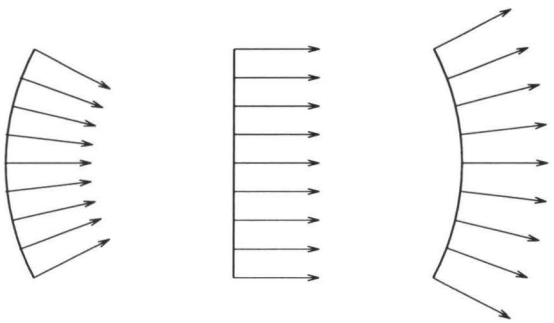


Figure 3.6: Visualization of convergence in the probe

3.2 Examples

In this section we present two applications of the local flow probe. In the first example a data set of a storm is used. The second one shows the use of the probe in a time dependent flow of a moving piston.

3.2.1 Interactive exploration of a storm

As was argued in chapter 2, the possibility of exploration is an important aspect of visualization. Exploration is possible with a probe that can be placed at arbitrary locations by the user. We developed an interactive application in which a user can explore flow data sets using the local flow probe.

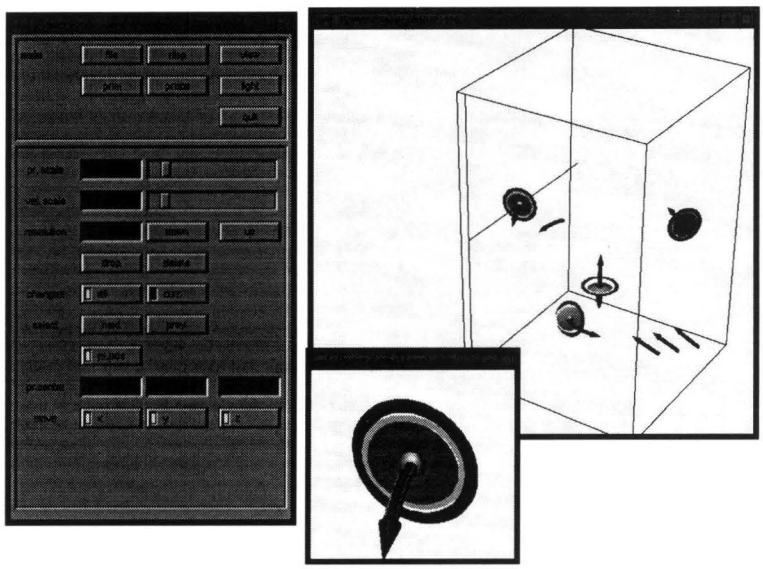


Figure 3.7: User interface of the application (see also Colour plate C.2 in the colour section)

The application provides the user with two views of the flow area (Figure 3.7). The first is a global view, which shows the position of the probe in the flow area. The second is a close-up view of the probe. The user can interact with the probe in both views. In the global view, the user can move the probe through the flow area. The position of the probe is indicated by two lines connected to the bounds of the flow area. When the probe is moved, the shapes of the probe's components are continuously updated in both views to reflect the local flow features. In the close-up view, the user can choose a different point of view for close inspection of the probe. The third window is a button window made using the *Forms* user interface toolkit [Overmars 1992]. In this window the user can perform additional interaction such as exact placement of the probe using numerical coordinates.

To evaluate the application we used a data set generated by a numerical simulation of a tornado. Figure 3.8 shows how the probe is applied to explore the data. A collection of probes at various positions in the data set give insight in the general structure of the flow. The user can interactively place copies of the probe to show interesting locations in the flow. The user can also generate streamlines, and let a probe move along it. Figure 3.8 shows a spiralling streamline with several probes.

The changes of velocity in the vertical direction in eye of the tornado are clearly visible. The velocity in the eye of the tornado is upward, however the magnitude varies.

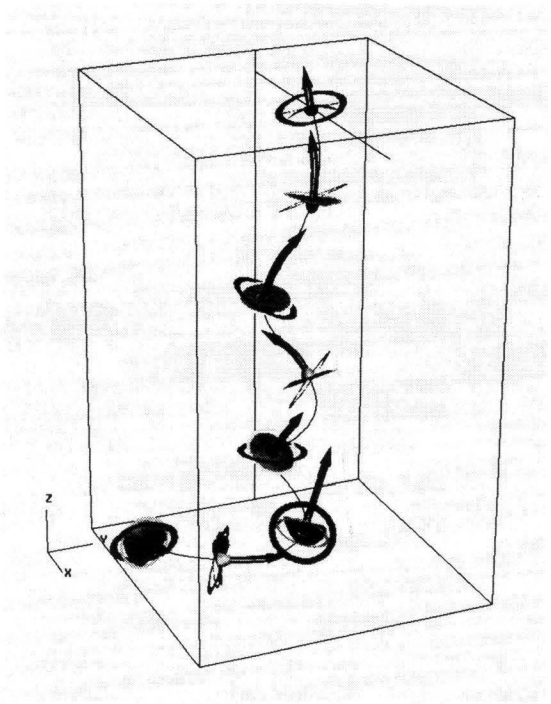


Figure 3.8: Interactively built image of the tornado data set, with probes located on a stream line. Data courtesy: R. Wilhelmson (NCSA).

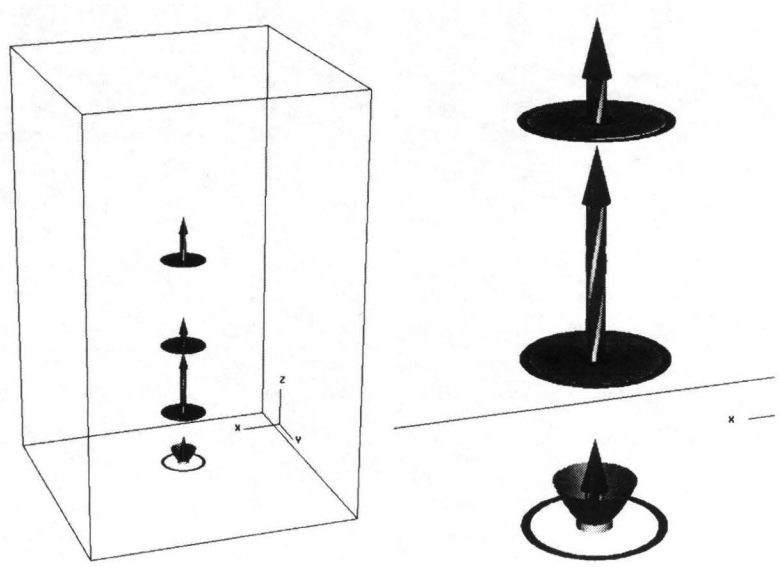


Figure 3.9: Left: probes in the tornado data set showing positions of zero acceleration in core of the vortex and convergence area at the bottom of the core, right: detail

Using the probe the locations where the flow changes from acceleration to deceleration can be easily found by looking at the acceleration part of the probe which is flat in these points. In figure 3.9 probes are positioned at these points.

3.2.2 Flow in a two-stroke engine

In this section we will describe an example of the use of the probe in a time-dependent flow. The case was performed by Monika Wierse and Rolf-Thomas Happe at the Institute for Applied Mathematics in Freiburg. More about it can be found in [Wierse 1994] and [Wierse, Geßner & Kröner 1995]. This flow is determined by the numerical solution of the non-stationary three dimensional compressible system of Euler equations. This equations are solved in a model of a two-stroke engine, with a moving piston (see Figure 3.10). The numerical program uses a finite volume method where the components of the system are approximated as piecewise constant on each tetrahedron.

Because the flow is non-stationary, some modifications to the probe were made in order to incorporate time-dependency. For the components in the direction of the flow, acceleration and curvature, the derivatives of a particle path were used instead of the spatial derivatives. Instead of an approximation of a streamline, which would be the case if the unmodified probe is applied to a time-dependent flow, an approximation of a

Figure 3.10: The shape of the grid used for the two-stroke engine

particle path was used to construct the probe. The changes in mathematics are described below.

Beside the velocity changes with respect to space, also the velocity changes with respect to time have to be taken into account. Let $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ be the velocity and $\mathbf{x} = \mathbf{x}(t)$ a particle path. Then $\mathbf{a}$, the derivative of $\mathbf{u}$ with respect to t, can be written as:

$$\mathbf{a} = \frac{d}{dt}\mathbf{u}(\mathbf{x}(t), t) = \mathbf{u}_t + \mathbf{J}\mathbf{u} \tag{3.23}$$

Where $\mathbf{u}_t$ is the partial derivative for time of $\mathbf{u}$. Using $\mathbf{a}$ instead of $\mathbf{J}\mathbf{u}$ the curvature vector given in Equation 3.10 is rewritten into:

$$\mathbf{c}^\star = \frac{d^2\mathbf{x}}{ds^2} = \frac{(\mathbf{u}\cdot\mathbf{u})\mathbf{a} - (\mathbf{u}\cdot\mathbf{a})\mathbf{u}}{|\mathbf{u}|^4} \tag{3.24}$$

The Frenet-frame is constructed in the same way as described in Section 3.1.2 using $\mathbf{c}^\star$ instead of $\mathbf{c}$. The centre of the osculating circle in this local coordinate frame, which is used to show the curvature, is given by:

$$\frac{1}{|\mathbf{c}^\star|^2}\mathbf{c}^\star. \tag{3.25}$$

The acceleration primitive determined by the distance in tangential direction the particle travels due to $\mathbf{a}$ is:

$$a = \frac{1}{2|\mathbf{v}|}(\mathbf{v}\cdot\mathbf{a})\Delta t^2 \tag{3.26}$$

The other components of the probe: convergence/divergence, torsion and shear rely on derivatives in directions perpendicular to the flow only, and the time dependency of $\mathbf{u}$ can be neglected. Note that if this modified probe is used in a stationary flow ($\mathbf{u}_t = 0$), the probe coincides with the original.

Figure 3.11 shows some positions where the flow is coming from one inlet and continues immediately to the outlet. Furthermore the corresponding shear effects can be identified and the acceleration due to the piston motion. The torsion has been scaled, otherwise the stripes would have been almost straight. I.e. the flow is not as helical as it seems to be.

3.3 The flow front

The osculating paraboloid used to show convergence (Section 3.1.4) suggests another mapping for flow data: a surface which is everywhere perpendicular to the flow. However, as will be shown, this is not possible in general three dimensional flow, but only under certain conditions. Before dealing with the problem in arbitrary 3D flows we will describe the existence of lines perpendicular to the flow in 2D and we will show the existence of such surfaces in a special type of 3D flow: a potential flow.

In two dimensions it is always possible to construct a line that is everywhere perpendicular to the flow by integration starting from a *seed point* and integrating perpendicular to the flow. In Figure 3.12 the principle is demonstrated. The idea is to turn each vector over an angle of 90^o and generate tangent curves in this modified field.

Potential flows are characterized by the existence of a potential p, a scalar function over space which is related to velocity in the following way:

$$\mathbf{u}(\mathbf{x}) = \nabla p(\mathbf{x}) \tag{3.27}$$

The gradient of a scalar field is always exactly perpendicular to the surface defined by a collection of points with equal value, an isosurface. Therefore, the flow front through a point is the same as the iso-potential surface of the field value at that particular point. From this we can conclude that a flow front exists for all potential flows. However for flows that do not satisfy equation 3.27, a potential cannot be defined and therefore we have to use another approach.

For a general treatment we will assume a vector field $\mathbf{u}(\mathbf{x})$. We will use the following parameterization of the flow front to simplify the mathematics. At the seed point we define a local coordinate frame (see Section 3.1.2) with the x-axis aligned with the velocity so the y- and z-axis are perpendicular to the flow in the seed point. These two perpendicular axes are used to parameterize the flow front (see Figure 3.13). This is only possible if nowhere on the flow front the velocity is perpendicular to the velocity at the seed point. This is no problem because a continuous field is assumed, and we are only interested in a small region around the seed point. Using this parameterization, a point on the flow front can be uniquely specified by a (y, z) pair. Each pair of values for y and z give a value for x, which is the distance from the normal plane at the seed point to the flow front.

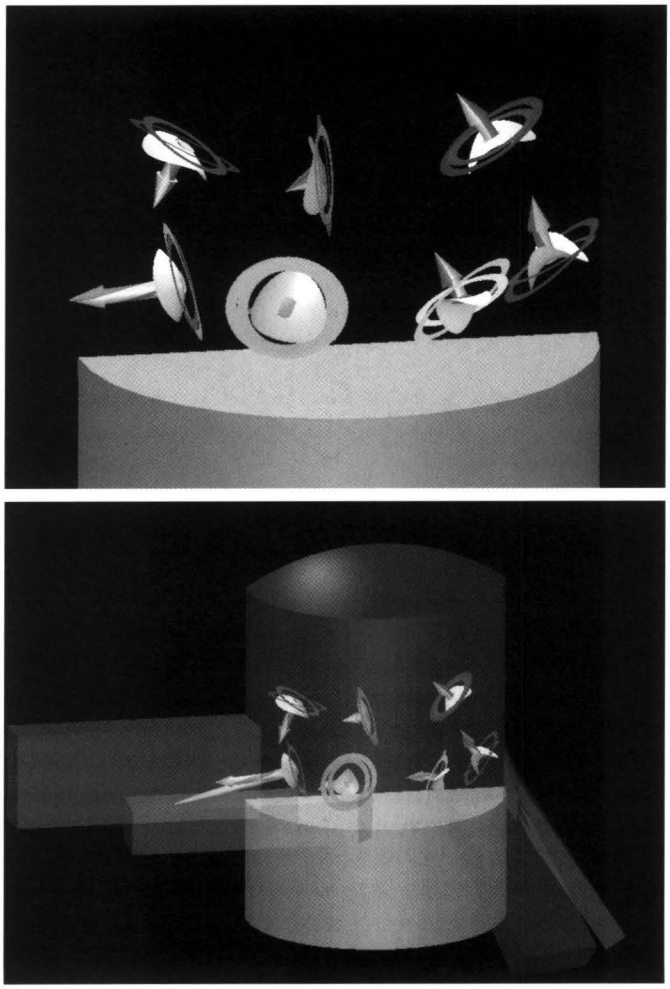


Figure 3.11: Local flow probes show shear and acceleration effects in two-stroke engine

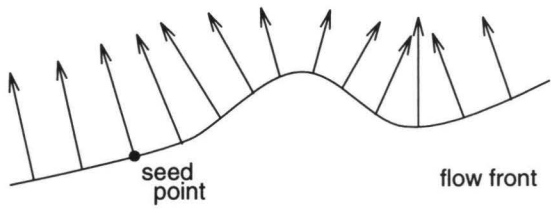


Figure 3.12: 2D equivalent of flow front

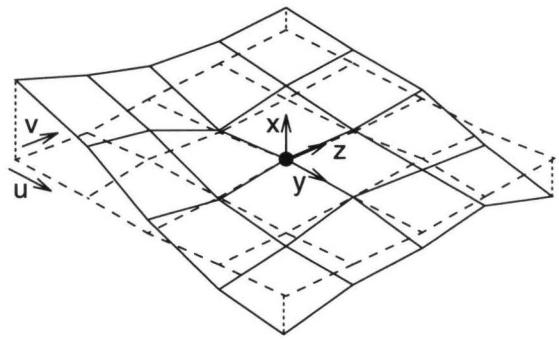


Figure 3.13: Parameterization of the flow front

In the flow front there are *front lines*. These are lines everywhere perpendicular to the velocity. It is obvious that any front line through the seed point must lie in the flow front and that any line in the flow front is a front line. By determination of a large number of front lines we can determine the flow front. Front lines can be calculated by numerical integration perpendicular to the flow just like in the 2D case. Note that we are free in choosing the direction of a front line as long as it remains in the flow front.

Using the parameterization described previously, it is clear from Figure 3.14 that we can describe the discrete integration of a front line in the y-direction in point p by the following equation:

$$\frac{\partial x}{\partial y} = \frac{v_p}{u_p} \tag{3.28}$$

A step in the z-direction is essentially the same:

$$\frac{\partial x}{\partial z} = \frac{w_p}{u_p} \tag{3.29}$$

For the existence of a flow front it is necessary that different front lines starting and ending at the same point give the same value for x in the common endpoint. We will check this condition in the following way (see Figure 3.15): integrate along the front

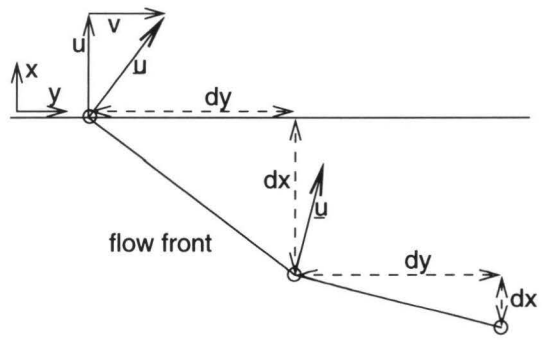


Figure 3.14: Calculation of a line in a flow front

lines ABD and ACD. Each step has a length of Δ. The first line starts in the y-direction for one integration step and the second step is in the z-direction. The second line starts in the z-direction and the second step is in the y direction. Therefore both lines end in point D.

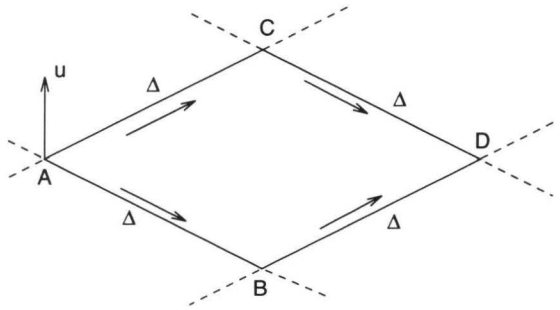


Figure 3.15: Point D on flow front calculated using different integration paths

To calculate x_D we use the first order approximation of the velocity in the neighbourhood of the seed point (Equations 3.1 and 3.2). With these equations we can calculate the x-value for the points B, C and D (see Figure 3.15). At the seed point A, u and v are zero because of the choice of coordinate frame. Thus

$$\begin{aligned} x_B &= \tfrac{v_A}{u_A}\Delta = 0 \\ x_{D_1} &= x_B + \tfrac{w_B}{u_B}\Delta = \tfrac{w(0,\Delta,0)}{u(0,\Delta,0)}\Delta \\ x_C &= \tfrac{w_A}{u_A}\Delta = 0 \\ x_{D_2} &= \tfrac{v_C}{u_C}\Delta = \tfrac{v(0,0,\Delta)}{u(0,0,\Delta)}\Delta \end{aligned} \tag{3.30}$$

To get a good representation of the data values the samples taken should be distributed evenly over the sampling domain, i.e. the sampling density should be constant over the sampling domain. Vector fields are represented as a grid of vector values. These grid values can be used as input for our statistical method if this grid is uniform. If this is not the case, other sampling patterns should be chosen because the grid density can vary strongly over the sampling domain. Interpolation has to be used if a sampling pattern does not coincide with the grid values. If the distribution of samples is not even over the domain, then weighting factors must be used. Each sample is assigned a weighting factor that is inversely proportional to the local sampling density.

When using non-grid-point sampling, in order to get a good representation of the field, samples should be taken from all cells of the domain of interest. Although statistical quantities can be calculated for an arbitrary-sized region, care has to be taken for a region covering only one or just a few cells. Different samples from a single cell are determined by the same data values.

4.1.1 Distribution function

In order to apply statistics to values in the vector field we consider the distribution of vector values in the region of interest: a function which maps the vector value (e.g. velocity) to a probability $p(\mathbf{v})$. This function gives the probability that a certain velocity $\mathbf{v}$ occurs in the region of interest. It is a three dimensional scalar function. Integrating this function over all possible velocities has 1 as a result, because it is a distribution function.

$$\int p(\mathbf{v})d\mathbf{v} = 1. \tag{4.1}$$

This equation assumes a continuous representation of the velocity distribution. What we have is a collection of samples. However these samples are considered to be taken from a continuous function.

4.1.2 Average velocity

An important property of the selected domain D is the average velocity $\overline{\mathbf{v}}$. It is calculated by the equation

$$\overline{\mathbf{v}} = \int_{-\infty}^{\infty} \mathbf{v}p(\mathbf{v})d\mathbf{v} = \frac{1}{V_D}\int_D \mathbf{v}dD. \tag{4.2}$$

where V_D is the volume of the domain. If we assume a rectangular grid with vector values given at grid nodes, and tri-linear interpolation inside the cells, then the average velocity in a cell is the average of the velocities at the eight corners of the cell:

$$\overline{\mathbf{v}} = \frac{1}{8}\sum_{i=1}^{8} \mathbf{v}_i \tag{4.3}$$

By summation of cell averages, the average over the region of interest is calculated. Special care has to be taken for cells partly inside the region of interest.

4.1.3 Variance/covariance matrix

Representing a field by an average is useful under certain circumstances, however information about spatial distribution of data is lost. More information is represented by higher order moments of the distribution. The multidimensional equivalent of the second-order moment (standard deviation) is the variance/covariance matrix $\mathbf{E}$ expressed continuously as:

$$\mathbf{E} = \frac{1}{V_D}\int_D \mathbf{v}\mathbf{v}^T dD - \frac{1}{V_D}\int_D \mathbf{v} dD \frac{1}{V_D}\int_D \mathbf{v}^T dD \tag{4.4}$$

and sampled as:

$$\mathbf{E} = \frac{1}{N-1}\begin{pmatrix} E_{1,1} & E_{1,2} & E_{1,3} \\ E_{2,1} & E_{2,2} & E_{2,3} \\ E_{3,1} & E_{3,2} & E_{3,3} \end{pmatrix}, \tag{4.5}$$

where

$$E_{i,j} = E_{j,i} = \sum_{k=1}^{N}(x_{i,k} - \overline{x}_i)(x_{j,k} - \overline{x}_j)$$

and where $x_{i,k}$ is the k-th sample of the j-th component of the data vector x and N is the number of data samples. This matrix is a symmetric second-order tensor. Using principal components analysis from multivariate analysis [Kendall 1965], the principal components of the distribution are calculated. Principal components are the eigenvectors of the variance/covariance matrix.

Higher-order moments of the distribution function such as skew and kurtosis can be determined easily. Their usefulness for the visualization of vector fields is debatable because these metrics provide information on the data based on the deviation from a normal distribution. The distribution function however, can have any shape; therefore other indicators to describe the shape are more useful.

4.2 Statistical visualization

In the previous section we discussed how statistical properties of selected regions from vector fields can be calculated. In this section we will discuss the visualization of these properties. The properties and therefore their visualization is independent of the size and shape of the region of interest. Using statistical methods for data representation in combination with techniques to specify a region of interest, a user can investigate the data at different levels. If the information presented at a certain level invites further investigation, the region of interest is reduced, thereby allowing investigation at a smaller scale.

The velocity distribution function is a $R^3 \rightarrow R$-function. Direct visualization of the function by existing techniques, for example volume rendering or isosurfaces, is possible. However this is not very intuitive and hard to interpret. The reason is that the space which is visualized is *velocity space*. The distribution function maps a velocity

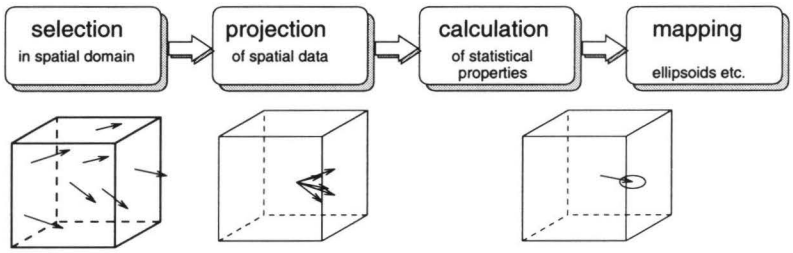


Figure 4.1: Generation of a statistical representation

to a probability, the units along the axis would be m/s. Therefore we use the statistical properties calculated from the distribution function for the visualization.

The process of visualizing statistical data of vector fields can be described by a sequence of steps (see Figure 4.1).

- Selection

  First, a region in space is selected: an axis-aligned box is used in the following discussion, but the techniques described can be easily generalized to arbitrarily shaped subsets of data, for example those obtained by the selections as proposed by van Walsum [van Walsum 1993*a*, van Walsum 1993*b*].

- Projection

  The next step in the pipeline is *projection*. The spatial dimensions in the data are projected to a point, i.e. relative positions within the domain of interest are ignored.

- Calculation

  After projection we calculate statistical properties: the individual values are replaced by a statistical description. A further reduction of data takes place at this step. The amount of reduction can be varied; the user can choose to generate only the average or more complex characteristics which show more detailed information on the distribution.

- Mapping

  The last step in the process is the mapping of the calculated statistical properties to geometric primitives. For different statistical characterizations of the data suitable mappings must be found to obtain an understandable picture of the data.

The Variance/Covariance Matrix (VCM) is visualized by an ellipsoid [Fung 1965, Silver et al. 1991] (see Figure 4.2). The three axes of this ellipsoid reflect the size and direction of the principal components of the VCM. This ellipsoid can be interpreted as a surface of equal probability for the normal distribution with equal average and variance

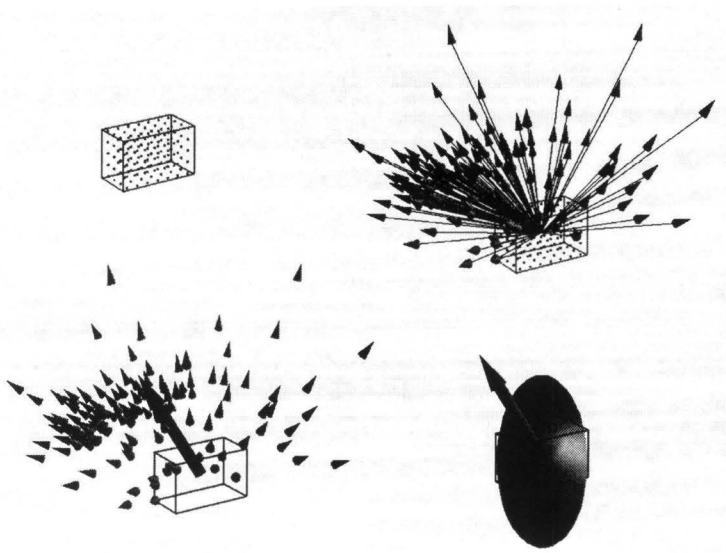


Figure 4.2: Construction of the statistical probe, showing the mapping to graphical primitives (see also Colour plate C.3)

as the given distribution. Instead of using a closed ellipsoid other mappings reflecting the shape of the ellipsoid which do not fully occlude the interior space are possible. Examples of such shapes can be found in Post et al. [Post, van Walsum & Post 1995].

A better approximation of the distribution function is achieved using higher order moments of the distribution function. The results of these calculations are higher order tensors. Interpretation of this data in relation to the actual vector field is difficult. A better approach in our view is the use of simple fitting or smoothing functions to model the velocity distribution. In case of bimodal or other distributions which are badly modelled by a normal distribution, this is a good alternative. The chosen fitting functions can be of arbitrary complexity, with the distribution function itself as a limiting case. Visualization of the (approximated) distribution function is possible using isosurfaces of probability density. If simple functions are used this is easier to understand than direct visualization of the distribution function with isosurfaces.

A different approach to represent the distribution function is by showing a number of samples, instead of a continuous representation. If a box is used as a region of interest, a regular grid of sample points is used to take samples. The values found can be used to generate a three dimensional scatter plot of the distribution function in velocity space. The velocity samples are shown as points in space. They can be represented as dots or small spheres. However the result is ambiguous because of the projection onto the screen. Therefore we used cones instead; this partly solves the projection ambiguity.

4.3 Statistical data exploration

So far we concentrated on the calculation and mapping of statistical properties given a region of interest without considering how this region was specified. Here we will describe the techniques used to specify the position and size of the box shaped region of interest. Three techniques will be explained: corner-dragging, box-dragging and box-splitting.

By corner-dragging the user can specify the corners of an axis-aligned box by direct manipulation. While dragging a corner of the box the data displayed is continuously updated showing the information on the current box.

Using box-dragging the user can drag the box as a whole and change its size while the centre of the box remains at the same position. Also in this case direct manipulation is used and feedback on the new situation is given during the process.

The third possibility is box-splitting. If the user wants more detailed information on a region, a command is issued which splits the region along a chosen axis. This results in two equal-sized sub-domains which are both visualized independently. The user can select one of these, and continue exploration.

4.4 Examples

So far we discussed the calculation and mapping of statistical properties of data. In this section we will show two examples of visualization techniques based on this representation, applied to flow visualization.

4.4.1 Flux probe

If a polygon is specified as a region of interest, then the average velocity over the polygon shows the average flow through the area of the polygon (see Figure 4.4). In incompressible flow the component of this vector perpendicular to the polygon (see Figure 4.3) is the mass flux F through the polygon:

$$F = \frac{1}{N} \sum_{i=1}^{N} \mathbf{v_i} \cdot \mathbf{n}, \tag{4.6}$$

where N is the number of samples and $\mathbf{n}$ the unit normal on the polygon. Flux is important in many flow problems such as problems dealing with convection. The flux polygon can be of any size, shape and orientation, to be used for measuring flux anywhere inside a flow area.

4.4.2 Finding vortices

Instead of showing velocity, other vector quantities defined over the region of interest can be selected for visualization. Here we will use the curvature vector of a streamline

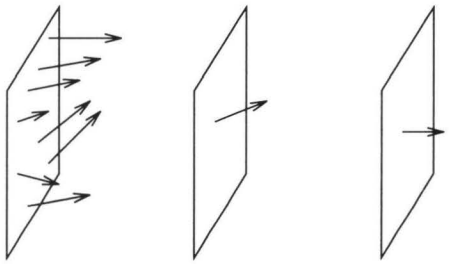


Figure 4.3: Steps in generation of the flux probe, sampling, averaging and projection

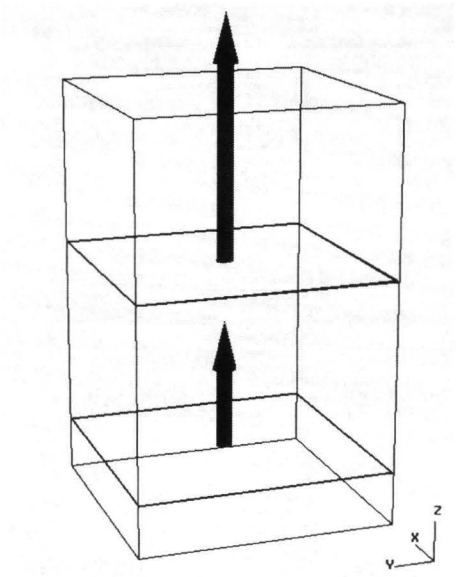


Figure 4.4: Flux probe used to compare the upward flux at two levels in the storm data set

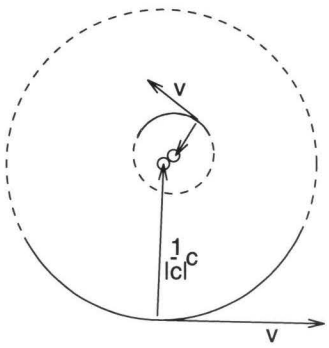


Figure 4.5: Using curvature to find vortex cores, the curvature in two points gives the estimated location of the rotation centre

through each point to locate vortex cores in the data. Using the curvature the centre of the osculating circle to the streamline is calculated (Figure 4.5).

First the curvature vector is calculated in a number of sampling positions in the region of interest, using the method described in Section 3.1.2. With this value, the osculating circle to the streamline is determined for each sample. The centres of these circles are estimates for the position of the vortex core. By drawing these points in the flow region, vortex cores can be located interactively (see Figure 4.6). This only works if the flow in the region of interest is mainly vortical. If there is no dominant centre of rotation there is a large variation in the estimated location of the rotation centre and images such as shown by Figure 4.7 result.

4.5 Conclusions

We described a method for the visualization of vector fields using statistical techniques. By projection the velocity values in an arbitrary region to a point, a 3D distribution function results. Using statistical techniques, characteristics of this distribution function are calculated. This results in efficient data reduction and a uniform presentation irrespective of the size and shape of the region.

Various methods based on statistical distribution parameters such as average and variance for the visualization of the distribution function are presented. Average and variance/covariance can be depicted by an arrow and an ellipsoid.

Two possible applications of the statistical method were given. The flux probe, which can be used to measure the flux through a polygon in the flow area, and the vortex finder which is useful for th interactive determination of vortices in the flow.

Representing a distribution function of velocity values by characteristics such as mean and standard deviation is easier to comprehend if the distribution function resembles a normal distribution. In some cases these characteristics can give a false impression

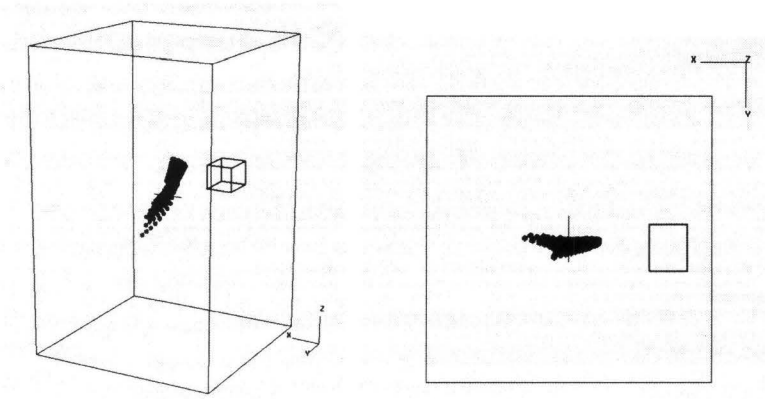


Figure 4.6: The vortex probe shows the centre of rotation in the storm data set, the location of the box is the same in both views

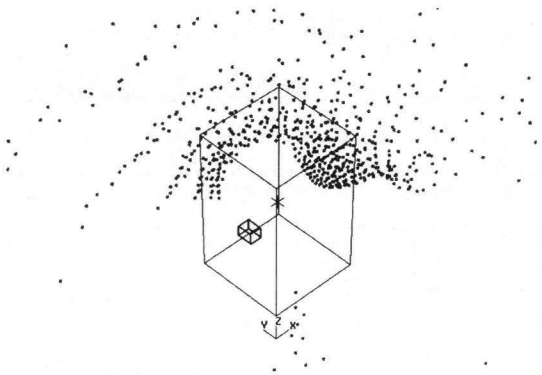


Figure 4.7: Vortex probe, used in a region without a dominant rotation

of the data. A test to decide if the distribution function resembles a normal distribution would be useful.

Chapter 5

Spot Noise

Visual information can be described by three visual primitives: shape, colour, and texture. Shape and colour have both been used extensively for the visualization of data. Texture is a visual primitive that has been much less explored. High-end workstations available today provide texture mapping implemented in the graphics hardware. Using this hardware large scenes of textured polygons can be rendered at interactive speed. Furthermore, texture can give a continuous view of a 2D field, while arrow plots or streamlines only show the results for discrete positions.

To use texture for visualization, a mapping of the data to texture parameters must be found. In this chapter a technique called spot noise will be described. With this technique textures can be generated that are suitable for the visualization of vector fields. First the algorithm itself will be described. After that some issues with regard to the visualization of vector fields using spot noise will be discussed. Then a comparison with another related texture synthesis technique called line integral convolution (LIC) will be given. Some results of the use of spot noise will be given in section 5.7.

5.1 Spot noise algorithm

In this section we will describe the ideas behind spot noise. Some mathematical background will be given. And we will show how spot noise can be applied to the visualization of vector fields.

5.1.1 Texture synthesis

A texture in the sense used for computer graphics is an arbitrary image. Such an image can be mapped to a three dimensional surface. If only the intensity of the image is of interest the texture can be defined by a scalar function f of a two dimensional position $\mathbf{x}$. A subclass of textures are the Gaussian textures which are described by a autocorrelation function $C_f(\boldsymbol{\tau})$. This function gives the correlation between two random samples at distance $\boldsymbol{\tau}$. For a two dimensional function $f(\mathbf{x})$ with a zero mean this function is

defined as

$$C_f(\boldsymbol{\tau}) = \langle f(\mathbf{x}) f(\mathbf{x} + \boldsymbol{\tau}) \rangle, \tag{5.1}$$

where $\langle \ldots \rangle$ denotes an average over a large number of samples. Another way to look at such textures is in the Fourier domain, especially the power spectrum. If $F_T(\omega)$ is the Fourier transform of a sample of a one dimensional function with length T, the power spectrum is defined by:

$$P_f(\omega) = \lim_{T \to \infty} \frac{1}{T} |F_T(\omega)|^2 \tag{5.2}$$

Generalization of this function to two dimensions is straightforward. It can be shown [van Wijk 1991] that both ways of describing a texture are equivalent, because they are a Fourier transform pair.

With spot noise, a Gaussian texture with a certain power spectrum can be generated in a way that is intuitively easy to understand. Spot noise [van Wijk 1991] is defined as

$$f(\mathbf{x}) = \sum a_i h(\mathbf{x} - \mathbf{x}_i), \tag{5.3}$$

in which $h(\mathbf{x})$ is called the spot function. It is an intensity function everywhere zero except for an area that is small compared to the texture size. Furthermore a_i is a random scaling factor with a zero mean, $\mathbf{x}_i$ is a random position. In non-mathematical terms: spots of random intensity are drawn and blended together at random positions on a plane (see Figure 5.1).

The mapping of a spot to a grey value in the texture is such that a zero value is mapped to the centre of the intensity spectrum. This is done to be able to handle negative values for spot intensity. Intensity values are mapped linearly to a grey value where black represents -1 and white $+1$. Intensity values higher than zero are rendered with an intensity higher than the grey level chosen as zero and negative values are rendered with a lower intensity than the zero level.

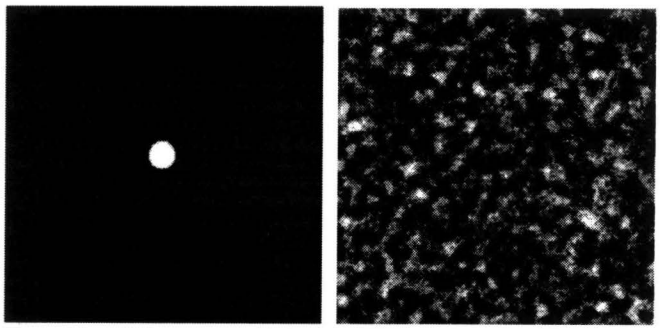


Figure 5.1: Principle of spot noise: single spot (left) resulting texture (right)

The use of a spot as a basis for a texture has two advantages. First, the spot shape determines the characteristics of the texture, and second, local control of the texture is

possible. In Figure 5.2 different spot shapes are used to generate spot noise. From this picture it is clear that the structures in the texture are related to the shape of a single spot. For example in the lower right image the edges of the triangle can be found as linear patterns aligned with the edges of the triangle in the texture. It can be shown that the power spectrum (Equation 5.2) of the spot and the resulting texture are equivalent except for a multiplication factor.

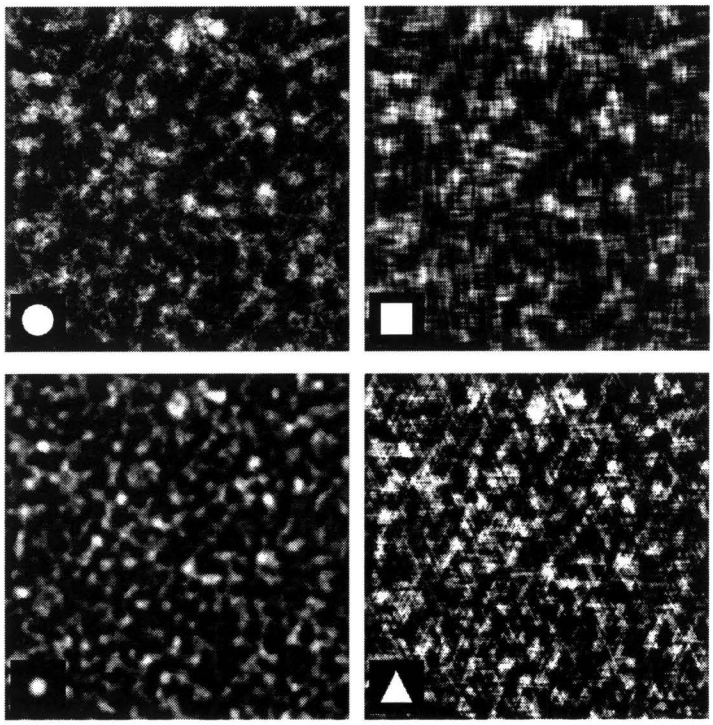


Figure 5.2: Different spot shapes result in different textures, in the bottom left corner of each texture the spot used is shown, magnified by a factor two

Variation of the spot shape depending on the position of the spot in the texture gives the possibility of local control of the texture. Vector fields can be effectively visualized, if the shape of the spot is adapted to the data at the position of the spot as is shown in Figure 5.3. Spots are scaled by a factor linearly proportional to the velocity magnitude in the direction of the flow and inversely proportional to the velocity magnitude in the direction perpendicular to the flow, thus keeping the surface of the spot constant. A disc-shaped spot is used for texture synthesis in the figure and all other figures in this chapter unless noted. This shape is used because other shapes of the spot tend to give false directional information.

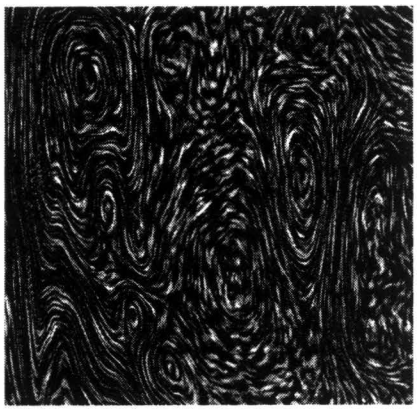


Figure 5.3: Spot noise used to visualize a vector field

Except for the spot shape, the final texture is also determined by the average density of spots used for synthesis. This can be expressed as the average number of spots which cover an arbitrary point in the texture. This quantity determines the standard deviation of the intensities found in the texture.

5.1.2 Animated textures

Spot noise can be used to generate animations by considering the spots as particles in the flow field. The random spot positions are advected by the flow field to determine new positions in the next frame. In a stationary flow cyclic display of textures can be used to give an impression of the flow. To prevent the animation to jump between the first and last frame, the time dimension is considered cyclic. This means that a spot position advected in the last frame is stored in the first frame. Apart from a random position and intensity, each spot is also assigned a random *centre frame* number. The spot has the assigned random position and random intensity at that frame. The position of the spot in the frames following the centre frame are calculated by advecting the position of the spot forward in time with a certain time step. This is done for half of the number of frames the animation contains. If the last frame is reached the position is recorded in the first frame. For the frames before the centre frame the position is determined similarly with the difference that the position is integrated backward in time. The intensity of the spot is modulated using a multiplication factor depending on the distance from the centre frame (see Figure 5.4). At the centre frame the spot reaches its maximum intensity determined by the assigned random intensity. Away from the centre frame the intensity declines linearly to zero. From experience it was found that eight frames of texture are sufficient for a smooth closed loop animation in stationary flow.

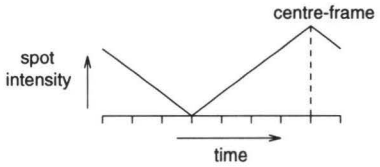


Figure 5.4: The intensity variation of a spot during animation

5.2 Spot bending

In standard spot noise the spots are rotated and scaled to reflect the local flow. First the spot is aligned with the flow direction by a rotation, then it is scaled. In the direction of the flow it is scaled proportional to $1 + |\mathbf{u}|$ and perpendicular to the flow by $1/(1 + |\mathbf{u}|)$, where $|\mathbf{u}|$ is the velocity magnitude. This scaling ensures that the total area of the spot and thus the average spot density in the texture remains constant. A zero velocity $|\mathbf{u}| = 0$ results in a round spot.

The spot influences a region in the texture, while its shape is based on data at a single point. If the velocity varies strongly over this region, the shape of the spot does not reflect the data properly. The result is a blurred texture (see Figure 5.6). The solution of using smaller spots would only partially solve the problem, for there are always cases in which the velocity gradient is too high to be well represented by a spot of a certain size. Also the characteristics of the texture change if smaller spots are used. The dominant frequencies in the texture increase if a smaller spot is chosen. This is often undesirable; if the spot becomes too small, in the order of the size of a pixel, the texture degenerates to white noise.

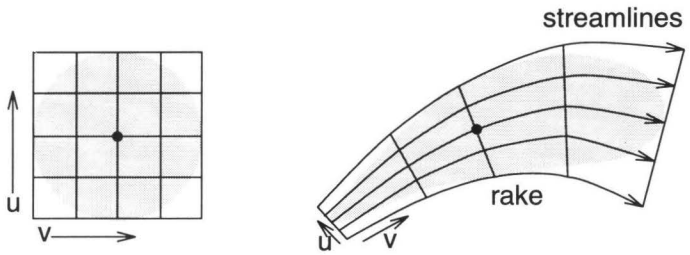


Figure 5.5: Spot bending left the parameterization of the spot and right the mapping of the spot to a parameterized stream surface generated around the spot position

To achieve a better representation of the flow field, we can apply a deformation to the spots. Cabral and Leedom suggested using a stream line to warp the major axis of an elliptical spot [Cabral & Leedom 1993]. This results in a correct representation of the

flow along a stream line. However, because the spot is 2D, not only effects of curvature have to be taken into account, but also effects of convergence and divergence of the flow.

For this deformation of the spot we used a two dimensional stream surface (see Figure 5.5). The same definition as for a three dimensional stream surface is used: a line in the flow is advected, thus tracing out a surface. The stream surface is constructed by integration of streamlines from a line segment perpendicular to the flow at the centre of the spot. The parameterization of this stream surface using u and v is used to deform the spot polygon that is parameterized in the same way. The effect of this method of deformation is that the spot is 'shaped' by the flow. The problem now is to keep the spot area constant regardless of the flow field; all spots should cover an equal area.

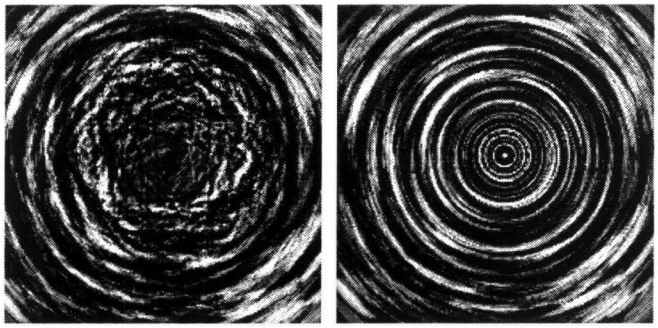


Figure 5.6: Standard spots (left) and bent spots (right) compared

The stream surface is generated as follows: the velocity $\mathbf{u}$ at the centre of the spot is determined. Next, a line segment (rake) is positioned perpendicular to the velocity direction. The length of the rake l_r is defined by

$$l_r = w\frac{1}{1+|\mathbf{u}|}, \tag{5.4}$$

where $|\mathbf{u}|$ is the velocity magnitude at the centre of the spot, and w a factor that determines the size of the spot with respect to the texture. The length of the rake is equal to the width of a standard spot, inversely scaled by the velocity magnitude. From the rake, streamlines are traced forward and backward at regular intervals. While the width of the spots is reduced according to the velocity (Equation 5.4), the length of the spot l_s is increased, such that the total area of the spot is constant:

$$l_r l_s = w^2. \tag{5.5}$$

Hence the total time interval T for which the stream surface is traced is

$$T = w\frac{1+|\mathbf{u}|}{|\mathbf{u}|}. \tag{5.6}$$

Note that this is an approximation, since we assume here that the velocity magnitude does not change over the surface of the spot. However due to the conservation of mass the spot area will not change in incompressible flows. For example, if the velocity decreases along the stream line, the length of the spot will decrease but it will also get wider, and thus the area will not change. However in areas where large density gradients occur, variations of spot size are possible. The stream lines are traced forward and backward for an equal time interval: $T/2$. The mesh of triangles resulting from the stream surface generation is rendered using the spot as a texture.

The difference between standard spots and bent spots is shown in Figure 5.6. The data set is a slice from a functionally generated vortex, with velocity inversely proportional to distance from the centre. The set was produced using an algorithm based on Wejchert and Haumann [Wejchert & Haumann 1991]. If standard spots are used, the centre of the vortex looks cluttered and the velocity cannot be deduced from the texture. To generate the picture with bent spots a rake with 8 streamlines was used, each integrated for 16 time steps. This gives a good result even close to the vortex core (see Figure 5.6).

Synthesis of bent spots is expensive, because streamlines have to be traced and a triangle mesh must be rendered for each spot. For performance reasons we implemented an option of conditional spot bending. The difference between a bent spot and a standard spot is determined by the change of velocity over the surface of the spot. In regions where the velocity changes are small, the difference between a standard spot and a bent spot is also small. We use the difference vectors between the velocity vector in the centre of the spot and those at the four corners to decide when a bent spot or a standard spot should be used. Bent spots are rendered only if the magnitude of the largest difference vector is larger than one tenth of the length of the spot. In experiments this value proved to result in a good balance between image quality and rendering speed.

5.3 Spot filtering

Spot noise pictures often have a coarse, low-frequency component. This component is not related to the shape of the spot, but only to its size. Irrespective whether we use circular or square spots, this low frequency component remains. If we can remove it, changing the shape of the spot will have more effect, and details in the texture will stand out more clearly. Because the power spectrum of a single spot and the resulting texture are equal (see 5.1.1), removal of the low frequencies from a spot results in textures without the undesired low frequencies. To remove low frequencies from a spot it is pre-filtered with a high-pass filter.

Theoretically the ideal high-pass filter is described by the equation

$$f(\mathbf{x}) = \delta(\mathbf{x}) - c\,\mathrm{sinc}(c|\mathbf{x}|) \tag{5.7}$$

where c determines the cut-off frequency and $\delta(\mathbf{x})$ represents the impulse function. The filtered signal is obtained by convolution of the original signal with the filter. Because

the sinc function has an unlimited extent, the filter has to be approximated in practice. As a high-pass filter $f(\mathbf{x})$ we used a negative Gaussian with an impulse at the origin (see Figure 5.7):

$$f(\mathbf{x}) = \delta(\mathbf{x}) - \frac{c}{\pi} e^{-c|\mathbf{x}|^2}, \tag{5.8}$$

where c is a parameter that controls the width of the Gaussian and thus the cut-off frequency of the filter. The filter function is discretized over an area of finite size. The value of the pixel at the centre of the filter, the $\delta(\mathbf{x})$ part of the equation, is chosen such that the average value over the discretized filter is zero.

A Gaussian filter is a suitable choice, that effectively reduces low-pass components. However, experiments showed that the choice is not very critical, other filters which approximate equation 5.7 can also be applied. The result of filtering a single spot is shown in Figure 5.7. The unfiltered disc-shaped spot is shown at the top right. The filtered version of the same spot is shown at the bottom.

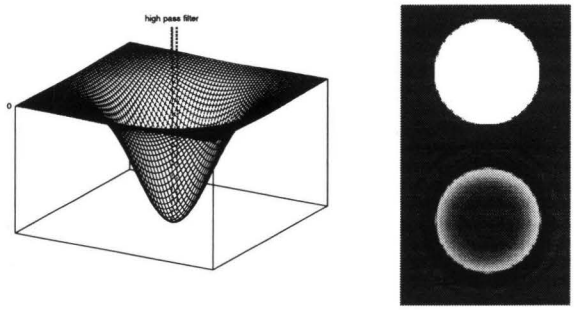


Figure 5.7: Spot filtering: the shape of the filter function (left), a unfiltered spot (top right) and the result of filtering this spot (bottom right)

The result of using filtered spots is shown in Figure 5.8. In this figure a vertical slice from a data set of a simulated storm is visualized using spots with a triangular shape. In the left image unfiltered spots are used. The low-frequency components can be clearly seen as large dark and light areas. In the right image a Gaussian filter is used. Here the texture is more homogeneous and shows the vector field accurately. Note also that the texture shows the magnitude of the flow. In the centre area in the lower part, the textures are fine-grained, while in the vortices at the left and right edge the texture is coarser. The use of a triangle as the basic spot shape results in a texture in which also lines *perpendicular* to the flow are present.

5.4 Use of graphics hardware

Visualization is an interactive process, therefore fast generation of pictures is essential. We used the graphics hardware available on current high performance graphics workstations such as the SGI Onyx to speed up both the texture synthesis and the rendering.

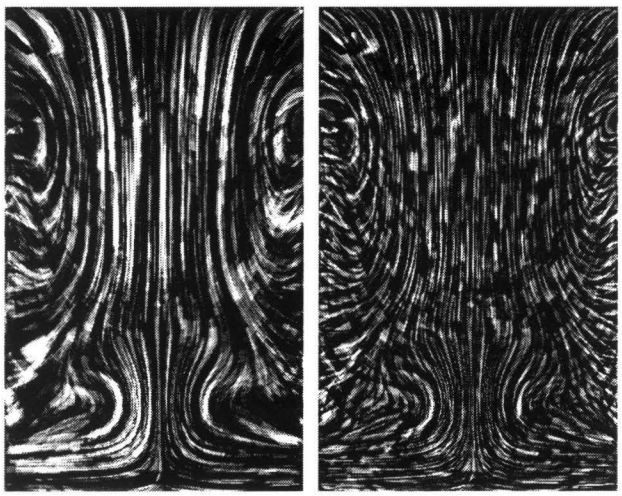


Figure 5.8: Data set visualized using unfiltered (left) and filtered (right) spots

In a preprocessing step we generate the textures; next these textures are mapped onto polygons for rendering.

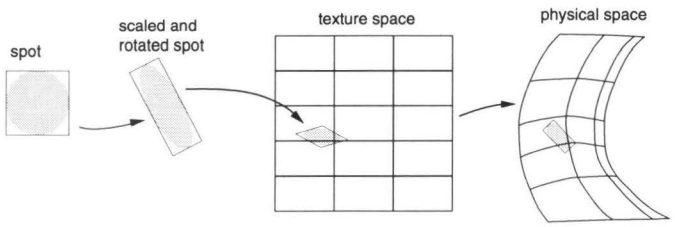


Figure 5.9: Steps in the synthesis of spot noise

The synthesis and inspection of spot noise is done in a sequence of steps, depicted in Figure 5.9. First the spot is transformed to reflect the data; then it is blended with the rest of the spots to generate the texture; finally, this texture is mapped onto the surface.

A typical spot noise texture consists of tens of thousands of spots. The transformation and blending of these spots takes a considerable amount of computation time. However, the nature of spot noise allows us to take advantage of the polygon rendering and texture mapping hardware. We represent the spot as a single texture [Max, Crawfis & Grant 1994]. This texture is mapped onto a unit polygon centred at the origin. To obtain a final spot in texture space, this polygon is scaled and rotated as a result of adapting the shape

of the spot to the vector value. Finally, the spot is translated to its position in texture space and blended. This series of transformations is achieved by using the matrix stack of the graphics hardware normally used for the viewing pipeline.

If bent spots are used, scaling and rotation transformations are not needed, because we generate the stream surface in texture space. The points on the generated stream surface can be used as positions for rendering the mesh which represents the spot. This mesh is rendered with the texture of the spot. The time and stream line number of a particular point are used to determine the corresponding location in the spot texture.

The variation in intensity of the spot is achieved by using the possibility of weighted blending of the rendered object with the picture generated so far. The hardware compositing only supports addition of intensities, but spots can also have a negative intensity, thus subtraction of intensity is needed. Solving this problem without losing the advantage of the hardware is done by splitting equation 5.3:

$$\begin{aligned} f(\mathbf{x}) &= \sum a_i h(\mathbf{x}-\mathbf{x}_i) \\ &= \sum b_i h(\mathbf{x}-\mathbf{x}_i) - \sum c_i h(\mathbf{x}-\mathbf{x}_i) \\ b_i &= \begin{cases} a_i & \text{if } a_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \\ c_i &= \begin{cases} 0 & \text{if } a_i \geq 0 \\ -a_i & \text{otherwise} \end{cases} \end{aligned} \tag{5.9}$$

This way positive-valued spots and negative-valued spots can be accumulated separately by the hardware and only a single subtraction per pixel is needed. Two separate textures are generated, one for positive spots and one for the negative spots. The two resulting textures are then subtracted in software to produce the final texture.

After synthesis of the textures, an interactive rendering module displays them in three dimensions. The textures are mapped onto a model description of the surface. The model can be inspected by varying the viewpoint with the mouse. Animated texture is achieved by cyclic display of a number of frames with advected texture. As the mapping from texture to physical space is done on the fly by the hardware the viewpoint can be varied during the animation. Times to generate spot noise range from less than a second for a single texture of reasonable quality, to several minutes for a high-quality sequence of texture frames for animation.

5.5 Non-uniform grids

As stated in the previous section, spot noise is generated on a uniform grid, whereas data often is defined on a non-uniform curvilinear grid. Spot noise generated in texture space is transformed to physical space. To prevent distortion of the texture, and thus wrong visualization of the data, the spots have to be pre-distorted in texture space. The transformation of a cell in computational space to a cell in physical space is specified by the Jacobian matrix. By applying the inverse Jacobian matrix transformation to the spots before they are rendered, the spots will have the desired shape in physical space.

We used linear interpolation to calculate the position and data values between grid points. For this purpose each rectangular cell is split into two triangles. A value $p(u, v)$ with local cell coordinates (u, v) is calculated by:

$$p(u,v) = \begin{cases} (1-u-v)p_{i,j} + up_{i,j+1} + vp_{i+1,j} & \text{if } u+v \leq 1 \\ (u+v-1)p_{i+1,j+1} + (1-v)p_{i,j+1} + (1-u)p_{i+1,j} & \text{if } u+v > 1 \end{cases} \tag{5.10}$$

where $p_{i,j}$ are the values at the grid points. Also for the transformation between texture space and physical space we used the decomposition in triangles. Each triangle of the surface has its own transformation. Because many spots have to be rendered all these transformations are pre-calculated and stored. The performance penalty for using this transformation is very small because it can be appended to the series of transformations needed to map a unit spot to the final spot in texture space as described in the previous section.

The advantage of this method of interpolation is that processing is fast. The surface can be rendered by rendering flat triangles instead of bilinear patches which must be rendered if bilinear interpolation is used. Also the interpolation of values needs less computation compared to bilinear interpolation.

As a result of this triangular mapping from physical space to computational space, the velocity field transformed to texture space becomes discontinuous. This can be seen in Figure 5.10, which shows a constant field on a trapezium-shaped cell with spot noise rendered on it. In texture space there is a discontinuity along the diagonal of the square. The velocity in computational space at a grid point depends on the triangle bordering the point chosen.

Because each triangle has its own transformation, a problem occurs if we want to apply the pre-distortion to a standard spot if it covers more than one cell. Several parts of the spot should be transformed differently, depending on the cell the part lies in. This problem does not occur if bent spots are used. If a streamline crosses a cell boundary, velocity will also change such that the spot will have the desired shape in the final image. In Figure 5.10 bent spots were used.

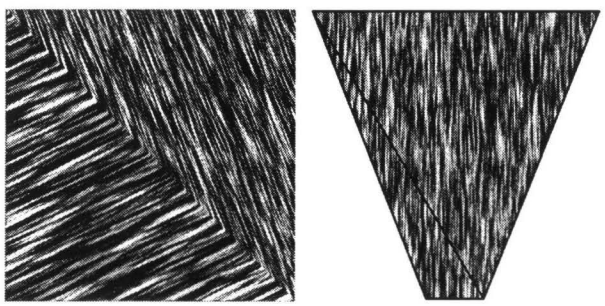


Figure 5.10: Texture space (left) and physical space (right) for spot noise. The field is constant and in the vertical direction

Many curvilinear grids are highly variable with respect to cell sizes. In regions where high velocity gradients are expected, such as boundary layers, the simulation is carried out at a higher resolution. If a uniform grid is used for texture synthesis, the pixel density of the generated texture is equal for each cell. If the texture is transformed back into physical space, in small cells many texels (texture elements) are squeezed into a small area, while texel-resolution is low in large cells.

Hence, it is more efficient to adapt the size of the texture cells to the size of the physical cells. We did this by using a rectilinear grid in texture space instead of a uniform texture space. The distance between two lines in the rectilinear texture space is chosen such that the area of the strip between the two lines has the same proportion to the total area as the equivalent strip in physical space (see Figure 5.11).

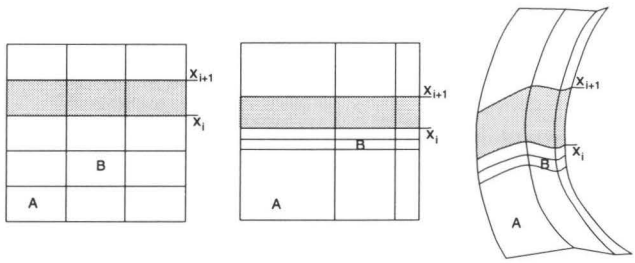


Figure 5.11: Less irregular scaling results if rectilinear texture space (middle) is used instead of uniform texture space (left) to represent the curvilinear physical space (right)

5.6 Spot noise and line integral convolution

Spot noise and line integral convolution (LIC) are both texture synthesis techniques which can be used for the visualization of vector fields. Although the definition is different, they are very similar as will be shown in this section.

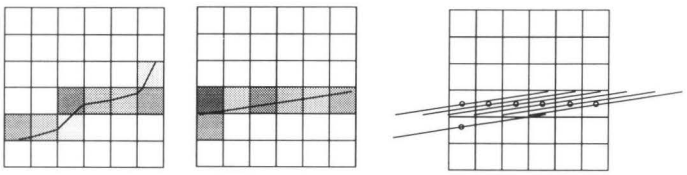


Figure 5.12: Input pixels used for DDA-convolution (left) and LIC (right)

LIC is generated by convolution of an input image with a one dimensional filter kernel (see Figure 5.12). So a pixel in the final image $C_{out}(\mathbf{x})$ is determined by an

integral along a line in the input image C_{in}:

$$C_{out}(\mathbf{x}) = \int_{s_0-L}^{s_0+L} k(s - s_0) C_{in}(\sigma(s)) ds. \tag{5.11}$$

In this equation $k(s)$ is the filter kernel and $\sigma(s)$ the line along which is integrated. In a vector field a stream line of a fixed length is used for this purpose. If we discretize this integral to be able to process it numerically we get:

$$C_{out}(i,j) = \sum_{p \in \tau} C_{in}(p) \cdot h(p) \tag{5.12}$$

where

τ	=	grid cells belonging to a line through (i,j)
$C_{in}(p)$	=	input texture pixel at grid cell p
$h(p)$	=	$\int_\alpha^\beta k(w)dw$ (the convolution integral)
α	=	the length of the streamline from the point (i,j) to where it enters cell p
β	=	the length of the streamline from the point (i,j) to where it exits cell p
$k(w)$	=	the convolution filter function

With the DDA convolution as described by Perlin [Perlin 1985] and by Cabral and Leedom [Cabral & Leedom 1993], the kernel is a straight line segment in the flow direction centred at the pixel position instead of a streamline. The set τ is the set of pixels making up this line when given in pixel format. If we assume a constant function as a filter kernel then the pixel value is calculated as follows:

$$C_{out}(i,j) = a \sum_{p \in \rho} C_{in}(i,j) \tag{5.13}$$

where a is a scaling factor.

In spot noise the colour (intensity) of a pixel is determined by the sum of random intensities of the spots which cover the pixel. Some simplifying assumptions are made to compare spot noise with LIC. If we assume a standard length (equal to the filter length in LIC) of the spot, and furthermore take a pixel size as the width of each spot and put one spot at each pixel, then spot noise can also be described by Equation 5.13 given for DDA-Convolution. In this case ρ is the collection of spots covering (i,j) and C is the intensity of these spots. If also the correlation between pixels values is taken into account then the equivalence also holds.

Thus, under the conditions stated above spot noise and LIC are equivalent. However, the similarity holds also in more general cases. The spot shape in spot noise is equivalent to the filter shape in LIC. Spot intensity scaling is equivalent to modulation of the pixel values in the input texture in LIC. And the bent spots are equivalent to using a stream line instead of a straight line segment to convolute along in LIC. The technique used in LIC to achieve animation could be realized by spot noise using temporal modulation of the spot shape keeping its position constant. The way it is achieved in spot noise by

advection of the spot positions could be emulated in LIC by advecting the pixel values in the input texture, which is achieved by the kernel shifting implemented by Stalling and Hege [Stalling & Hege 1995]. In conclusion, LIC can be regarded as spot noise where bent spots are used and where each spot has a fixed length and width, the width being one pixel.

In Figure 5.13 both methods are used to visualize the same data set: a slice from a simulation of flow in the bay of Gdansk. Both textures are generated on a 512×512 resolution. An obvious difference is the higher texture frequency in the image with the LIC-texture. In LIC this frequency is directly linked to the texture resolution because the algorithm works on pixels. Some vortices are more obvious in the LIC image. This is due to the representation of velocity magnitude by spot stretching in spot noise. The low velocity in some areas leads to nearly round spots and thus nearly isotropic texture from which direction is hard to deduce. It can, however, also be regarded as an advantage that spot noise also shows velocity magnitude.

LIC uses lines instead of surfaces as basic primitives and works directly in image space, therefore synthesis of the texture is theoretically faster. However because of the possibility to exploit the dedicated graphics hardware for the synthesis of spot noise, which is not so easy to achieve for LIC, in practice the differences in generation time are small. Furthermore it is easy to vary the balance between quality and speed with spot noise by variation of the spot density in the texture. For exact comparison of generation times a quality criterion is needed to let both algorithms produce a texture of equal quality. So far objective quality measures do not exist and therefore comparison can only be a subjective judgement.

5.7 Examples

In this section we present two examples of the use of enhanced spot noise in flow data sets. Further examples of the use of spot noise for the visualization can be found in the next chapter.

5.7.1 Stream surfaces

Stream surfaces are surfaces swept out by releasing a line in the flow field. A common way to generate stream surfaces is to release a number of particles along the line into the flow. By following each particle for a number of time steps a matrix of positions results. By tiling this curvilinear point grid with polygons the desired stream surface results.

Rendering of a stream surface with a single colour only gives partial information of the flow on the surface. It is a M_2^1 mapping. The magnitude and flow direction within the plane is lost. Spot noise generated for the curvilinear stream surface can be used to visualize this information. In this way a stream surface with spot noise representing velocity is a M_2^3 mapping. Because the velocity on a stream surface is in the direction of the surface, no projection of velocity to the surface is needed to generate spot noise for

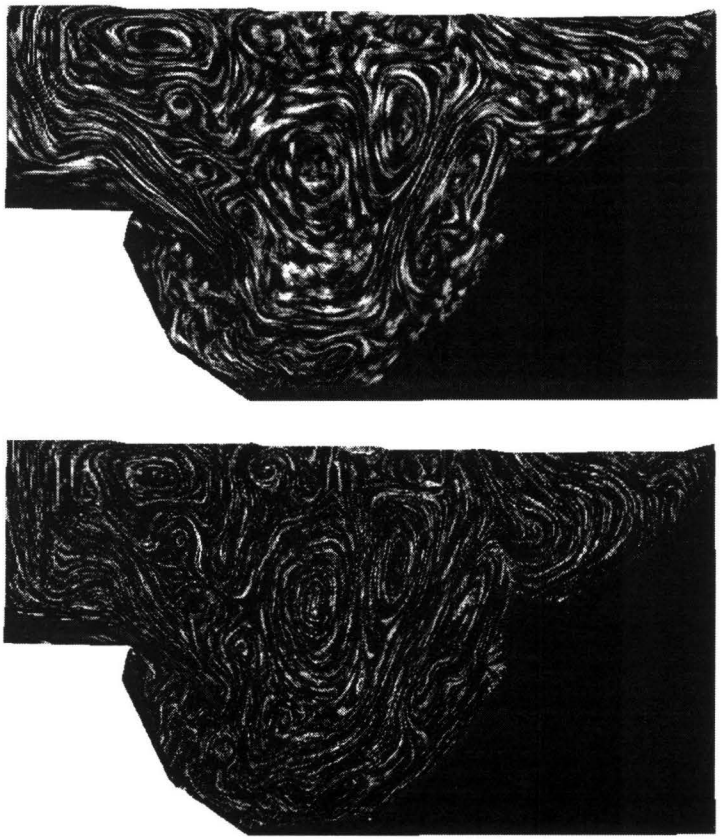


Figure 5.13: The bay of Gdansk data set visualized using spot noise (top) and LIC (bottom). Data courtesy: Delft Hydraulics

a three dimensional vector field. The result of mapping spot noise onto a stream surface is shown in Figure 5.14. In the storm data set a stream surface was generated using a vertical rake discretized using 50 points. The stream lines were traced using 48 time steps, resulting in a 50×48 curvilinear grid.

5.7.2 Backward facing step

In many data sets a large variation in velocity magnitude occurs. Spot noise can be used to give an qualitative impression of the velocity variations that occur in the data set. The surface used here is a slice through a data set of a backward facing step. The use of spot bending is essential here to show the main flow in which the velocity is high together

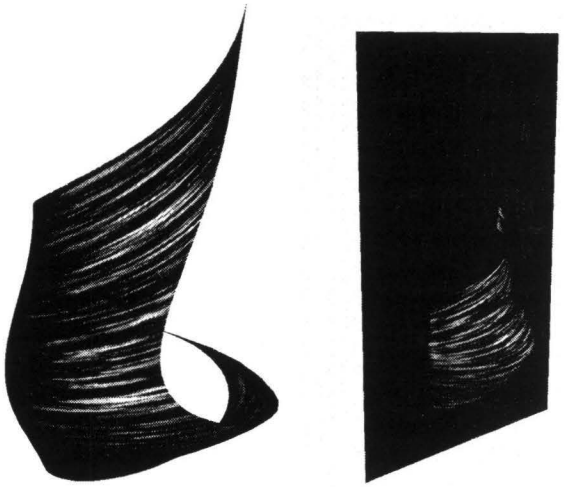


Figure 5.14: Spot noise on a stream surface in a data set of a storm. In the right image a slice through the set shows the location of the stream surface.

with the recirculation behind the step where the velocity magnitude is much lower.

Colour can be used to visualize an independent variable on the same surface. A colour map and spot noise are blended by using the value from the spot noise image as the intensity and the hue from the colour map as the hue in the blended picture. Flow simulations often produce other data along with velocity, such as pressure and temperature. In Figure 5.15, pressure is mapped to colour.

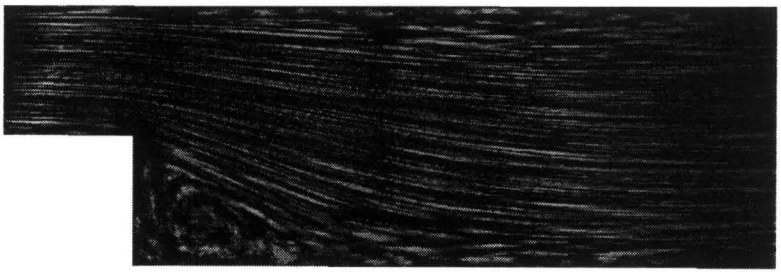


Figure 5.15: Slice of a backward facing step, colour showing pressure (see also Colour plate C.4 in the colour section). Data courtesy: Department of Numerical Mathematics, Delft University of Technology

5.8 Conclusions

In this chapter the synthesis of spot noise has been described. It was shown how spot noise can be used for the visualization of two dimensional vector fields which may be curved in three dimensions. Using spot bending, highly curved vector fields can be accurately visualized. Fast generation and interactive inspection of the textures is possible by exploiting the possibilities of graphics hardware.

Compared to other graphical primitives, texture has some advantages for the visualization of vector fields. It can give a continuous overview of a vector field in an intuitively clear way. There is no clear distinction with visualization using large number of small icons or glyphs [Littlefield 1983, van Wijk 1993]. Using many particles with random placement can provide similar result as texture [Stolk & van Wijk 1992, van Wijk 1993], however this is also expensive, and control of the quality of the final image is more difficult because it is not easy to guess in advance at which size the particles should be rendered or and at which density they should be rendered. Especially if the icons are advected in the flow, convergence and divergence of the flow will make an even distribution of the icons in the field a very difficult task to achieve. Using texture instead alleviates the user from such problems.

Texture is an extra dimension onto which data can be mapped. It can be used in combination with shape and colour. Visualization techniques based on the use of geometry or colour can be combined with spot noise. In the next chapter some further extensions and possible uses of spot noise will be described in more detail.

A drawback of texture is the lack of understanding of the perception of texture. Some efforts in this direction have been made for example by Rao and Lohse [Rao & Lohse 1993]. They identified three orthogonal dimensions in texture based on analysis of experiments in which subjects were asked to describe a number of textures with respect to 12 different rating scales. However much work in this field remains to be done.

If we compare spot noise to LIC, then there are some advantages and disadvantages. An advantage of spot noise over LIC is that the velocity magnitude is represented in the texture. LIC could be adapted to do this also by using a variable filter length. By linking the filter length to the velocity magnitude, magnitude information would be visible. Another advantage of spot noise is the use of the spot as a primitive. In vector fields this spot can be thought of as a particle advected in the flow. Using this metaphor animated textures are easy to generate. More data can be visualized by linking variables to shape parameters of the spot.

Some problems still deserve further investigation. An important problem is the visualization of time dependent vector fields. Using spot noise it is easy to produce textures representing a flow field at a certain time, however in animation the coherence between frames is important. Using the particle metaphor the spot should be advected along particle paths in the flow. To prevent variations in spot density in the texture due to convergence and divergence of the flow, a scheme where individual spots have a limited life time with an increasing and decreasing intensity must be used. Similar to the case of stationary flow, a scheme with random positions of spots which are advected forward

and backward in time while the intensity decreases further away from the initial position seems promising. The random position can be re-used after the particle linked to that position has disappeared, i.e. the intensity has become zero.

Although the use of graphics hardware speeds up the texture synthesis process it is still not possible to generate spot noise animations of a reasonable quality in *real time*. This means the texture is generated in a separate rendering stage. Speed is very important for the interactive inspection of vector fields. Using this capability, a slice could be moved interactively through a three dimensional data set while spot noise is mapped onto it, reflecting the vector data. Further research in the direction of using parallel computing is needed to achieve this rendering speed.

Chapter 6

Case studies with spot noise

In this chapter we will present two case studies on the use of spot noise. In both cases the potential of spot noise is further explored. Additional data along with the vector data are shown using the central idea of spot noise: generate a texture using spots which shapes are modified according to the underlying data.

In Chapter 2 it was stated that the user often has a specific question that should be answered by the visualization. In the first case described below the question was to find differences and similarities between a windtunnel experiment and a numerical simulation of the same flow case: a fin/wedge configuration. Spot noise is used to achieve a visualization of the numerical data comparable to the result of windtunnel experiments. This presentation allows a researcher to find similarities and differences between the two.

In the second section of this chapter visualization of turbulent flow using spot noise will be discussed. Turbulent flow simulations using a statistical model of turbulence have a turbulence intensity field along with an average velocity field as output. By carefully mapping these two fields to parameters of spot noise, an intuitively clear image can be generated, that shows the combined effects of these fields on the flow patterns.

6.1 Visualization of wall friction

The issue of comparative visualization has recently received attention (see [Pagendarm & Post 1995] for further references). Comparative visualization may be used to compare data from different sources, such as two simulation codes, which address similar physical phenomena or a comparison of experiment and numerical simulation. It may also be used to compare visualization methods. Comparison may be performed in various ways, such as data level comparison or image level comparison. Results of such comparisons may be presented side-by-side as well as integrated in a single image by various techniques. These techniques help the viewer to concentrate on differences in the data rather than needing to integrate the differences resulting from the visualization methods. Pagendarm and Post [Pagendarm & Post 1995] give an example of an experimentally acquired Schlieren image compared with a numerical flow simulation.

Pagendarm and Walter [Pagendarm & Walter 1994] demonstrated a number of visualization methods on a flow field around a fin/wedge configuration. They also used a photograph of a windtunnel experiment of a similar flow field to compare the data from the numerical solution with the experiment. The visualization of the numerical simulation did not come close to the experimental visualization technique. Given the visual similarity of the experimental visualization technique with spot noise, data from the same case was prepared at DLR in Göttingen to be visualized using the techniques described in the previous chapter. The resulting comparative visualization showed a high degree of similarity. This makes computer simulation images easy to interpret by aerodynamicists who are used to the old representations from their work at windtunnel facilities.

By adjusting the data processing steps and the visualization parameters the similarity may be improved. These variations can show the importance of the various factors. Some questions concerning the experimental visualization technique may as well be answered by specific tests with visualization of numerical simulations. This refers in particular to the interesting question whether the oil flow technique provides local information about the surface flow field at each point on the surface, or whether there exists an integrating effect in the streamwise direction.

6.1.1 The flow case

The flow field studied here is a simplified configuration of an air-intake of a hypersonic transport vehicle. Figure 6.1a shows a generic geometry of such an aircraft and its large air duct. The flow condition within the air duct which will contain the engines, will largely determine the performance of such vehicles. A particularly complex region may be found near the corners of the entry section which is expected to have a typical size of 26 m^2. A complex shock/boundary-layer interaction phenomenon occurs due to the great length of the forebody of the aircraft.

In order to study this complex phenomenon a simplified geometry defined by a blunt fin and a flat plate with a wedge is analysed by numerical flow simulation as well as windtunnel experiments. In the example given here the flow is simulated at a Mach number of 5. The numerical simulation of the compressible flow and the turbulent boundary layers was performed by T. Gerhold [Gerhold 1994].

Visualization of the numerical simulation of this flow field has been reported in [Pagendarm & Walter 1994]. The flow is characterized by several dominant features. Horseshoe vortices are formed in front of the fin and are bent downstream. Shock waves caused by the fin as well as the wedge intersect and interact with the vortices.

Windtunnel experiments were performed in the Ludwieg tube test facility (RWG) at DLR in Göttingen by P. Krogmann [Gerhold & Krogmann 1993]. Hypersonic flow conditions may be established in this test facility for a very short duration of 0.35 seconds. Visualization is the main result of such experiments. In this case an oil sublimation technique was used to visualize the surface flow field. Colour pigments are mixed with oil and painted onto the surface of a windtunnel model. During the windtunnel test the oil

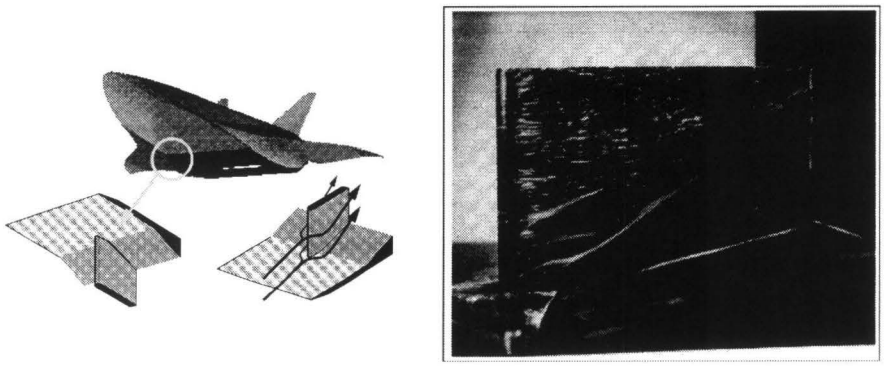


Figure 6.1: The flow case. (a) schematic illustration of the geometry near a corner of an air-intake of a hyper sonic transport aircraft (H.G. Pagendarm) (b) the experimental oil flow visualization (photo: P. Krogmann)

will moved over the surface and evaporate leaving behind the colour. Patterns are created because the first pigments which stick to the surface tend to accumulate more paint in a small wake behind them. These streaks visualize the direction of the flow close to the solid walls. Photographs or video recording are used to record the final pattern. A more detailed description of this experimental visualization technique was given by Merzkirch [Merzkirch 1987].

The technique is extremely useful to determine regions of separation or re-attachment which may be identified from converging or diverging streaks. Due to the fact that wall friction is usually much lower near separation the time needed to blow away or evaporate the oil becomes longer. More pigment finds the time to aggregate in the streak and thus the streaks are more clearly visible in these regions. For the case discussed here (Figure 6.1), the oil flow pattern was recorded on a photograph. The streaks clearly indicate the near wall flow direction. Separation is visible on the fin as well as on the plate in front of the wedge.

6.1.2 Wall friction

Since the numerical simulation is a solution of the Navier-Stokes equations, the flow velocity is zero at the wall. Therefore it is impossible to calculate streamlines on the walls in order to find out more about the surface flow field. A common way of solving this problem is by visualizing wall friction. The wall shear vector τ_w is the derivative normal to the wall of the velocity vector $\mathbf{v}$. In general it is non-zero and points in the direction of the near-wall velocity vectors when projected onto the wall. The wall shear τ_w may be calculated from the gradient of the velocity vector $\mathbf{v}$ after introducing a wall coordinate system where $\mathbf{n}$ is the coordinate vector of unit length normal to the wall.

$$\tau_w = \nabla\mathbf{v} - (\nabla\mathbf{v}.\mathbf{n})\mathbf{n} \tag{6.1}$$

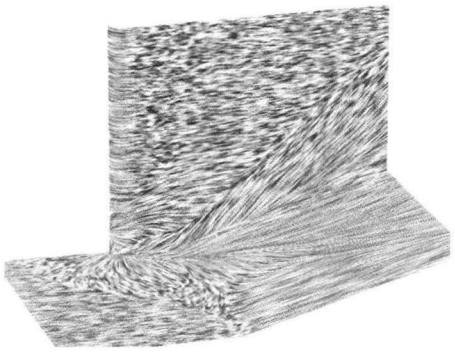


Figure 6.2: Spot noise generated from wall-friction-vector data

In a first feasibility study the wall-friction-vector data generated from the results of the numerical flow simulation at DLR were used to generate a spot noise pattern upon the fin/wedge geometry (see Figure 6.2).

While the texture of this first study looks promising, it became obvious that additional refinements were required in order to create images which would be as easy to interpret as oil flow patterns. Looking at the experimental oil flow visualization (see Figure 6.1a) reveals that oil streaks are brighter at separation areas. The physical mechanism behind this was explained earlier. Since these areas are characterized by convergent streamlines the divergence (or convergence) of the vector field might be a good candidate to be introduced as a scaling parameter for spot noise.

6.1.3 Control of the spot noise

To get a result which can be compared to the photograph of the experiment several modifications were made to the spot noise algorithm. The basic idea was to view the spots used to generate the texture as (white) oil on the (black) surface in the simulation, and thus to emulate the effects of the oil flow on the surface.

First, the intensity function was modified. In standard spot noise a uniform distribution with zero mean is used for the scaling value for the spot intensity. Here a uniform distribution of positive values was used for scaling the spot intensity. This improved the appearance of the texture as a black surface with white traces on it. A second step was the use of the convergence data. In standard spot noise, a constant maximum value $I_{\max}$ is used for the random range from which spot intensity I is chosen. Convergence is mapped to the maximum value of the random range that is used to determine spot intensity. We determine the maximum here by multiplying the maximum intensity by the convergence C at the spot position, mapped to the range of $0..1$. The intensity I of a

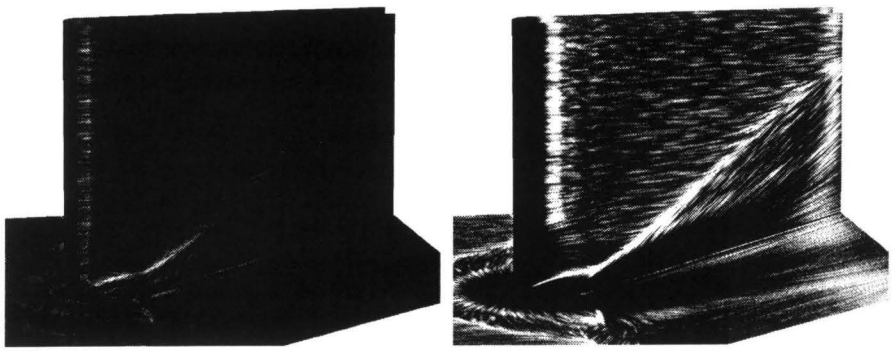


Figure 6.3: Adaptations to spot noise for the simulation of oil flow. (a) Convergence data used to scale the intensity distribution function (b) Spot advection used to simulate oil accumulation in high convergence areas

spot is thus determined by

$$I = r \cdot I_{\max} \cdot \frac{C - C_{\min}}{C_{\max} - C_{\min}}, \tag{6.2}$$

where r is a random value between 0 and 1 and $C_{\min}$ and $C_{\max}$ are the minimum and maximum values of convergence. The effects of these modifications are shown in Figure 6.3a. Although the result resembles the photograph more than the original, some differences are still obvious. These differences are caused by the fact that the technique does not take into account the accumulation of oil in regions of high convergence. To emulate this effect we introduced 'spot advection'. This means that the positions of the spots were not chosen completely random. Instead, the random positions which would normally be used as the locations to render the spots were advected by the wall friction vector field for a certain number of time steps. The resulting positions where used as positions for the spots. This is a similar technique as used for animated textures described in section 5.1. The difference is that all spot positions are generated at the first time step and only the positions in the last time step are used as input for spot noise synthesis. The result of this advection is a higher spot density in regions where oil converges (see Figure 6.3b).

Combination of the variable-intensity scaling and spot advection techniques leads to our final result shown in Figure 6.4. Here the spot positions are advected and rendered using intensity scaling based on the convergence value at the final position of the spot.

6.1.4 Comparison

The resulting visualization using the spot noise technique provides realistic images of high quality which are easy to comprehend and show a high degree of similarity with

sualization which shows both data fields and how they are related in an intuitively clear way. Different aspects of turbulent flow can be shown using different mappings. It will be shown that the use of spot noise is especially useful for the visualization of turbulent dispersion.

Several techniques have been developed for the visualization of turbulent flow. Briscolini and Santangelo visualized the vortex structures of two-dimensional turbulence using animation [Briscolini & Santangelo 1991]. They show the vortices of various sizes due to turbulence, but they need a complete description of the flow field (as generated by direct numerical simulations of turbulent flows) to make such pictures. The method cannot handle separate fields for turbulence and average velocity. Sakas and Westermann [Sakas & Westermann 1992] presented a method for the synthesis of turbulent phenomena using fractal functions, which are visualized using volume rendering.

In this section we will show that spot noise can also be used to visualize turbulent flow field data. Different aspects of turbulence such as dispersion and velocity variation can be mapped onto parameters of spot noise so that intuitively clear pictures are produced.

6.2.1 Turbulence

Although a direct simulation of turbulent flow using Navier-Stokes equations is possible in theory, in practice statistical models of turbulent flow are often used to allow analysis of large-scale flows. Statistical models are based on Reynolds decomposition, in which a turbulent flow is described by mean velocity and turbulence intensity components. A Reynolds-averaged turbulent flow simulation uses a statistical model, such as the k-ϵ model, where turbulence is characterized by kinetic energy k and dissipation rate ϵ. The resulting data from a Reynolds-averaged simulation can be mean velocity and eddy-diffusivity E. In general, E is a tensor quantity, but it is usually represented by a vector if the main directions align with the coordinate axes. For isotropic turbulence, E reduces to a scalar quantity.

Turbulent flow has many characteristics and effects, such as randomness and rotational motion. The detailed motion patterns, such as the rotational pattern of eddies, cannot be fully reconstructed from data generated by a statistical model, but the random nature of the motion can be visualized in a generic way. For example, the amount of fluctuation can be shown as a perturbation of particle motion.

In Hin and Post [Hin & Post 1993], particle motion is used to visualize both convective and turbulent motion by combining particle path integration of mean velocity data and random fluctuations derived from eddy-diffusivity data. This technique effectively shows local turbulence in 3D flow fields, but is limited by the number of particles in the flow. Ma and Smith [Ma & Smith 1993] have visualized turbulent dispersion using conical surfaces. The core of the conical surface is determined by a stream line of a particle advected by the average velocity while the radius of the cone increases downstream depending on turbulent dispersion. This increase of the radius is proportional to the local turbulence along the stream line. Hin [Hin 1994] also visualized turbulent dispersion by

animation of concentration fields, derived from particle motion. Both methods do not show turbulent motion itself, but only the dispersion of material caused by this motion. Turbulent particles [Hin & Post 1993] show this motion in a generalized way, but this technique is local by nature. We will use texture generation to give a global view of fluid motion, including turbulent effects.

6.2.2 Visualization of turbulent flow

The goal is a clear representation of the data, in our case the average velocity and scalar turbulence intensity data. In this section we will consider four different ways to achieve this. We will use an artificial data set of a jet in an almost still fluid, as described in detail by Hin [Hin 1994]. It was generated using profile functions for mean velocity and eddy-diffusivity on a regular grid, and we have used a vertical slice of 40×20 cells from the middle of the set for the spot noise visualizations.

Colour

Separate visualization of mean velocity and eddy-diffusivity is possible using for example spot noise for the mean velocity and colour for eddy-diffusivity (figure 6.6). This separate visualization gives a good view of the spatial distribution of turbulence intensity, but it does not show any turbulent motion, nor any effects of it, such as dispersion.

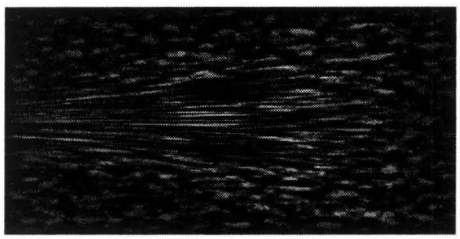


Figure 6.6: The plume data set visualized using spot noise for the average velocity and colour for the eddy-diffusivity (see also Colour plate C.5 in the colour section). Data courtesy:Andrea Hin, Delft University of Technology

Intensity range scaling

One parameter of spot noise that can be used to visualize an additional scalar value of a field is the intensity range of the spots. The intensity of a single spot is random, but changing the range from which the intensity of a spot is chosen affects the appearance of the texture. Here we scaled the intensity range by the turbulence intensity.

In images made using this method the range of random spot intensity increases with turbulence intensity, and thus the high-turbulence regions are highlighted by a higher

contrast. Attention is focussed on these regions. Low-turbulence regions with low contrast tend to recede into the background. In practice it appeared to be better to use a minimum intensity range: even where eddy-diffusivity is zero the spots still have some variation in intensity. This is to ensure that even in regions without turbulence the flow is still visible.

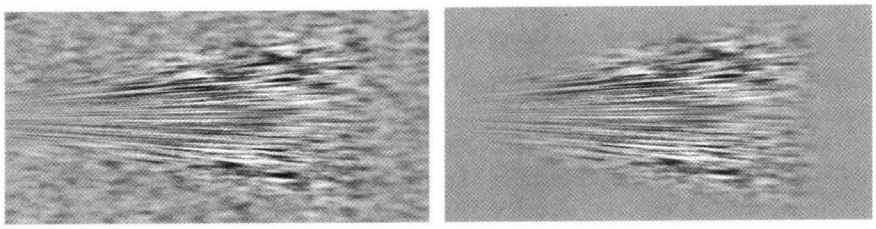


Figure 6.7: The plume data set visualized using spot noise with eddy-diffusivity mapped to the intensity of the spots. Left with, and right without a minimum intensity range for zero eddy-diffusivity.

Velocity perturbation

One effect of turbulence is fluctuation of velocity. The simulation has a mean velocity as a result, but the actual velocity at a point is unknown. However, by combining the *Fokker-Planck equation* and the *advection-diffusion equation* (Equation. 6.4) a velocity distribution function of possible velocities can be constructed [Hin & Post 1993]. Particle paths can be calculated with the following differential equation:

$$dX_\alpha = (\overline{u}_\alpha + \frac{\partial E_\alpha}{\partial \alpha} + \sqrt{2E_\alpha}N_\alpha)dt \tag{6.3}$$

where X is the position of a particle, α = the coordinate direction (x, y or z), $\overline{u}$ is the mean velocity, E is the eddy-diffusivity vector, and N is a trivariate normal distribution with zero mean and unit standard deviation. This equation can be used to generate velocity values with a realistic distribution. These values can be used as input to the transformation and advection of spots in the spot noise generation process.

In this way the velocity directions will vary strongly in areas of high turbulence intensity, giving an appearance of randomness. In low turbulence areas, the appearance of smooth motion is achieved by mean velocity with little directional perturbation.

In figure 6.8 two images are shown of the plume data set with different mappings of eddy-diffusivity to velocity magnitude. The most turbulent regions can be easily detected in both images. The higher frequency of the texture in regions with high turbulence is caused by the fact that the spot orientation becomes almost random, so the width of the spot determines the texture frequency in all directions. This effect is in accordance with the intuitive idea of chaos increasing with the amount of turbulence.

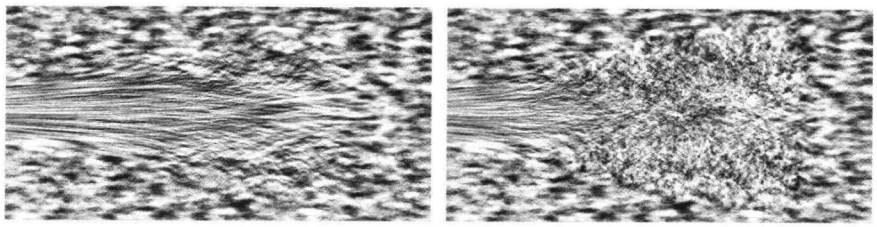


Figure 6.8: The plume data set visualized using velocity perturbation spot noise for average velocity and eddy-diffusivity data. In the picture on the right the maximum eddy-diffusivity is 4 times as high as on the left side.

Texture blurring

Another effect of turbulence is dispersion. The idea we used for this mapping is to regard the spots in the textures as the concentration of material inserted in the flow. The spot initially is disc-shaped with sharp edges, but due to turbulence the material is diffused and therefore the sharp edges become blurred proportional to the turbulence. This will show an effect of blurred texture in turbulent regions. This blurring effect is an intuitive representation of the uncertainty of the velocity information presented. We will now describe the physical foundation and implementation of this idea in more detail.

In fluid mechanics the fluid dispersion is modelled by the *advection-diffusion equation* given by:

$$\frac{\partial c}{\partial t} = -\nabla \cdot \overline{\mathbf{u}} c + \nabla(\mathbf{E}\nabla c) \tag{6.4}$$

In this equation c denotes concentration of inserted material, $\overline{\mathbf{u}}$ is the average velocity and $\mathbf{E}$ is the eddy-diffusivity. If we view the spot function as a local concentration function of the inserted material, then we can use the above equation to modify the shape of the spot over time. A full solution of equation 6.4 would give the local dispersion pattern of the spot, but would be very time consuming. To limit the calculation time we make some simplifying assumptions: $\overline{\mathbf{u}}$ and E are assumed constant over the surface of a spot. The justification of this is that spots are small with respect to the whole texture. If we apply these simplifications to equation 6.4 it reduces to:

$$\frac{\partial c}{\partial t} = \mathbf{E}\nabla^2 c \tag{6.5}$$

in discrete form for a 2D position (x, y) and a time step Δt this equation can be rewritten as:

$$C_{x,y,t+\Delta t} = C_{x,y,t} + (E_x \frac{\Delta^2 C_x}{\Delta x^2} + E_y \frac{\Delta^2 C_y}{\Delta y^2})\Delta t. \tag{6.6}$$

Assuming that $E = E_x = E_y$ (isotropic eddy-diffusivity) and $\Delta y = \Delta x$ this becomes:

$$C_{x,y,t+\Delta t} = C_{x,y,t} + \frac{E\Delta t}{\Delta x^2}(C_{x+\Delta x,y,t} + C_{x-\Delta x,y,t} + C_{x,y+\Delta y,t} + C_{x,y-\Delta y,t} - 4C_{x,y,t}) \tag{6.7}$$

This equation can be implemented by adding to the original spot image an image of the spot processed using the digital filter below:

0	1	0
1	-4	1
0	1	0

scaled by the value of $E\Delta t$.

Care has to be taken of numerical stability of this integration process. If the time step Δt is chosen too large, i.e. the value of $E\Delta t$ has a dynamic range which is too large, numerical instability will occur. The result of using a too large time step for the integration can be seen in figure 6.10. If filtering with a large time step is desired this can be achieved by filtering n times, with a time step equal to $\Delta t/n$. In the pictures presented here (except Figure 6.10) different amount of filtering was applied using a time step within the bounds of numerical stability, while changing the value of n. It can be proved using Von Neumann analysis [Roache 1985] that $E\Delta t$ should not exceed 0.25 in order to keep the process stable.

In figure 6.9 the result of applying this technique to the plume data set is shown. The difference between the images is a result of using a different time step used for the filtering operation.

Because spot noise is the accumulation of intensities of individual spots, this equation can be applied to each spot individually as well as to the whole texture. Because each pixel in the texture is covered by multiple spots, it is obvious that the latter is cheaper.

Visualization in a single image is possible by choosing a time interval to determine how long the spots are influenced by the turbulence. Another possibility is to use animation. Here all spots have the same size initially, and they are influenced by the flow (by advection) as well as the turbulence over time.

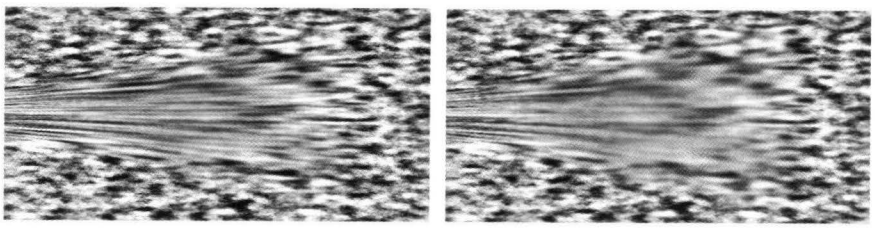


Figure 6.9: The plume data set visualized using spot noise with texture blurring.

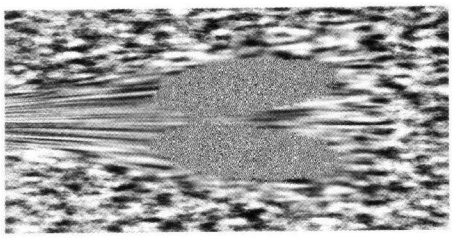


Figure 6.10: Texture filtering to visualize eddy-diffusivity: this image results from choosing a too large value of $E\Delta t$.

6.2.3 Examples

In this section we will describe two cases in which the techniques presented are applied to real-life data sets. Both data sets are slices from 3D layered hydrodynamic simulations [Mynett, Sadarjoen & Hin 1995].

In the process of mapping velocity to spot noise we can adjust many parameters. In the following cases the parameters showing the velocity are equal for each image and chosen such that mean velocity direction and magnitude is clearly visualized over the entire field. Bent spots and rectilinear texture space (see Chapter 5.2 and 5.5) are used for better texture and image quality. The size of the textures is 512×512.

Each of the data sets is visualized using the four methods described in section 6.2.2. The results are shown in figures 6.11 and 6.12.

The *Lith harbour* (figure 6.11) is a 3D simulation of a flow in a river passing a harbour entrance [Mynett, Sadarjoen & Hin 1995]. The river is shown at right in the figure, and the main flow direction is upward in the images. The harbour is at the top left, and is partly separated from the river by a narrow dam. No texture was mapped onto the land regions at the lower left. The middle horizontal grid slice was used here, with a size of 79×38 cells. The goal of the simulation was to study the material exchange between the river and the harbour, and to design the harbour entrance geometry to minimize the deposition of river sediment in the harbour.

All images clearly show the large difference in flow speed between the river and the harbour. Turbulence intensity is highest at the transition, with two peaks at each side of the harbour entrance. The speed difference complicates the perception of turbulence levels for all methods except the colour mapping. Discrimination between velocity variations and turbulence (both occurring at the harbour entrance) is difficult, because both turbulence and velocity variations affect material transport, and thus can be visualized in the same way.

The *bay of Gdansk* (figure 6.12) is a simulation of the flow in a coastal region in the north of Poland driven by wind and the inflow of the river Vistula. This results in a complicated flow pattern with a few small regions of high turbulence. The set size is 42×27 cells.

Because of the small size of the turbulent area, intensity range scaling of the spots to

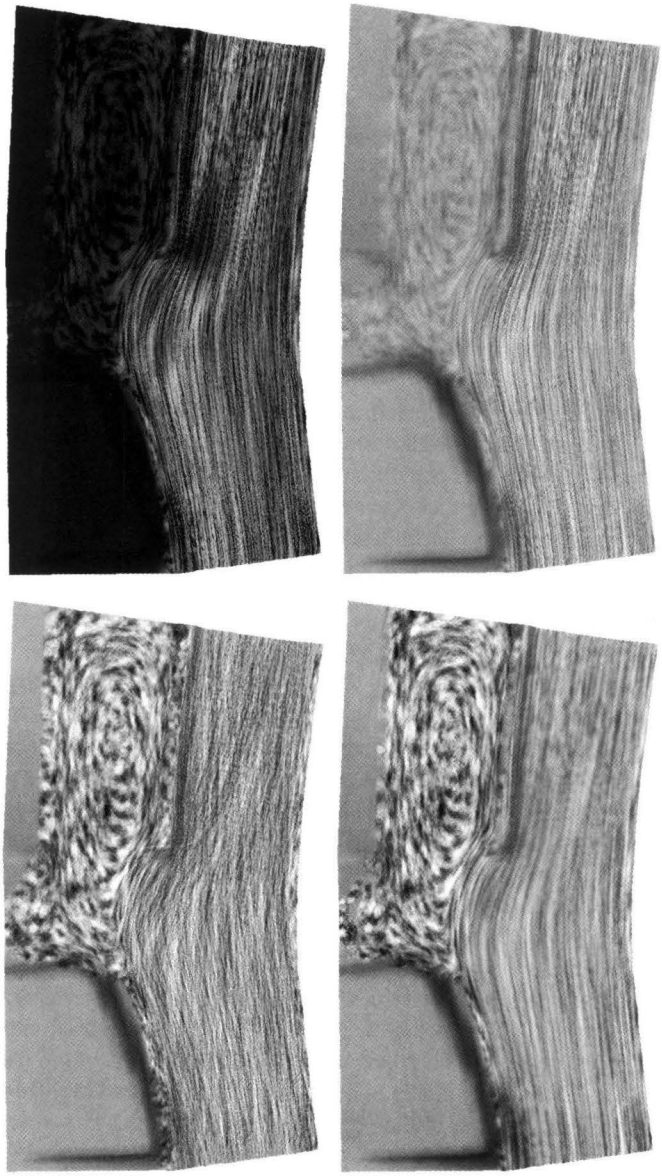


Figure 6.11: Slice from the Lith harbour data set visualized with different methods: colour (see also Colour plate C.6 in the colour section), spot intensity scaling, velocity perturbation and texture blurring. Data courtesy: Delft Hydraulics.

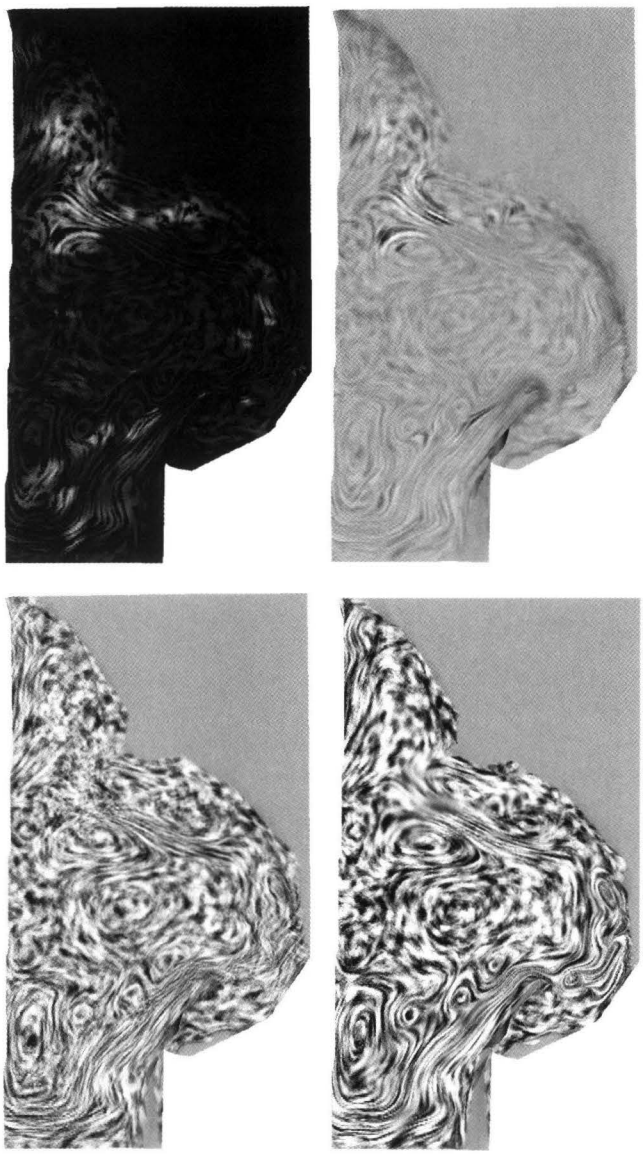


Figure 6.12: Slice from the bay of Gdansk data set visualized with different methods: colour (see also Colour plate C.6 in the colour section) , spot intensity scaling, velocity perturbation and texture blurring.

Chapter 7

Conclusions and future research

In the preceding chapters we have given a short overview of visualization followed by an in depth description of three new visualization techniques. In this chapter the conclusions will be drawn and some directions for future research will be given.

7.1 Conclusions

Visualization of flow is an interactive process in which exploration and presentation are keywords. As stated in Chapter 2 the visualization of three dimensional flow is a challenging problem because of the complexity of the data. The domain is three dimensional and the range three or more dimensional depending whether quantities such as temperature and pressure are also taken into account by the simulation. Presentation of all data in a single image has not been realized so far and may perhaps never be achieved. Instead, the visualization of flow can be realized by using a collection of techniques which reduce the dimensionality of the domain or range of the data. Data is inevitably lost in this process. An answer to a specific question can be found by using techniques which reduce the data dimensions in ways suitable to the specific question combined with the possibility of interaction such that exploration is possible.

The techniques presented in this thesis are all examples of different ways to reduce the dimension of the data. The probe shows data in a very small region of the flow, reducing the domain of the visualized data. Using statistical visualization aggregate data on a region is shown. Spot noise presents a two dimensional slice of data. Each of the techniques provides ways for interaction: the probe can be freely moved in the domain; statistics can be calculated on a user specified region; and different mappings of data can be chosen for spot noise. In the remainder of this section we will briefly return to the conclusions about the three techniques described in Chapters 3, 4 and 5. After that general conclusions will be drawn.

- **Local flow probe**

  The local flow probe is used for the interactive inspection of a small region in the flow. We have shown how the probe is constructed by transforming the gradient

tensor to a local coordinate frame, decomposing it into components, and mapping the separate parts onto geometric primitives. The probe is by no means the only possible representation of a tensor, but the special decomposition and mapping used makes it useful in flow analysis. The probe can be used as a tool to study details of a flow. It is useful as a complement to global visualization techniques.

- **Statistical visualization**

  In Chapter 4 we described a method for the visualization of vector fields using statistical techniques. By projection of the velocity values in an arbitrary region to a point, a 3D distribution function results. Using statistical techniques, characteristics of this distribution function are calculated. This results in efficient data reduction and a uniform presentation irrespective of the size and shape of the region. Various methods based on statistical distribution parameters such as average and variance for the visualization of the distribution function are presented. Average and variance/covariance can be depicted by an arrow and an ellipsoid. Two possible applications of the statistical method were given: The flux probe, which can be used to measure the flux through a polygon in the flow area, and the vortex finder which is useful for the interactive determination of vortices in the flow. By representation of a distribution function of velocity values by characteristics such as mean and standard deviation information is lost. In cases where the distribution deviates strongly from a normal distribution these characteristics give a false impression of the data. A test to decide if the distribution function resembles a normal distribution would be useful.

- **Spot noise**

  Spot noise synthesis was described in Chapter 5. It was shown how spot noise can be used for the visualization of two dimensional vector fields possibly curved in three dimensions. Using spot bending, highly curved vector fields can be accurately visualized. Fast generation and interactive inspection of the textures is possible by exploiting the graphics hardware. Compared to other graphical primitives, texture has some advantages for the visualization of vector fields. It can give a continuous overview of a vector field in an intuitively clear way. Texture is an extra dimension, it can be combined with shape and colour in a single image. This extra dimension can be very useful for the visualization of multivariate data. An important advantage of spot noise in flow visualization is the use of the spot as a primitive. First the spot offers possibilities for interaction: the user can influence the appearance of the textures by specification of a single spot. Second, a spot can be thought of as a particle advected in the flow. Using this metaphor, animated textures can be easily synthesised. However some problems still deserve further investigation. Two important problems are the visualization of time dependent vector fields and the generation of spot noise at interactive rates.

There are many visualization techniques for the visualization of flow. Which visualization technique should be applied in a certain case depends on a number of factors:

- The domain and range dimensions of the data to be visualized. As was described in Chapter 2, each visualization technique is capable of showing data of a certain number of domain and range dimensions. The data to be visualized and the visualization technique should match in this respect.

- The question to be answered. For example: is a global overview of the data needed or is the user interested in a certain detail of the data. Different techniques are available to answer different questions.

- Background of the user. A visualization technique can be seen as a language to express data. A language which must be learned. A less suitable visualization technique which the user is familiar with might be more effective than a technique which is better according to other criteria, which the user never used before.

The essence of presentation is careful mapping of data, taking into account the factors above. The presentation aspect of the probes described in this thesis is the mapping used which translates abstract data to primitives that are easy to comprehend. Also, spot noise presents the velocity data in a way which can be easily understood. Presentation using texture offers the possibility to generate images which resemble images resulting from physical experiments.

A requirement for exploration is an application in which easy interaction is possible. This means among other things that the time needed to generate an image should be short. Due to their local nature the probes can be interactively moved through the flow and placed at arbitrary locations. Using the graphics hardware spot noise can be generated in reasonable time. By storing the sequences of texture, interactive exploration is possible. Comparative visualization is an other way to look at exploration. Using spot noise to imitate oil flow images it is possible to visually explore differences between a windtunnel experiment and a numerical simulation.

7.2 Future Work

The visualization techniques described in this thesis complement each other in the sense that spot noise gives a global impression of the data, the statistical probe gives information on an intermediate level and the tensor probe is useful for the detailed investigation of the data. So far the techniques described have all been implemented as separate programs. Thus can not be combined into the same visualization. Combination of the techniques is possible by incorporating them as modules in a visualization package. This would offer the user the possibility of switching quickly between the techniques or simultaneous use.

The presented techniques all reduce the dimensionality of the data before visualization. However, they are not a structured set of facilities. A framework in which the options for reduction are set out would give visualization research ideas for the development of new techniques and the possibility to compare different techniques. To a user

such a framework could be a tool to choose the technique which is most suitable to the problem at hand.

Although the visualization of flow was referred to as a challenging problem, flow data sets are not the most complex type in existence. Other types of simulations generate data which are far more complex. Some of the ideas behind the presented techniques could be generalized such that they can be applied to field data with higher dimensions. For example the concept of calculation of statistical properties (Chapter 4) can be applied to higher dimensions. Suitable mappings for presentation of such data still have to be found. Also texture probably can be used for the visualization of multivariate data. Research of perception of texture in relation to scientific visualization would be useful to fully exploit the possibilities of texture.

References

ADVANCED VISUAL SYSTEMS INC. 1992. *AVS User's Guide, Release 4.*

BATCHELOR, G.K. 1967. *An Introduction to Fluid Dynamics.* Cambridge: Cambridge University Press.

BECKER, B.G., D.A. LANE & N.L. MAX. 1995. Unsteady Flow Volumes. In *Proceedings Visualization '95*, ed. G.M. Nielson & D. Silver. Los Alamitos (CA): IEEE Computer Society Press pp. 329–335.

BRISCOLINI, M. & P. SANTANGELO. 1991. Animation of Computer Simulations of Two-Dimensional Turbulence and Three-Dimensional Flows. *IBM Journal of Research and Development* 35(1/2):119–138.

BRYSON, S. & C. LEVIT. 1991. The Virtual Windtunnel: An Environment for the Exploration of Three-dimensional Unsteady Flows. In *Proceedings Visualization '91.* Los Alamitos (CA): IEEE Computer Society Press pp. 17–24.

CABRAL, B. & L. LEEDOM. 1993. Imaging Vector Fields Using Line Integral Convolution. *Computer Graphics (SIGGRAPH '93 Proceedings)* 27(4):263–272.

DO CARMO, M.P. 1976. *Differential Geometry of Curves and Surfaces.* Englewood Cliffs: Prentice-Hall.

CHAMBERS, J.M., W.S. CLEVELAND, B. KLEINER & P.A. TUKEY. 1983. *Graphical methods for data analysis.* Belmont (CA): Wadsworth International Group.

CHERNOFF, H. 1973. The use of Faces to represent points in k-dimensional space graphically. *Journal of American Statistical Association* 68(342):366–368.

CHUI, C.K. 1992. *An Introduction to Wavelets.* Wavelet Analysis and its Applications Academic Press.

CRAWFIS, R.A. & N.L. MAX. 1993. Texture Splats for 3D Scalar and Vector Field Visualization. In *Proceedings Visualization '93*, ed. G.M. Nielson & R.D. Bergeron. Los Alamitos (CA): IEEE Computer Society Press pp. 261–265.

DAUBECHIES, I. 1992. *Ten Lectures on Wavelets.* Philadelphia, Pennsylvania: SIAM.

DELMARCELLE, T. & L. HESSELINK. 1993. Visualizing Second-Order Tensor Fields with Hyperstreamlines. *IEEE Computer Graphics and Applications* 13(4):25–33.

VAN DYKE, M. 1982. *An Album of Fluid Motion*. The Parabolic Press.

FOLEY, J.D., A. VAN DAM, S.K. FEINER & J.F. HUGHES. 1990. *Computer Graphics: Principles and Practice*. Reading (MA): Addison - Wesley.

FORSSELL, L. K. & S. D. COHEN. 1995. Using line integral convolution for flow visualization: curvilinear grids, variable-speed animation, and unsteady flows. *IEEE Transactions on Visualization and Computer Graphics* 1(2):133–141.

FRANKE, R. & G.M. NIELSON. 1980. Smooth interpolation of large sets of scattered data. *Int. Journal for Numerical Methods in Engineering* 15:1691–1704.

FRIEDHOFF, R.M. 1989. *Visualization: The Second Computer Revolution*. New York: Harry N. Abrams, inc.

FUNG, Y.C. 1965. *Foundations of Solid Mechanics*. Englewood Cliffs: Prentice-Hall.

GERHOLD, T. 1994. Numerische Simulation und Analyse der turbulenten Hyperschallströmung um einen stumpfen Fin mit Rampe. Technical Report DLR-FB 94-19 DLR Göttingen.

GERHOLD, T. & P. KROGMANN. 1993. Investigation of the hypersonic turbulent flow past a blunt fin/wedge configuration. In *5th Intern. Aerospace Plane and Hypersonic Technology Conference Munich, Germany, Nov 30 - Dec 3 1993*. AIAA paper no. 93-5026.

GIBSON, J.J. 1950. *The perception of the visual world*. Boston: Houghton Mifflin.

HABER, R.B. 1990. Visualization Techniques for Engineering Mechanics. *Computing Systems in Engineering* 1(1):37–50.

HABER, R.B. & D.A. MCNABB. 1990. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In *Visualization in Scientific Computing*, ed. G.M. Nielson, B.D. Shriver & L.J. Rosenblum. IEEE Computer Society Press pp. 74–92.

HAMMING, R.W. 1962. *Numerical Methods for Scientists and Engineers*. New York: McGraw-Hill.

HELMAN, J.L. & L. HESSELINK. 1991. Visualizing Vector Field Topology in Fluid Flows. *IEEE Computer Graphics and Applications* 11(3):36–46.

HIN, A.J.S. 1994. *Visualization of Turbulent Flow*. PhD thesis Delft University of Technology.

HIN, A.J.S. & F.H. POST. 1993. Visualization of Turbulent Flow with Particles. In *Proceedings Visualization '93*, ed. G.M. Nielson & R.D. Bergeron. Los Alamitos (CA): IEEE Computer Society Press pp. 46–52.

INSELBERG, A., G. GRINSTEIN, T. MIHALISIN & H. HINTERBERGER. 1994. Visualizing multidimensional (multivariate) data and relations. In *Visualization in Geographical Information Systems*, ed. H.M. Hernshaw & D.J. Unwin. Chichester: John Wiley & Sons.

JERN, M. & R.A. EARNSHAW. 1995. Interactive real-time visualization systems using a virtual reality paradigm. In *Visualization in Scientific Computing*, ed. M. Göbel, H. Müller & B.Urban. Wien: Springer-Verlag pp. 174–189.

KAISER, M.K. & D.R. PROFITT. 1989. Perceptual issues in Scientific Visualization. In *Three Dimensional Visualization and Display Technologies*. Bellinghame.

KENDALL, M.G. 1965. *A Course in Multivariate Analysis, Introduction*. Chas. Griffin.

KENWRIGHT, D.N. & D.A. LANE. 1995. Optimization of Time-Depenent Particle Tracing Using Tetrahedral Decomposition. In *Proceedings Visualization '95*, ed. G.M. Nielson & D. Silver. Los Alamitos (CA): IEEE Computer Society Press pp. 321–328.

DE LEEUW, W.C. & F.H. POST. 1995. A Statistical View on Vector Fields. In *Visualization in Scientific Computing*, ed. M. Göbel, H. Müller & B. Urban. Wien: Springer-Verlag pp. 53–62.

DE LEEUW, W.C., F.H. POST & R.W. VAATSTRA. 1996. Visualization of Turbulent Flow by Spot Noise. In *Visualization in Scientific Computing*. Wien: Springer-Verlag.

DE LEEUW, W.C., H.-G. PAGENDARM, F.H. POST & B. WALTER. 1995. Visual Simulation of Experimental Oil-Flow Visualization by Spot Noise Images from Numerical Flow Simulation. In *Visualization in Scientific Computing '95*, ed. R. Scateni, J.J. van Wijk & P. Zanarini. Wien: Springer-Verlag pp. 135–148.

DE LEEUW, W.C. & J.J. VAN WIJK. 1993. A probe for local flow field visualization. In *Proceedings Visualization '93*, ed. G.M. Nielson & R.D. Bergeron. Los Alamitos (CA): IEEE Computer Society Press pp. 39–45.

DE LEEUW, W.C. & J.J. VAN WIJK. 1995. Enhanced spot noise for vector field visualization. In *Proceedings Visualization '95*, ed. G.M. Nielson & D. Silver. Los Alamitos (CA): IEEE Computer Society Press pp. 233–239.

LITTLEFIELD, R.J. 1983. Using the Glyph Concept to Create User-Definable Display Formats. In *Proceedings of the Fourth Annual Conference an Exposition of the National Computer Graphics Association*. pp. –.

MA, K.-L. & P. SMITH. 1993. Cloud Tracing in Convection-Diffusion Systems. In *Proceedings Visualization '93*, ed. G.M. Nielson & R.D. Bergeron. Los Alamitos (CA): IEEE Computer Science Press pp. 253–260.

MARR, D. 1982. *Vision*. San Francisco: Freeman.

MAX, N.L., R.A. CRAWFIS & C. GRANT. 1994. Visualizing 3D Velocity Fields Near Contour Surfaces. In *Proceedings Visualization '94*, ed. R.D. Bergeron & A.E. Kaufman. Los Alamitos (CA): IEEE Computer Society Press pp. 248–255.

MERZKIRCH, W. 1987. *Flow Visualisation, second edition*. Academic Press.

MYNETT, A.E., I.A. SADARJOEN & A.J.S. HIN. 1995. Turbulent Flow Visualization in Computational and Experimental Hydraulics. In *Proceedings Visualization '95*, ed. G.M. Nielson & D. Silver. Los Alamitos (CA): IEEE Computer Society Press pp. 388–391.

NIELSON, G.M. 1987. Coordinate free scattered data interpolation. In *Topics in Multivariate Approximation*, ed. L.L. Schumaker, C.C. Chui, F. Utreras et al. New York: Academic Press pp. 175–184.

OVERMARS, M.H. 1992. *Forms Library, a graphical user interface toolkit for Silicon Graphics Workstations, version 2.0*. University of Utrecht.

PAGENDARM, H.-G. & B. WALTER. 1994. Feature detection from vector quantities in a numerically simulated hypersonic flow field in combination with experimental flow visualization. In *Proceedings Visualization '94*, ed. R.D. Bergeron & A.E. Kaufman. IEEE Computer Society Press pp. 117–123.

PAGENDARM, H.-G. & F.H. POST. 1995. Comparative visualization – approaches and examples. In *Visualization in Scientific Computing*, ed. M. Göbel, H. Müller & B. Urban. Wien: Springer-Verlag pp. 95–108.

PERLIN, K. 1985. An Image Synthesizer. *Computer Graphics (SIGGRAPH '85 Proceedings)* 19(3):287–296.

POST, F.H. & J.J. VAN WIJK. 1994. Visual Representation of Vector Fields: Recent Developments and Research Directions. In *Scientific Visualization: Advances and Challenges*, ed. L.J. Rosenblum et al. Academic press pp. 367–390.

POST, F.H. & T. VAN WALSUM. 1993. Fluid Flow Visualization. In *Focus on Scientific Visualization*, ed. H. Hagen, H. Müller & G.M. Nielson. Berlin: Springer Verlag pp. 1–40.

POST, F.J., T. VAN WALSUM & F.H. POST. 1995. Iconic Techniques for Feature Visualization. In *Proceedings Visualization '95*, ed. G.M. Nielson & D. Silver. Los Alamitos (CA): IEEE Computer Society Press pp. 288–295.

RAO, A.R. & G.L. LOHSE. 1993. Towards a Texture Naming System: Identifying Relevant Dimensions of Texture. In *Proceedings Visualization '93*, ed. G.M. Nielson & R.D. Bergeron. Los Alamitos (CA): IEEE Computer Society Press pp. 220–227.

RAPPOPORT, A., R. KOSLOFF & R. MAYER. 1992. Visualizing Wave Functions in Molecular Dynamics Using Polygon Encoding. In *Proceedings of the Third Eurographics Workshop on Visualization in Scientific Computing*, ed. P. Palamidese. Focus on Computer Graphics Springer-Verlag pp. 239–253.

RIBARSKY, W., E. AYERS, J. EBLE & S. MUKHERJEA. 1994. Glyphmaker: Creating customized visualizations of complex data. *IEEE Computer* 27(4):57–64.

ROACHE, P.J. 1985. *Computational Fluid Dynamics, Revised Edition*. Albuquerque (NM): Hermosa publishers.

SADARJOEN, I.A., T. VAN WALSUM, A.J.S. HIN & F.H. POST. 1996. Particle Tracing Algorithms for 3D Curvilinear Grids. In *Scientific Visualization: Overviews, Methodologies, and Techniques*, ed. G.M. Nielson, H. Müller & H. Hagen.

SAKAS, G. & R. WESTERMANN. 1992. A Functional Approach to the Visual Simulation of Gaseous Turbulence. *Computer Graphics Forum (proceedings Eurographics '92)* II(3):107–117.

SCHROEDER, W.J., C.R. VOLPE & W.E. LORENSEN. 1991. The Stream Polygon: A Technique for 3D Vector Field Visualization. In *Proceedings Visualization '91*, ed. G.M. Nielson & L.J. Rosenblum. Los Alamitos (CA): IEEE Computer Society Press pp. 126–132.

SCHUMANN, H., N. LÓPEZ DE CHÁVEZ & K.-U. GRAW. 1996. Visual representation of multiparameter data with spatial dependence. In *Proceedings of the 7th Eurograpics Workshop on Visualization in Scientific Computing, Prague, April 1996*. pp. 67–79.

SILICON GRAPHICS INC. 1992. *IRIS Explorer User's Manual*. Silicon Grahics Inc.

SILVER, D., N. ZABUSKY, V. FERNANDEZ & M. GAO. 1991. Ellipsoidal Quantification of Evolving Phenomena. In *Scientific Visualization of Physical Phenomena*. Springer Verlag pp. 573–588.

STALLING, D. & H.C. HEGE. 1995. Fast and Resolution Independent Line Integral Convolution. In *SIGGRAPH '95 Conference Proceedings*, ed. R. Cook. ACM SIGGRAPH pp. 249–256.

STOLK, J. & J.J. VAN WIJK. 1992. Surface Particles for 3D Flow Visualization. In *Advances in Scientific Visualization*, ed. F.H. Post & A.J.S. Hin. Springer pp. 119–130.

UPSON, C. ET AL. 1989. The Application Visualization system: A Computational Environment for Scientific Visualization. *IEEE Computer Graphics and Applications* 9(4):30–42.

VAN WALSUM, T. 1993*a*. Content-based grid node selection for vector field Visualization. Presented at the 4th Eurographics Workshop on Visualization in Scientific Computing, Abingdon, UK.

VAN WALSUM, T. 1993*b*. Content-based grid node selection for vector field Visualization. Technical Report TWI 93-35 Delft University of Technology.

VAN WALSUM, T. 1995. *Selective Visualization Techniques for Curvilinear Grids*. PhD thesis Delft University of Technology.

WEJCHERT, J. & D. HAUMANN. 1991. Animation Aerodynamics. *Computer Graphics (SIGGRAPH '91 Proceedings)* 25(4):19–22.

WIERSE, M. 1994. *Higher order upwind schemes on unstructured grids for the compressible Euler equations in timedependent geometries in 3D*. PhD thesis Freiburg University.

WIERSE, M., T. GESSNER & D. KRÖNER. 1995. Numerial Methods, Simulations and Visualization for Compressible Flows. Presented at the Vismath Workshop, Berlin, may 30 - june 2 1995.

VAN WIJK, J.J. 1991. Spot noise – Texture Synthesis for Data Visualization. *Computer Graphics (SIGGRAPH '91 Proceedings)* 25(4):263–272.

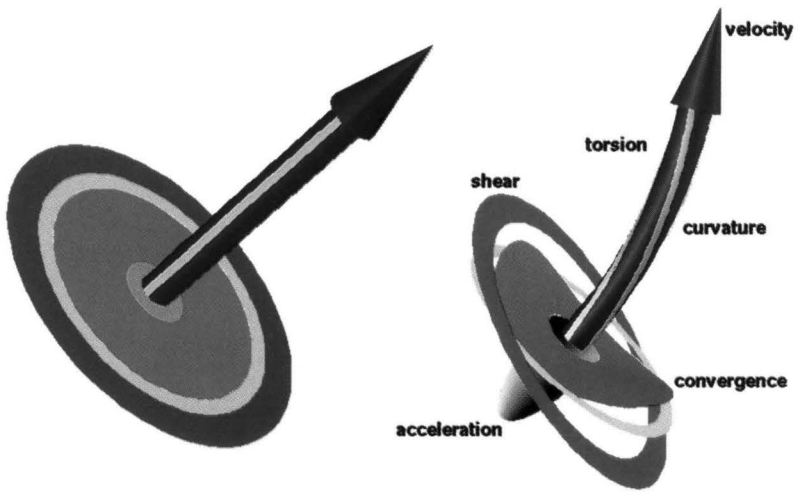
VAN WIJK, J.J. 1993. Flow Visualization with Surface Particles. *IEEE Computer Graphics and Applications* 13(4):18–24.

VAN WIJK, J.J., A.J.S. HIN, W.C. DE LEEUW & F.H. POST. 1994. Three Ways to Show 3D Fluid Flows. *IEEE Computer Graphics & Applications* 14(5):33–39.

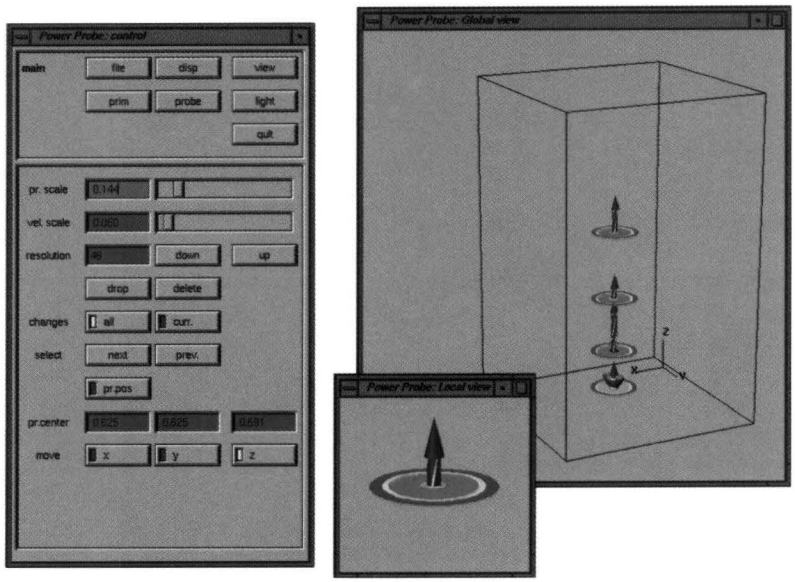
VAN WIJK, J.J. & R. VAN LIERE. 1993. HyperSlice – Visualization of Scalar Functions of Many Variables. In *Proceedings Visualization '93*, ed. G.M. Nielson & R.D. Bergeron. Los Alamitos (CA): IEEE Computer Society Press pp. 119–125.

VAN WIJK, J.J. & R. VAN LIERE. 1996. An environment for Computaional Steering. In *Scientific Visualization: Overviews, Methodologies, and Techniques*, ed. G.M. Nielson, H. Müller & H. Hagen.

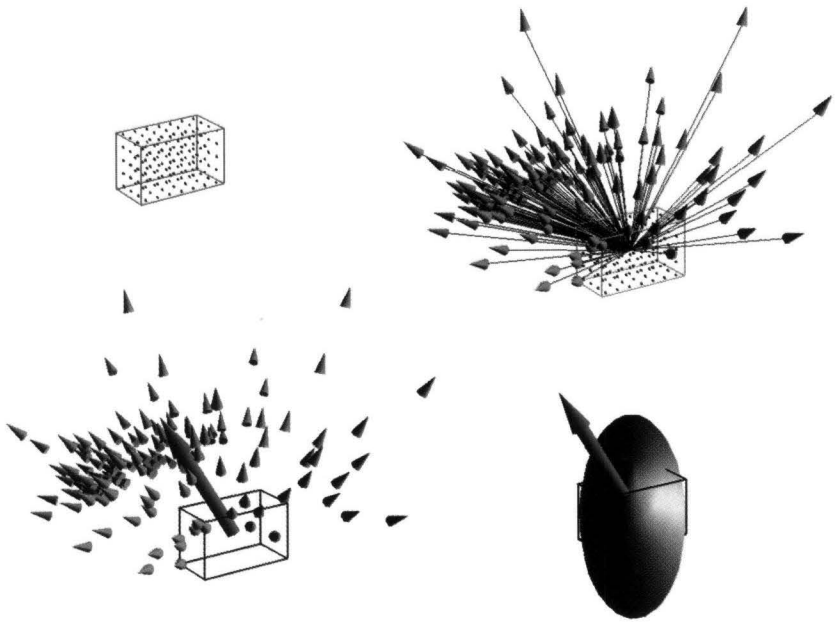
Colour section



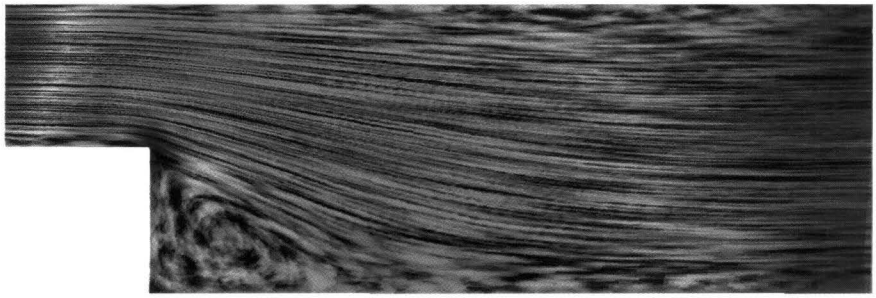
Color plate C.1: Components of the probe. Left: all gradient values are zero. Right all gradient values are non zero (see also Figure 3.3)



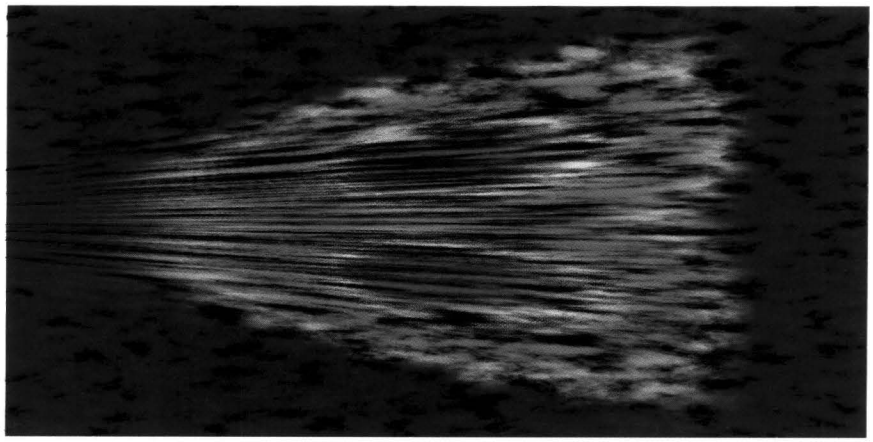
Color plate C.2: User interface of the application (see also Figure 3.7)



Color plate C.3: Construction of the statistical probe, showing the mapping to graphical primitives (see also Figure 4.2)



Color plate C.4: Slice of a backward facing step, colour showing pressure (see also Figure 5.15) Data courtesy: Department of Numerical Mathematics, Delft University of Technology



Color plate C.5: The plume data set visualized using spot noise for the average velocity and colour for the eddy-diffusivity (see also Figure 6.6)

Color plate C.6: Slice from the Lith harbour (left) and the bay of Gdansk (right) visualized using colour to show turbulence (see also Figures 6.11 and 6.12)

Summary

Presentation and Exploration of Flow Data

This thesis discusses techniques for the presentation and exploration of flow data. The complexity of three dimensional flow data does not allow direct graphical presentation of all data to the user. A way to achieve insight into flow data is by using a collection of interactive tools which show different aspects of the data. Three such techniques are presented: a local flow probe, statistical visualization, and spot noise.

The local flow probe is used to visualize a small region of the flow. Along with the velocity also the velocity gradients are visualized. The velocity gradients are described by a second order tensor. For visualization this tensor is decomposed into five components: acceleration, shear, torsion, convergence, and curvature. The resulting components are mapped onto easy to comprehend geometric shapes. The result is a geometric object which shows the characteristics of a small region in the flow. This object can be used as a probe for the interactive investigation of flow fields.

Statistical visualization is used to give a compact presentation of a user-selected region of a flow field. By applying statistics to the values occurring in the region the characteristics of the data are reduced to average and variance/covariance. We describe techniques for the selection of a region in the data, calculation of the statistical quantities and the presentation of these quantities. Two applications of the idea are shown. First, the flux probe which can be used to show the flux through a surface in the flow. Second, vortex cores can be found using the vortex probe.

Spot noise is a texture synthesis technique which can be used to visualize vector fields. Enhancements and extensions of this technique are described. First, better visualization of highly curved vector fields with spot noise is achieved, by adapting the shape of the spots to the local velocity field. Second, filtering of spots is proposed to eliminate undesired low frequency components from the spot noise texture. Third, methods are described to utilize graphics hardware for fast synthesis of the texture, and to produce variable viewpoint animations of spot noise on surfaces. And fourth, the effective synthesis of spot noise on grids with highly varying cell sizes is described.

Two case studies were performed using spot noise. In the first, spot noise was used to generate images from numerical data which resemble photographs taken during wind-tunnel experiments. This was done to enable easy comparison of the results of wind-tunnel experiments and numerical simulation of a flow case. In the second case the visualization of turbulent flow using spot noise was investigated. Different mappings

of the turbulence intensity data to parameters of the spot and the resulting texture were shown and compared.

Wim de Leeuw

Samenvatting

Presentatie en Exploratie van Stromingsgegevens

In dit proefschrift wordt een aantal nieuwe technieken voor de presentatie en exploratie van stromingsgegevens bescreven. Door de complexiteit van driedimensionale stromingsgegevens is het niet mogelijk alle gegevens direct aan de gebruiker te presenteren. Een manier om inzicht in stromingsgegevens te krijgen is het gebruik van een collectie van interactieve visualisatiegereedschappen die verschillende aspecten van de gegevens tonen. Drie van zulke gereedschappen worden beschreven: een stromingssonde, statistische visualisatie en spot noise.

De stromingssonde wordt gebruikt voor de visualisatie van een klein gebied in de stroming. Samen met de stroomsnelheid worden ook de snelheidsafgeleiden weergegeven. De snelheidsgradienten worden beschreven door een tweede orde tensor. Ten behoeve van de visualisatie wordt deze tensor ontbonden in vijf componenten: versnelling, afschuiving, torsie, convergentie en buiging. Deze componenten worden vertaald naar eenvoudig te begrijpen grafische objecten. Het resultaat is een geometrisch object dat de eigenschappen van een klein gebied in de stroming weergeeft. Dit object kan worden gebruikt als sonde voor interactief onderzoek van stromingsvelden.

Statistische visualisatie wordt gebruikt om een compacte presentatie van een gebied in een stroming te geven. Door toepassing van statistiek op de waarden van de in het gebied voorkomende gegevens worden de karakteristieken van het veld gereduceerd tot gemiddelde en variantie/covariantie. We beschrijven technieken voor de selectie van gebieden in de gegevensverzameling, berekening van statistische grootheden en de grafische presentatie van deze grootheden. Twee toepassingsmogelijkheden worden beschreven. Ten eerste de flux-sonde die gebruikt wordt voor het visualiseren van de flux door een oppervlak in de stroming. Ten tweede een methode om wervelkernen te vinden door gebruik te maken van de wervel-sonde.

Spot noise is een techniek voor de synthese van textuur die gebruikt kan worden voor de weergave van stromingsvelden. Verbeteringen en uitbreidingen van deze techniek worden beschreven. Ten eerste wordt de visualisatie van sterk gekromde velden door gebruik van spots die worden gedeformeerd op basis van de lokale stroming beschreven. Ten tweede beschrijven we het filteren van spots, wat een meer homogene textuur tot gevolg heeft. Ten derde wordt beschreven hoe grafische hardware gebruikt kan worden voor de snelle synthese van de textuur en voor de generatie van animaties van spot noise op oppervlakken. Ten vierde wordt de effectieve synthese van spot noise op roosters met

een sterk variërende celgrootte beschreven.

Een tweetal toepassingsgevallen waarin gebruik is gemaakt van spot noise wordt beschreven. In de eerste is spot noise gebruikt voor de generatie van beelden uit numerieke gegevens die overeenkomsten vertonen met foto's gemaakt tijdens windtunnelexperimenten. Dit is gedaan om de resultaten verkregen uit een windtunnelexperiment en numerieke simulatie eenvoudig te kunnen vergelijken. In het tweede geval is de visualisatie van turbulente stroming met spot noise onderzocht. Verschillende manieren om de turbulentiegegevens aan de parameters van de spot, en daarmee de resulterende textuur, te koppelen worden beschreven en vergeleken.

Wim de Leeuw

Curriculum Vitae

Wim de Leeuw werd op 11 maart 1967 geboren in Leerbroek. In 1985 behaalde hij zijn VWO-diploma aan de Christelijke Scholengemeenschap "Oude Hoven" te Gorinchem. In datzelfde jaar begon hij de studie Informatica aan de Technische Universiteit Delft. In 1992 studeerde hij af bij de leerstoel computer graphics, met als afstudeeronderwerp het onderzoek naar versnellingstechnieken voor ray tracing.

In augustus 1992 begon hij als onderzoeker in opleiding (OIO) bij dezelfde leerstoel aan zijn promotie-onderzoek op het gebied van wetenschappelijke visualisatie. Gedurende deze periode heeft hij tevens een bijdrage geleverd aan het door de vakgroep Technische Informatica verzorgde onderwijs.

Sinds november 1996 werkt hij als onderzoeker bij het Centrum voor Wiskunde en Informatica (CWI) te Amsterdam.