The Density Surfaces Module

(Version 1.0)

Arturas Mazeika, Michael Böhlen, Peer Mylov

December 2002

TR-3

A DB Technical Report

Title The Density Surfaces Module

(Version 1.0)

Copyright © 2002 Arturas Mazeika, Michael Böhlen, Peer Mylov. All rights reserved.

Author(s) Arturas Mazeika, Michael Böhlen, Peer Mylov

Publication History December 2002. A DB Technical Report.

Contents

Chapter 1

Introduction

This manual describes the design and implementation of the 3DVDM density surface (DS) module, which is part of the 3DVDM System. Density information is crucial statistical information that can be used for a number of purposes. This manual illustrates how density surfaces support visual data mining. The DS module offers a broad range of tools to compute and display density surfaces for 3D data.

The purpose of this manual is to describe the DS module and related software components such that you a) can install the required components and run the DS module, b) get to know the functionality of the DS module and learn how to use it, c) have the essential information to extend or modify the DS module, and d) have an example how density surfaces can be used for the purpose of visual data mining. Towards this end the manual is structured into three parts.

Part I describes the graphical user interface and is a general purpose introduction to the DS module. In the first subpart all elements of the GUI are described. In the second subpart we use a series of examples to illustrate the types of data analyzes the density surface module supports.

Part II describes the internals of the DS module. This part is relevant if you want to extend or modify the functionality of the DS module. The code consists of three largely independent blocks. The APDF class is the core part. It implements the density estimation and calculates density surfaces. The DSMapper class integrates the DS module into the 3DVDM System. We employ various code fragments illustrate the integration. Finally, there's an extensive toolbox to generate artificial data sets on the fly.

Part III describes a data mining case study. We discuss how density surfaces can be used to analyze clickstream data.

Once you have installed the 3DVDM System (cf. Chapter 2) we recommend that you read Part I to get a feel for the functionality of the DS module. Although not required it might be convenient to have the 3DVDM System available and try out the various tools. The next step depends on your goals. You should proceed to Part III if you are mostly a data analyst and want to analyze data. If you want to program and progress the functionality of the DS module you should proceed to Part II.

Chapter 2

Installation

2.1 Hardware and Software Requirements

There are no special hardware requirements to run the 3DVDM System. Essentially, the system should work on any hardware compatible with Linux/IRIX. Note though that the software is computation and graphical resource intensive. For the exploration of large datasets (containing several million observations) we recommend 512MB memory, a fast graphical card, and a state-of-the-art CPU.

All experiments presented in this manual were produced on a 1GHz Pentium III PC with 512MB of RAM and a Geforce2 GTS 220 graphical card. The computation of the probability density function for good visual results typically requires 1-3 seconds. The 3DVDM System is able to visualize up to 100'000 objects (displayed as tetrahedra) and still ensure a smooth interactive navigation.

The DS module is integrated into the 3DVDM System and was compiled and run on the following operating systems:

- SGI IRIX64 6.5
- RedHat Linux 7.1, 7.2, 7.3, 8.0
- Mandrake Linux 9.0
- Debian Linux 3.0

Currently, RedHat 7.2 is used as the main development platform for the DS module. Released versions have also been tested on SGI/IRIX, Mandrake and Debian Linux systems.

To install the 3DVDM System you basically need a standard Linux installation that includes support for the development libraries for `OpenGL`, `Glut`, and `GTK`. Most Linux stock installations include all the necessary components.

2.2 Download and Installation Instructions

Below we give a minimal description of the installation procedure for Redhat 7.2 that should also work for most other Linux installations. For more detailed information we refer to the installation instructions of the component packages [4, 5, 1, 2]. Here we only provide the URLs of the required software packages and list the commands to install the packages. Note that the version numbers will evolve and are likely to be different. Usually, you should be OK with the latest version of the respective software packages.

Downloading:

```
lam-6.5.8-sysv.1.i386.rpm
            from http://www.lam-mpi.org/download/files/
3dvdm-data.tar.gz
3dvdm-0.2.0-pre3.tar.gz
vr++_0.5.1.tar.gz
            from http://www.cs.auc.dk/3DVDM/software.html
vrjuggler-1.0.6.linux-rh72-gcc2.tar.gz
            from http://prdownloads.sourceforge.net/vrjuggler/
```

Unpacking:

```
$ tar xvfz soft-3dvdm-0.2.0-pre3.tar.gz
$ tar xvfz soft-vr++_0.5.1.tar.gz
$ tar xvfz soft-3dvdm-data.tar.gz
$ tar xvfz vrjuggler-1.0.6.linux-rh72-gcc2.tar.gz
```

Installation:

```
$ rpm -Uvh lam-6.5.8-sysv.1.i386.rpm
$ (cd vr++_0.5.1; make)
$ (cd 3dvdm-0.2.0-pre3; make)
```

The installation process creates directories with the required binaries. The final step before you can execute the binaries is to set an environment variable that points to the VR Juggler directory, e.g.,

```
export VJ_BASE_DIR=$HOME/3DVDM/vrjuggler-1.0.6.linux-rh72-gcc2
```

Several notes are in order:

- The installation of LAM/MPI requires root privileges.
- If you have a multi-processor machine you might want to use `lam-6.5.8-usysv.1.i386.rpm` instead.
- If you use gcc 3 you should download the corresponding version of VR Juggler.
- If you compile the 3DVDM System on SGI or if you have a non-standard setup you have to edit the user make files and adjust them as appropriate (cf. `user.makes/user.make.linux` for Linux and `user.makes/user.make.sgi` for SGI IRIX).

2.3 Running

In order to run the DS module, change the current directory to the binary directory of the 3DVDM system:

```
cd 3dvdm-0.2.0-pre3/bin
```

To start the DS module simply type

```
./ds
```

or

```
./ds <filename>
```

If the program is launched without the `<datafile>` option then the program will look for data files in the `3dvdm-data` sub-directory. All files with a .cvs extension qualify and will be available for selection. In the latter case `<filename>` must be the name of a comma separated data file.

Because the 3DVDM System is implemented as a multi-process system there will always be two open windows: the menu of the DS module and the main window with the density surfaces and/or the respective data. Thus, in contrast to other window applications the menu is never closed and menu selections are immediately propagated. This last feature requires some care in order to prevent the display of intermediate (and thus unwanted) graphs (cf. Chapter 3).

Part I

Using the Graphical User Interface to Analyze Data

This part is a general introduction to the functionality and working of the DS module. Reading this part is a good way to become familiar with the DS module and we recommend new users to start out here.

In Chapter 3 we use the graphical user interface to give an overview of the various parameters of the DS module. Basically the GUI shows the state the DS module is in and allows the user to modify this state. We explain the components of the GUI, discuss dependencies and individual parameters, and explain how to control GUI and data visualization in a multi-process environment.

Chapter 4 explains selected data explorations tools in more detail. As mentioned in the introduction density information is general purpose statistical information that can be used for a number of purposes. While we have chosen to use it to compute and display density surfaces this still leaves open how to use density surfaces to analyze data. This is a crucial parameter because the effective use of density surfaces has a major impact upon the data analysis process. We illustrate the following data analyzes methods:

- Animating density surfaces
- Displaying multiple independent density surfaces
- Displaying density surfaces for conditional data sets
- Equalizing density surfaces
- Density Surfaces for data windows ("smart" filtering and rescaling)

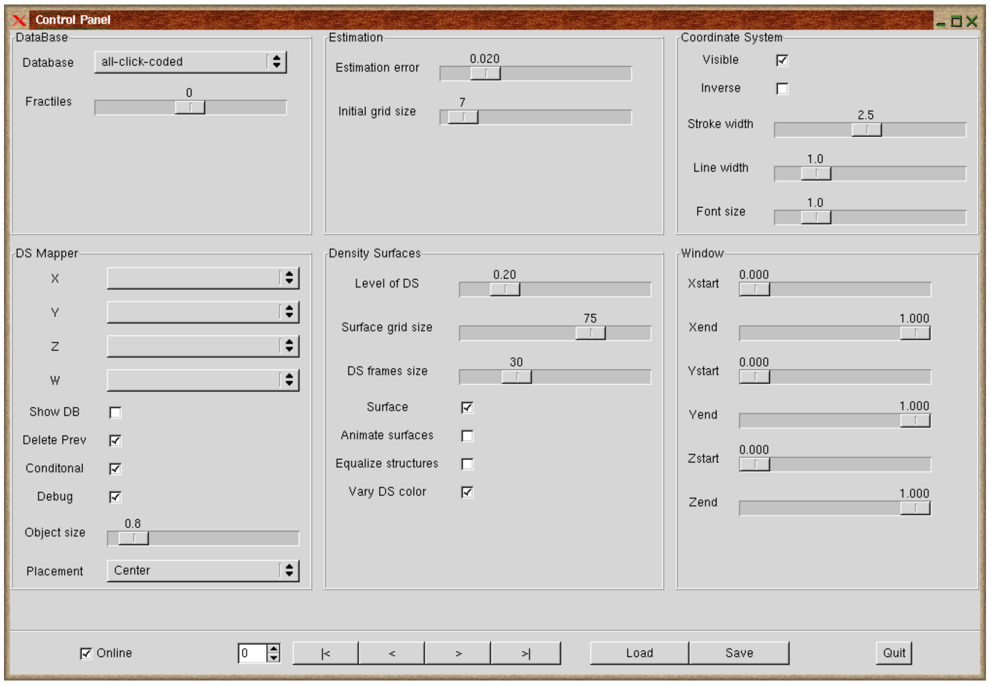
These methods illustrate the versatility of density surfaces and provide a good feel for the type of data analyzes that can be performed. We expect that these method will be progressed and refined in the future.

Chapter 3

The Graphical User Interface

3.1 Structure and Philosophy of the GUI

The graphical user interface (GUI) of the DS module consists of six groups: `DataBase`, `DS Mapper`, `Estimation`, `Density Surfaces`, `Coordinate System`, and `Window` (cf. Figure 3.1). The `DataBase` and `Coordinate System` groups are inherited from the abstract task class of VR++ and are documented in detail elsewhere.



DataBase
Database: all-click-coded
Fractiles: 0

Estimation
Estimation error: 0.020
Initial grid size: 7

Coordinate System
Visible ✓
Inverse ☐
Stroke width: 2.5
Line width: 1.0
Font size: 1.0

DS Mapper
X
Y
Z
W
Show DB ☐
Delete Prev ✓
Conditonal ✓
Debug ✓
Object size: 0.8
Placement: Center

Density Surfaces
Level of DS: 0.20
Surface grid size: 75
DS frames size: 30
Surface ✓
Animate surfaces ☐
Equalize structures ☐
Vary DS color ✓

Window
Xstart: 0.000
Xend: 1.000
Ystart: 0.000
Yend: 1.000
Zstart: 0.000
Zend: 1.000

Online ✓ 0

Figure 3.1: The GUI of the Density Surfaces Module

The `DataBase`, `DS Mapper`, and `Window` groups control the data that shall be visualized (data file, attributes, and ranges). The `Estimation` group controls the estimation error of the estimated probability density function (PDF). The `Density Surfaces` group controls the display of the density surfaces (quality, animation, density level, etc.).

Basically, the items in the GUI are organized according to the dependencies among the blocks of the DS module. For example the attributes (`DS Mapper` $\rightarrow$ `(X,Y,Z,W)`) can only be selected if the dataset (`DataBase` $\rightarrow$ `Database`) has been selected. Similarly, the estimation of the probability density function requires that attributes have been selected, and density surfaces can only be displayed after the density has been estimated (thus `Estimation` depends on `DS Mapper`, and `Density Surfaces` depends on `Estimation`).

The GUI represents the state of the DS module (cf. Part II for details about the internal representation of the state). Any change of a GUI element translates into a change of the state of the DS module. Any change triggers a possible recalculation of the density estimation, density surfaces, and actual visualization.

The parameters of the GUI has to be updated carefully to avoid the display of intermediate unwanted graphs. A simple and systematic approach is the following:

- The attribute `X` has to be unselected (as a result of this all rendering of graphs is disabled).
- Make all the required changes (e.g., change the estimation error, modify the data window, and toggle the display of conditional surfaces). None of the changes will have an effect on the display because the `X` attribute is not selected.
- The attribute `X` has to be selected again. This re-enables rendering.

Before we describe the different part of the GUI in detail we list and summarize all elements:

DataBase Group

- `Database`. The name of the input data file.

DS Mapper Group

- `X,Y,Z,W`. The DS module requires three spatial attributes (`X,Y,Z`) and a categorical attribute (`W`). After the `Database` attribute (cf. above) has been selected the pull down menus for `X,Y,Z,W` show all available attributes. Attribute `W` is not used if `Conditional` (cf. below) is not checked.
- `Show DB`. Toggles the display of a sample of the data.
- `Delete Prev`. If checked each new visualization deletes all previous visualizations. If not checked visualizations are additive.
- `Conditional`. Splits the data into conditional data sets according to the categorical attribute `W`. Each conditional data set is processed separately.
- `Debug`. Outputs debug information (computational time, memory consumption, etc.) to the standard output.
- `Object Size`. The size of the visualized tetrahedra.
- `Placement`. Controls the placement of the next visualized dataset.

Estimation Group

- `Estimation Error`. Controls the error of the density estimation. The computational time increases as the error decreases.
- `Initial grid size`. Controls the initial approximation of the PDF. The approximation is used to determine where to add additional grid points. It has to be as low as possible to be fast but large enough to not miss structures in the dataset.

Density Surface Group

- `Level of DS`. Density level of the density surface. This controls the density of the data points enclosed by the density surface.
- `Surface grid size`. Controls the visual quality of the displayed surface. A higher value yields a more detailed surface (more surface points are drawn on the screen). The computational time increases slightly as the value of the parameter increases.
- `Animated surface`. If checked the surfaces are animated automatically by varying the density level. The value of `DS frame size` determines the number of intermediate density surfaces.
- `Equalize structures`. Equalizes not equally pronounced structures. The equalization makes it possible to visually compare the structure of high and low density regions.
- `Vary DS color`. If checked the color of the density surfaces is varied.

Window Group

- `Xstart, Xend, Ystart, Yend, Zstart, Zend`. Defines the data window. All observations that are outside the cube are not considered.

3.2 The DataBase Group

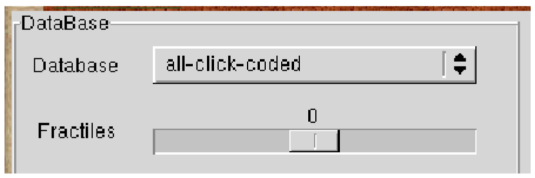

DataBase
Database all-click-coded
Fractiles 0

Figure 3.2: The DataBase Group

The `DataBase` group allows to select the data file that shall be used for the analysis. The system scans for comma separated files (`.csv`) in the `3dvdm-data` directory (cf. Section 2) and lists them in the `DataBase` drop-down menu. If the dataset is being selected for the first time, the system pre-calculates basic statistical information (for example min, max of the data) and prepares the data for fast later access (for a more detailed discussion consult [4]).

If the `Database` parameter is not selected the DS module turns into the waiting state. No events are handled until `Database` and the four attributes `X`, `Y`, `Z`, and `W` (cf. Section 3.3) are selected.

Changing `Database` triggers the recalculation of the `DS Mapper` group.

3.3 The DS Mapper Group

Once the `Database` (cf. Section 3.2) has been selected the `DS Mapper` group is updated with the names of attributes of the dataset.

If `Conditional` is not checked only the spatial coordinates `X,Y`, and `Z` are used. However, even in this case a (dummy) value for attribute `W` has to be selected. If `Conditional` is checked the data is divided

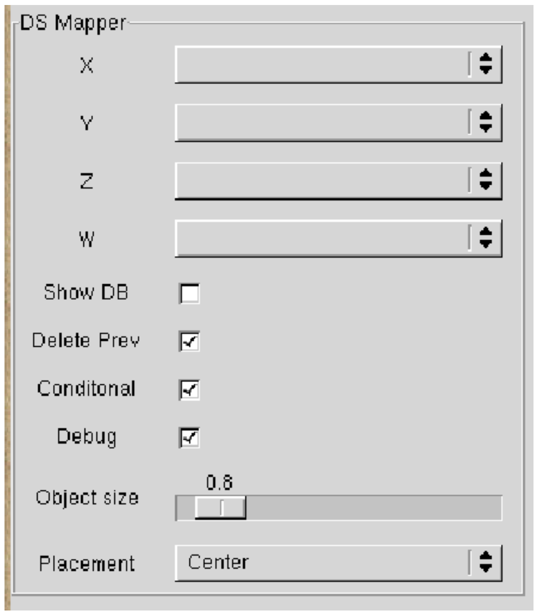

DS Mapper
X
Y
Z
W
Show DB ☐
Delete Prev ☑
Conditonal ☑
Debug ☑
Object size 0.6
Placement Center

Figure 3.3: The DS Mapper Group

into conditional datasets according to the categorical attribute `W`. Each conditional dataset is processed separately.

The DS module turns into the waiting state if one of the attributes `X`, `Y`, `Z`, or `W` is unselected. No events are handled until `Database`, `X`, `Y`, `Z`, and `W` have been selected.

A change in the `DS Mapper` group triggers the recalculation of the PDF (the `Estimation` group) and density surface (the `Density Surfaces` group).

3.4 The Window Group

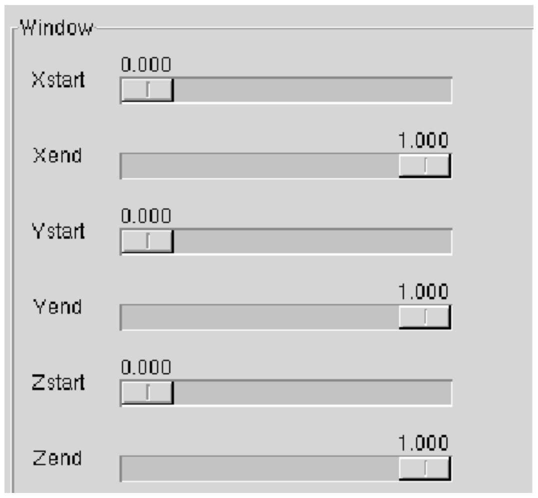

Window
Xstart 0.000
Xend 1.000
Ystart 0.000
Yend 1.000
Zstart 0.000
Zend 1.000

Figure 3.4: The Window Group

The window functionality makes it possible to filter all observations that do not fall into a user-defined data window. The entire data range varies between 0% and 100% and the sliders can be used to limit this range for each coordinate individually. The DS module scales the filtered observations to the unit cube and estimates the PDF .

Changing the data window triggers the recalculation of the PDF (the `Estimation` group) and density surface (the `Density Surfaces` group).

3.5 The Estimation Group

The `Estimation` group controls the error of estimated probability density function. Two parameters are used for this purpose.

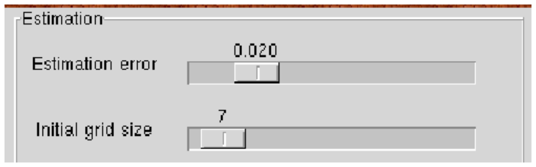


Figure 3.5: The Estimation Group

The `Estimation error` controls the quality of the estimation. The precision increases as the `Estimation Error` decreases. The default value (0.02) ensures a visually good quality of the PDF and a low computational time (1-2 seconds) for typical datasets. If the error is decreased to 0.01 the visual quality becomes very good.

The DS module estimates the PDF by processing a sequence of buffers with data points. After a buffer has been processed the estimation is updated with new grid points in areas where the PDF is non-linear. The `Initial grid size` parameter determines the initial number of grid points.

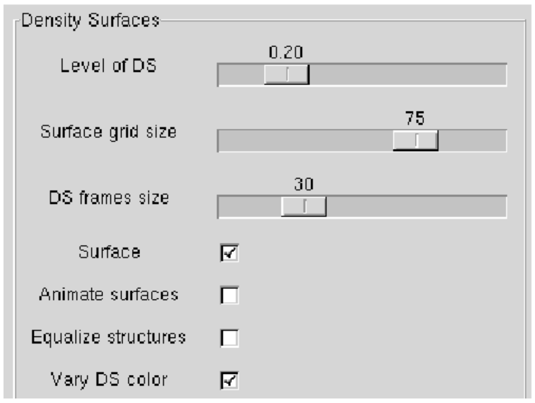
A low value for `Initial grid size` ensures a low computational time. However, a coarse initial grid can yield an inadequate estimation of the PDF. Some structures of the dataset can be "overlooked". Increasing `Initial grid size` improves the quality of the PDF but also increases the computational time fairly quickly.

Setting `Initial grid size` to 7 is a good choice for most of applications. For non-smooth surfaces the parameter has to be increased, though.

3.6 The Density Surface Group

If `Animated` is not checked then `Level of DS` determines the level of the density surface. A high value will only display surfaces for high-density data regions. If `Animate` is checked the density surfaces are animated. `DS frame size` determines the number of density surfaces between the largest and smallest surface.

The animation of surfaces supports a comprehensive analysis of the data because it investigates and displays regions at varying density levels (more and less pronounced structures). This is particularly useful for getting a quick overview of the data.


Density Surfaces

Level of DS 0.20

Surface grid size 75

DS frames size 30

Surface ✓

Animate surfaces ☐

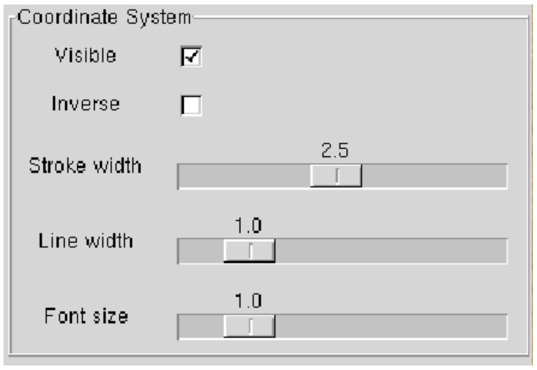
Equalize structures ☐

Vary DS color ✓

Figure 3.6: The Density Surface Group

3.7 The Coordinate System Group

The `Coordinate System` group allows to show/hide the coordinate system (`Visible`), invert the background (`Inverse`) change the thickness of grid lines (`Line width`), font size of the labels (`Font size`) and (`Stroke width`).


Coordinate System

Visible ✓

Inverse ☐

Stroke width 2.5

Line width 1.0

Font size 1.0

Figure 3.7: The Coordinate Group

The group is primarily designed to create figures that can be printed on paper.

Chapter 4

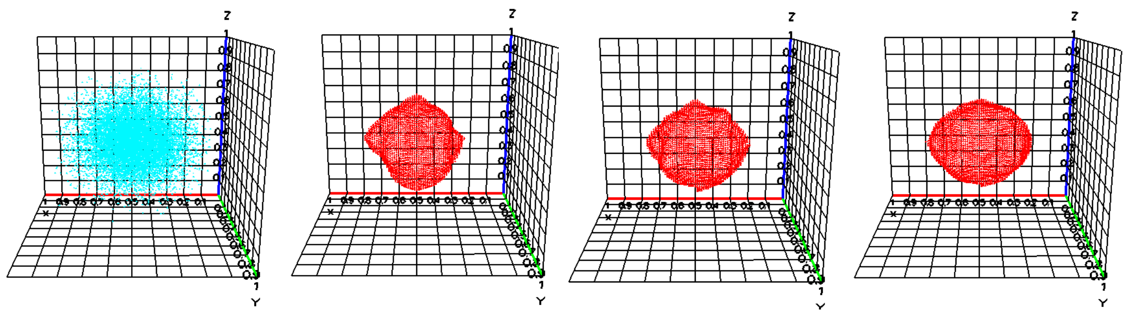
Sample Data Explorations

This section describes different uses of the DS module: animation of density surfaces, visualization of multiple and conditional datasets, equalization of structures, and the window functionality.

4.1 The First Session

In order to calculate a density surface four parameters have to be chosen: `Database` (the `DataBase` group) and `X`, `Y`, `Z`, `W` (the `DS Mapper` group). After these parameters have been selected the DS module computes and displays a density surface with default values for all other parameters. Below we discuss the effects of changing the default values of these parameters.

The `Estimation Error` (the `Estimation` group) controls the precision of the estimation of the probability density function of the dataset. A lower estimation error produces more accurate results. Figure 4.1 shows density surfaces for the ball dataset (a three dimensional normal random vector) for different estimation errors ε. $\varepsilon = 0.1$ produces a rough estimation for a very low memory consumption and computational time (cf. Table 4.1). As the estimation error ε decreases to 0.03 (0.01) we get good (very good) precision of the PDF. (Because of scaling some of the printed surfaces are deformed and resemble an ellipsoid rather than a sphere. This is not the case if you display them on a computer monitor.)



(a) The Scatter Plot (b) $\varepsilon = 0.1$ (c) $\varepsilon = 0.03$ (d) $\varepsilon = 0.01$

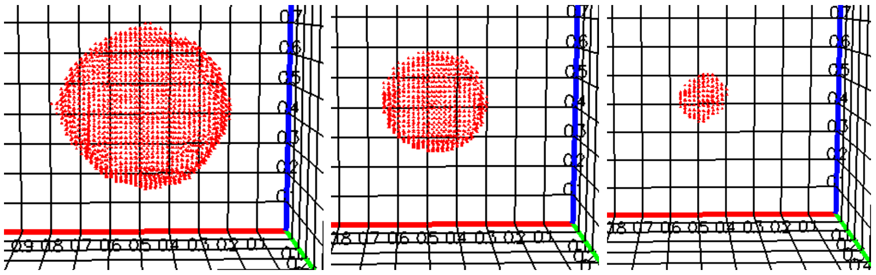
Figure 4.1: The Impact of Estimation Error to Density Surfaces

As the estimation error decreases the computational time and memory consumption increase. Note also that a higher precision also requires larger datasets (cf. Table 4.1). If the total number of observations n is too small to obtain the selected ε, the estimation error is increased to the smallest error that can be achieved with n observations.

ε	0.1	0.03	0.01	0.001
Memory Consumption (KB)	4	14	60	132
Computational Time (sec)	0.29	0.35	0.46	13.00
Number of required observations	224	1’746	10’705	>100’000

Table 4.1: Memory Consumption, Time, and Number of Required Observations for Different Estimation Errors ε

The `Level of DS` parameter (the `Density Surfaces` group) controls the density level of the displayed surface. Figure 4.2 shows the density surface for different density levels for the ball dataset (cf. Figure 4.1(a)). As the density level increases the density surfaces are adjusted to enclose high density data regions only. As a result the area of a surface decreases as the density level increases.



(a) $\alpha = 0.2$ (b) $\alpha = 0.5$ (c) $\alpha = 0.8$

Figure 4.2: Density Surfaces for Different Density Level α

The DS module visualizes a density surface as a set of points located on the density surface. Each point is displayed as a tetrahedra. Visually such a set of points is perceived as a surface as illustrated in Figure 4.3. The `Surface Grid Size` (the `Density Surfaces group`) controls the quality of the density surface by adjusting the coarseness of the surface grid. Figure 4.3 shows the density surfaces for the ball dataset for different surface grid sizes g_s. $g_s = 30$ produces a rough surface (cf. Figure 4.3(a)). As g_s increases to 75 we get visually good results (cf. Figure 4.3(c)).

As g_s increases the time to calculate and render the surfaces also increases. Timings and memory consumption for different surface grid sizes are presented in Table 4.2.

g_s	30	50	75	100
Computational Time (sec)	0.02	0.14	0.44	1.07
Memory Consumption (# surface points)	403	1’173	2’707	4’887

Table 4.2: Time (Sec) and Memory Consumption (Number of Points) for Different Surface Grid Size g_s

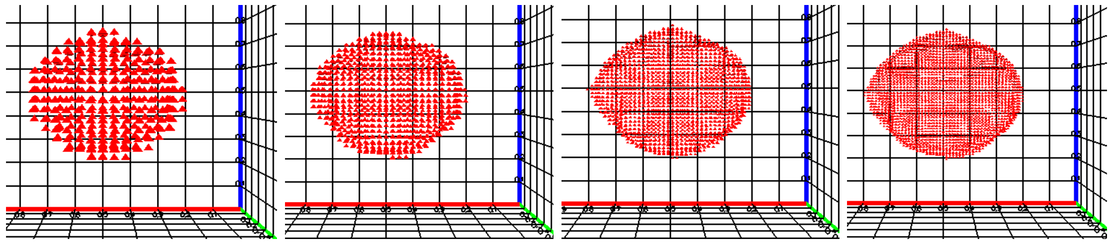

(a) $g_s = 30$ (b) $g_s = 50$ (c) $g_s = 75$ (d) $g_s = 100$

Figure 4.3: Density Surfaces for Different Surface Grid Size g_s

4.2 Animation of Density Surfaces

The animation of density surfaces is a visualization of a sequence of density surfaces with a decreasing density level.

Two parameters control the animation of DSes: `Surface grid size` and `DS frame size`. Basically, `Surface grid size` determines the level of detailness of the surfaces (cf. previous section). `DS frame size` determines the number of intermediate density surfaces in the sequence. For example if `DS frame size` is chosen to be four then four density surfaces of levels 1/5, 2/5, 3/5, and 4/5 will be visualized in a sequence. Since the printed version of the manual is limited and cannot show the animation we present the animation of density surfaces for the ball dataset side by side in Figure 4.4.

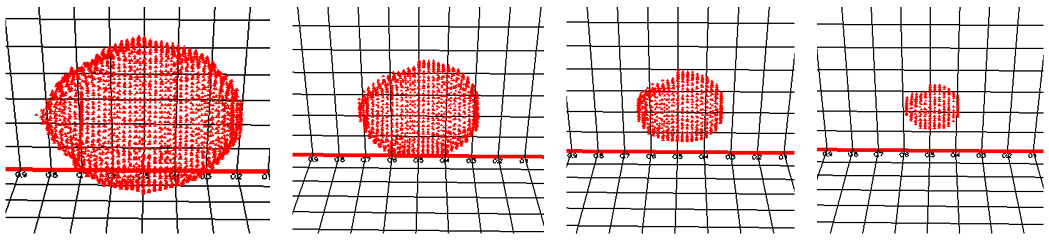

(a) $\alpha = 1/5$ (b) $\alpha = 2/5$ (c) $\alpha = 3/5$ (d) $\alpha = 4/5$

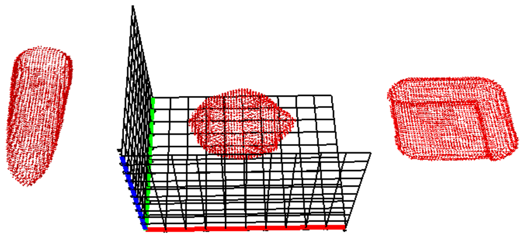
Figure 4.4: Density Surfaces for Different Surface Grid Size g_s

The speed of animation is determined by the speed of the computer, i.e., the surfaces are visualized as fast as possible. `DS Frame Size` (and `Surface grid size`) can be used to control the speed of the animation to some degree.

4.3 Multiple Data Sets

Three parameters control the visualization of density surfaces for multiple datasets: `Placement`, `Delete Prev` (the `DS Mapper` group), and `Vary DS color` (the `Density Surfaces` group). By default, the surface is visualized in the center of coordinate systems. The other possibilities include Left/Right, Bottom/Top, and Back/Front (cf. `Placement` in the `DS Mapper` group of the GUI).

If one continues with selecting different values of the `DataBase` and `X,Y,Z,W` parameters and `Delete Prev` is not checked, the program visualizes the density surfaces of all selected datasets during the investigation. It yields simultaneous visualization of the DSes for several datasets. En example of such a visualization is presented in Figure 4.5(a). First, the ball dataset was selected and visualized in the `Center` (`Placement`) of the coordinate system. Then we changed `Placement` to `Left` and we chose the cone (cf. Section 6.2) dataset. Finally, we changed the `Placement` to `Right` and visualized the plane (cf. Section 6.2) dataset.


(a) Surfaces are Visualized Side by Side

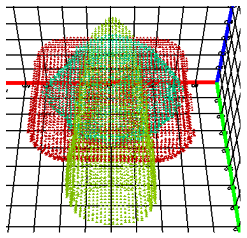

(b) Surfaces are Visualized on Top of Each Other

Figure 4.5: Density Surfaces for Different Surface Grid Size g_s

If `Vary DS color` is checked and several surfaces are visualized in the same `Placement`, the DSes will be shown in different colors. The colors are chosen automatically, maximizing the distance among them. The first visualized density surface is assigned to the hottest color (red), the last visualized DS is assigned to the coldest color (blue). An example where the ball, cone, and plane are visualized in the `Center` of `Placement` is presented in Figure 4.5(b).

If `Delete Prev` check-box is selected, the visualization is reset (all previuos visualizations are cleared).

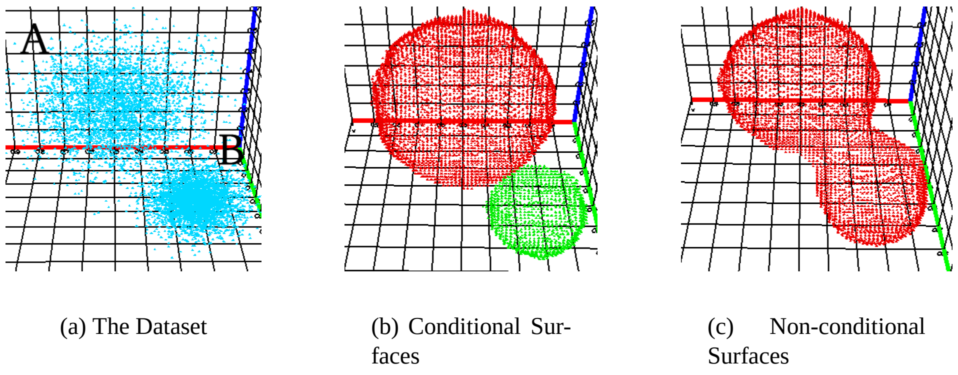
`Vary DS color` controls the coloring schema of the density surfaces. If check-box is selected the surfaces will be shown in different colors. Otherwise the same red color is used for all the surfaces.

4.4 Conditional Datasets

Conditional datasets are a special but important case of multiple data sets as described in Section 4.3.

The DS module can visualize the DSes for conditional datasets. If `Conditional` (the `DS Mapper` group) check-box is selected the module will split the dataset according to the `W` attribute into separate datasets. Then the DSes are calculated for each of the dataset and visualized in the `Placement`.

An example of density surfaces for conditional datasets is presented in Figure 4.6. We used the two-spheres example to illustrate the approach. The two-spheres dataset contains two normally distributed clusters in the three-dimensional space. The fourth attribute determines the cluster the observation belongs to. All observations, which fall into the cluster A has 1 in the `W` attribute, while observations, which fall into cluster B has 2 in `W` attribute. Figures 4.6(c) shows a DS for the dataset where the fourth attribute `W` is ignored. It yields one DS, which enclose both structures in space. Figure 4.6(b) shows a DS for the conditional dataset. First, the dataset is split into two datasets and then a separate DS is calculated for each of the dataset. It yields two (possibly intersecting) DSes.

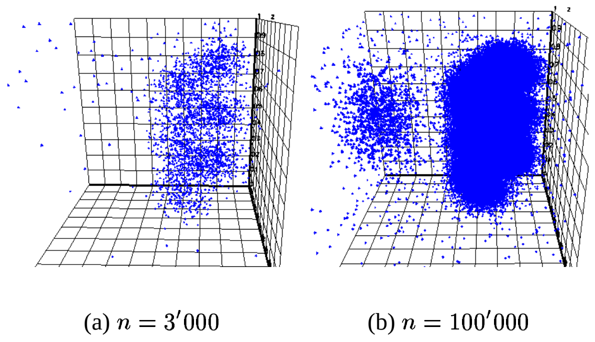


(a) The Dataset (b) Conditional Surfaces (c) Non-conditional Surfaces

Figure 4.6: Difference between Conditional/Unconditional Investigation

4.5 Equalization of Structures

Equalization of structures is particularly useful if the dataset contains several not equally pronounced structures. We describe the equalization of structures in terms of the spiral-sphere example.



(a) $n = 3'000$ (b) $n = 100'000$

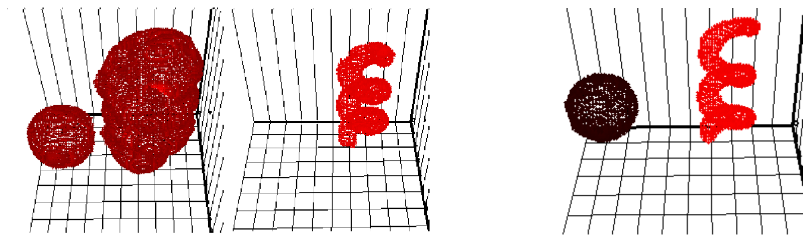
Figure 4.7: Scatterplots of Non-Equalized Structures

The spiral-sphere data in Figure 4.7 contains two structures: a spiral (80% of all observations) and a sphere (20% of all observations). In Figure 4.7(a) we chose the number of observations that yields the best visualization of the spiral. The result is a scatter plot that does not show the sphere. In Figure 4.7(b) we increase the number of observations until the sphere can be identified. This yields a scatter plot that makes it difficult to identify the spiral.

It is much easier to identify the structures if density surfaces are used (cf. Figure 4.8(a)). As the density level increases the density surfaces reveal the structures of the dataset (a spiral and a sphere). Since the densities of the structures are very different, a meaningful density level interval for the sphere is [0.1;0.15] and [0.2;0.8] for the spiral. The difference in the pronouncements of the structures complicates the investigation and comparison of the structures.

A solution is presented in Figure 4.8(b). Here the structures of the dataset are equalized by adaptively (based on the density of a structure) calculating the density level of the surfaces. The solution enables a fair investigation of the structures of not-equally pronounced structures.

The equalization of structures is available through the `Equalize structures` (the `Density Sur-`



(a) DS, Not Equalized (b) DS, Equalized

Figure 4.8: Equalization of Structures

`faces` check-box. Three preconditions have to be satisfied in order to use this functionality:

- The input dataset has to be sorted according to sub-streams (structures). The sub-streams do not overlap in the sequence (cf. Figure 4.9)
- Any subsequent slice of a sub-stream contains representative sample of the sub-stream (the sub-streams are not ordered)
- The sub-streams do not overlap in space.

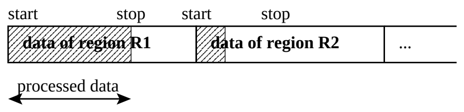


Figure 4.9: A Stream of Sub-streams

4.6 Selecting a Data Window

The window functionality enables to investigate a subset of the data in more detail. Technically, the function filters out the data points, which fall outside the selected window and re-scales the data points to be in unit cube.

Conceptually, the window functionality enables an investigation at the micro level (a “smart” zoom in the selected dataset). Note that a single surface at macro level can split into several surfaces at micro level.

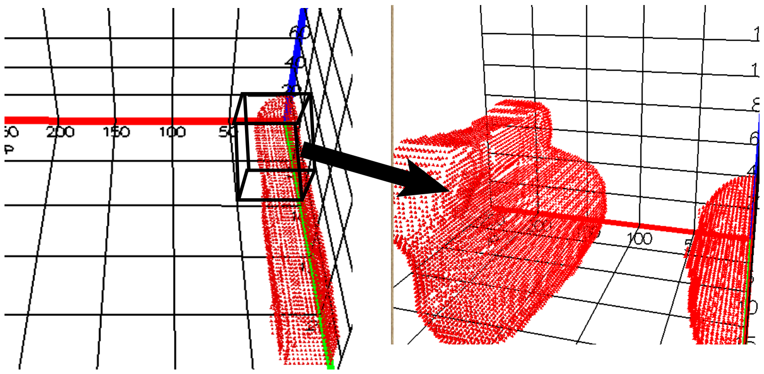


Figure 4.10: A Stream of Sub-streams

Part II

The Internals of the Density Surface Module

This part describes the internals of the density surface module and is relevant for developers who want to extend the method or the user interface. We show various code fragments to illustrate how the density surface module is implemented. By modifying these code fragments it is possible to expand the functionality of the density surface module.

Chapter 5 shows the class diagram of the DS module. Basically the DS module consists of three blocks. The main block is the APDF class. This class implements the density estimation and computes density surfaces. The algorithms are described in Chapter 5.2. The Mapper class integrates the DS module into the 3DVDM System. We show various code fragments that show how to do this integration. The conceptual design of the mapper class is part of the core 3DVDM System and is described elsewhere [3].

The last block, described in Chapter 6, is a utility class for generating artificial data sets. Artificial data is a valuable yardstick when interpreting density surfaces of real world data. Therefore, the DS module offers built-in support to simulate basic geometric shapes and render the corresponding surfaces together with the surfaces of the random dataset.

Chapter 5

Structure of the Code

5.1 Class Diagram

Figure 5.1 depicts the component diagram of the DS module. There are three main components in the system: the `Density Surface`, `Utility Functions`, and `Random Variables`.

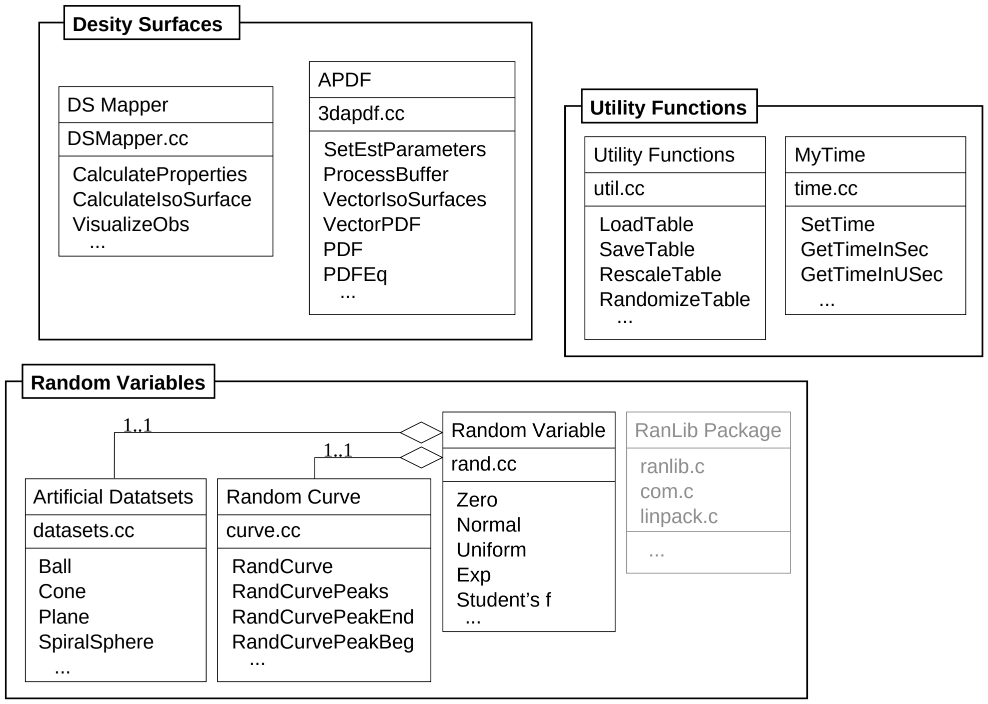


Figure 5.1: The Component Diagram of the Density Surfaces Module

The key parts of the DS module are the individual functions (methods) and the associated data structures. The structure of the code is kept simple and does not contain any inheritance or association relationships. There are two aggregation relationships in the `Random Variables` component, though.

The `Density Surface` component consists of the `DS Mapper` and `APDF` classes. The `APDF` class is used to estimate a PDF of the dataset (the `setEstParameters` and `ProcessBuffer` functions, Density Surface Component, Figure 5.1) and calculate a density surface from the PDF (the VectorIsoSurface function). The DS Mapper class integrates the APDF functionality with the rest of the 3DVDM/VR++ system. Basically, the DS module implements three main functions: CalculateProperties, CalculateIsoSurface and VisualizeObs. CalculateProperties implements the DS module algorithm (cf. Section 5.3). The function uses CalculateIsoSurface and VisualizeObs sub-functions to recalculate the DS or visualized observations.

The Utility Function component implements auxiliary functionality like randomization and rescaling of the dataset (RescaleTable, RandomizeTable), and keeping track of the computational time of the relevant functions (the MyTime class).

The Random Variables component implements simulation of random variables. The component uses the RanLib package developed by Barry W. Brown and James Lovato. The Random Variables contains three classes: Artificial Dataset, Random Curve, and Random Variable. Basically the Random Variable class provides a C++ interface to the RanLib package. The class provides functions to simulate basic one-dimensional random variables (for e.g., normal, uniform, exponential, etc.). The Random Curve class implements a general approach to simulate a dataset around a curve in the three-dimensional space. The Artificial Datasets implements a number of artificially simulated datasets. Examples include the ball, cone, plane, and spiral-sphere datasets (cf. Section 6.2). Some of the functions uses the Random Curve class to simulate the datasets (for example the spiral-sphere, torus datasets).

5.2 The APDF Class

The estimatePDF algorithm processes the data in slices to estimate the PDF of the dataset. It starts with a uniform grid data structure (the granularity is determined by the InitG parameter for each axis) and processes the first slice of the data. The estimation process can be in two states: the estimation or skipping state. In the estimation state the algorithm refines the PDF by adding new kernels and refining the grid structure in regions where the PDF is non-linear. The process is continued until the desired precision ε is reached. When the estimation is precise enough it computes the range of the structure in space (a minimum bounding cube) and enters the skipping state. The range is used for the equalization of structures. In the skipping state the algorithm skips the slices until it finds new information that is not yet reflected in the PDF. If such a slice is found the processing is resumed. The individual steps of the algorithm are shown in Algorithm 5.2.1.

Algorithm 5.2.1 *Estimation of the Probability Density Function*

```
Algorithm: estimatePDF
Input:
  vDataset: sequence of data points
  $\varepsilon$: accepted precision
  InitG: initial number of grids

Output:
  APDF tree $a$
Body:
  1. Initialize $a$
  2. skipState = FALSE
  3. FOR EACH slice $s_i$ DO
       3.1 IF !skipState THEN
```

```
            3.1.1 Process slice $s_i$.
            3.1.2 Split the space according to the precision of the estimation
            3.1.3 IF precisionIsOK(a) THEN
                    skipState = TRUE
                  END IF
      3.2 ELSIF newStream($s_i$) THEN
            3.2.1 skipState = FALSE
          END IF
    END FOR
```

The calculateDS algorithm calculates a density surface of α density level. Mathematically, the idea of the algorithm can be expressed by the formula $\partial\{(x,y,z) : PDF(x,y,z) > \alpha \cdot \max\}$, where ∂A is the border of dataset A. Basically, the algorithm scans the unit cube and checks whether the value of the PDF is on the border of the $\alpha \cdot \max$ density. The algorithm returns the border points that, when visualized, are perceived as a surface. To get equalized structures the algorithm uses the local maxima. The global maxima is used for not equalized structures. The individual steps of the algorithm are presented in Algorithm 5.2.2.

Algorithm 5.2.2 *Algorithm of the DS Calculation Module*

```
Algorithm: calculateDS
Input:
  PDF         : PDF of the dataset
  iDSGridSize : initial number of grid points
  $\alpha$           : density level
  equalization: equalization of the structures

Output:
  S           : DS of level $\alpha$

Body:
  1. FUNCTION IsBorderPoint($PDF$,i,j,k,$\beta$)
  2.   RETURN ($PDF[i,j,k] \geq \beta$) AND ($PDF[i',j',k'] < \beta$)
                 for some $(i',j',k') \in (i+h_1, j+h_2, k+h_3)$
                 where $h_1,h_2,h_3 = -1,0,1,\ \ |h_1|+|h_2|+|h_3| = 1$
  3. END FUNCTION

  4. $S = \emptyset$
  5. FOR $i,j,k = 1$ TO iDSGridSize DO
       5.1 Calculate the local (max$_l$) and the global ((max$_g$))
           maximum of the PDF of the structure $i,j,k$ is in
       5.2 IF equalization THEN
           5.2.1 IF IsBorderPoint($PDF,i,j,k,\alpha \cdot \max_l$) THEN $S = S \cup PDF[i,j,k]$
       ELSE
           5.2.2 IF IsBorderPoint($PDF,i,j,k,\alpha \cdot \max_g$) THEN $S = S \cup PDF[i,j,k]$
       END IF
     END FOR
```

5.3 The DS Mapper Class

The DS module operates in multi-process/multi-threading environment. Once started, the module is loaded into main memory and can be in one of two states: active (in the process mode) and inactive (waits for an wake up event). The module is active if: (i) a change of the input parameters has triggered the recalculation of the PDFs and/or density surfaces or (ii) the animation of density surfaces is switched on.

We show the core code fragments to illustrate the working of the DS module.

The global state variables of the module are mapped to corresponding GUI elements (cf. Figure 3.1). At any particular time the parameters determine the state the module is in.

Pseudo Code Fragment 5.3.1 *Input/State/Output parameters of the DS module*

```
Input parameters:
  bAnimate       : animation of the DSes (yes/no)
  bConditional   : conditional treatment of the dataset (yes/no)
  bEqualized     : equalization of the structures (yes/no)
  bVaryColor     : vary the DSes color (yes/no)
  bDeletePrev    : delete previous animations (yes/no)

  cPlacement     : placement of the DSes (left/right/bottom/top/back/front)

  fDSLevel       : density level of the DS ([0.0,1.0] value)
  iDSGridSize    : DS grid size ([1,100] value)

  EstErr         : Estimation Error of PDF
  InitG          : Initial number of grid points in each dimension

  vDataset       : input dataset (a vector of (X,Y,Z,W) values)

Output/State parameters:
  vPDF           : the set of created PDFs
  (vSurfaces, vPlacement) : the set of DSes and corresponding placements
  animateDSLevel : density surface number in a sequence of animated surfaces
  animateDSFrames: size of sequence of animated surfaces
```

The initialization part is invoked when the program is loaded into main memory.

Pseudo Code Fragment 5.3.2 *Initialization of the DS module*

```
Initialization:
  vPDF = vSurfaces = vPlacement = ∅
  bAnimate = NO, bConditional = YES, bEqualized = NO, bVaryColor = YES
  EstErr = 0.02, InitG = 7
  fDSLevel = 0.2, iDSGridSize = 75
```

Pseudo Code Fragment 5.3.3 sketches the event handling of the DS module. Basically, the code consists of three conditional (blocks 1–3) and one unconditional block (the 4th block)

The first block handles new dataset event. The block deletes the previous visualizations if `Delete Prev` is checked (sub-block 1.1), splits the dataset into conditional datasets if `Conditional` is checked (sub-block 1.2), and calculates the PDFs for the individual datasets (sub-block 1.3).

The second block handles animation events. It calculates the next density surface level (sub-blocks 2.1-2.2) and computes the corresponding density surfaces (sub-blocks 2.3-2.4).

The third block handles events triggered by a change of the density level. This requires recalculation of the density surfaces of the PDFs at the requested density level.

The 4th block visualizes the computed density surfaces and, if selected, corresponding data samples.

Pseudo Code Fragment 5.3.3 *Algorithm of the DS module*

```
1. IF new(vDataset) THEN
   1.1 IF bDeletePrev THEN
       1.1.1 vPDF = vSurfaces = vPlacement = ∅
       END IF
   1.2 IF bConditional THEN
          1.2.1 Split vDataset according to the categorical attribute into
                $D_1$,...,$D_k$.
       ELSE
          1.2.2 $D_1$ = vDataset
       END IF
   1.3 FOR EACH Dataset $D_i$ DO
       1.3.1 vPDF = vPDF ∪ estimatePDF($D_i$, EstErr, InitG)
       1.3.2 vPlacement = vPlacement ∪ cPlacement
       END FOR
   END IF

2. IF bAnimate THEN
      2.1 animateDSLevel += 1.0 / animateDSFrames
      2.2 IF animateDSLevel >= 1.0 THEN
             2.2.1 animateDSLevel = 1.0 / animateDSFrames
      END IF
      2.3 vSurface = ∅
      2.4 FOR EACH $vPDF_i$ DO
          2.4.1 vSurface = vSurface ∪
          calculateDS($vPDF_i$, animateDSLevel, iDSGridSize, bEqualized)
          END FOR
      2.5 Request for calculation
    END IF

3. IF !bAnimate AND changed(fDSLevel) THEN
      3.1 FOR EACH $vPDF_i$ DO
          3.1.1 vSurface = vSurface ∪
           calculateDS($vPDF_i$, fDSLevel, iDSGridSize, bEqualized)
   END IF

4. visualizeObs()
```

5.4 Application Programming Interface

The functions of the APDF class can be divided into two groups: object creation and query functions. The object creation group enables the estimation of the PDF in a distributed environment where data is streamed between components. The query functions provide functionality to query density related information.

The object creation group consists of the `APDF`, `setEstParameters`, and `processBuffer` functions:

```
APDF();
```

```
void setEstParameters (
       float err = 0.1,
       int interpFunction = LINEAR,
       int stop = 1,
       int constHopt = 1,
       int initGrid = 3,
       int numberOfsplits = AUTO_SPLIT,
       int debugInfo = YES );

int requiredBufferSize();

float requiredError(
        int nObs );

void processBuffer (
       float **buffer,
       int bufferSize );
```

In order to estimate Probability Density Function of the dataset, the functions have to be invoked in certain order. First, the APDF object (a memory resident tree) has to be created (instantiated), and then the estimation parameters (the `setEstParameters` function) can be set. The estimation of the PDF is refined by invoking `processBuffer` for a sequence of buffers. The size of the buffers has to be consistent with the selected parameters, and can be obtained with the help of the `requiredBufferSize` function. The arguments of these functions are summarized below:

- `err`. The estimation error.
- `interpFunction`. Interpolation function that is used to calculate the PDF at non-grid points; can be CONST or LINEAR.
- `stop`. Controls the skip state. If the parameters is `1` then skipping is enabled. Otherwise all buffers will be processed.
- `initGrid`. The initial number of grid points per axis.
- `numberOfSplits`. Number of new grid points that shall be added in regions where the PDF is non-linear.
- `debugInfo`. Determines whether debug info will be printed to standard output.
- `buffer`. A buffer of three-dimensional data points (e.g., `buff[i][0]` is the first coordinate of the ith point)

An example of the creation of an APDF object is given in Section 6.1.

The query group consists of the following functions:

```
float pdf (
        float x,
        float y,
        float z );

float pdfEq (
        float border,
        float x,
        float y,
        float z );

vector<float *> *vectorIsoSurface (
                   float level,
                   int granularity,
                   int equalized = 0,
                   int recalcMax = 1,
                   bool debug = YES );
```

The pdf (pdfEq) function calculates the (equalized) value of the PDF at point (`x,y,z`). The `vectorIsoSurface` function calculates the density surface. The arguments are described below:

- `level`. The density level
- `granularity`. Number of grid points used to calculate the surface
- `equalized`. If `1` then the surfaces will be equalized.
- `recalcMax`. Forces the recalculation of the max of the PDF.
- `debug`. Determines whether the debug info will be printed to standard output.

Function `vectorIsoSurface` returns a vector of points on the surface. The points are ordered according to X,Y,Z. The ordering is illustrated in Figure 5.2.

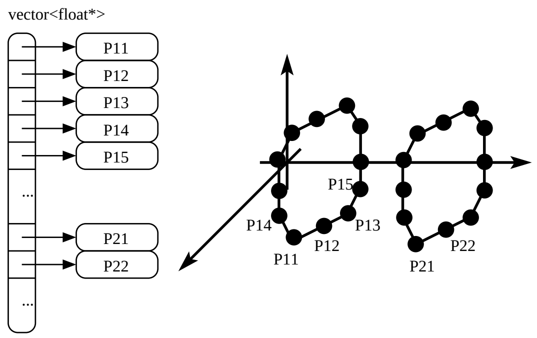


Figure 5.2: The Order of the Surface Points

En example of how to use this function is given in Section 6.1.

Chapter 6

Data Sources

The DS module comes with a library of mathematical functions that can be used to generate artificial data sets. Artificial data sets are valuable yardsticks when evaluating (extensions of) the density method and when analyzing real world data sets. The first section describes how to work with externally stored data sets (e.g., real world data sets). The second section describes how to generate artificial data sets on the fly.

6.1 External Data Sets

In order to calculate a PDF and a density surface for a dataset stored in a file six basic steps have to be performed. First, the estimation parameters have to be set. Next, the required size of buffer is calculated. Then a data stream is opened and the slices are formed and processed (steps 3–5). The density surface at the given level (in our example 0.2) is calculated in the last step.

Example 6.1.1 Create a PDF and a DS from a file.

```
#include <3dvdm/ds/3dapdf.h>

int bufSize
APDF ap;

// 1) Setting the estimation parameters
ap.setEstParameters( 0.03, LINEAR, 2, 0, 7, AUTO_SPLIT, YES );

// 2) Calculating the required buffer size
buferSize = ap.requiredBufferSize();

// 3) Opening a data stream from the file
ifstream fi( ‘‘datafile’’ );
while (fi.eof() != 1) {

  // 4) Preparing a slice of data points
  // We assume that the file contains numerical values scaled to [0..1]
  // 3D data points
  for (int i = 0; i < bufSize; i++) {
    fi >> buf[i][0];
    fi >> buf[i][1];
    fi >> buf[i][2];
  }

  // 5) Processing the slice
```

```
   ap->processBuffer( buf, bufSize );
 } // The APDF is created

 // 6) Calculation of the DS
 vector<float *> vIsoSurface = ap.vectorIsoSurface( 0.2, 7, NO, 1, YES );
```

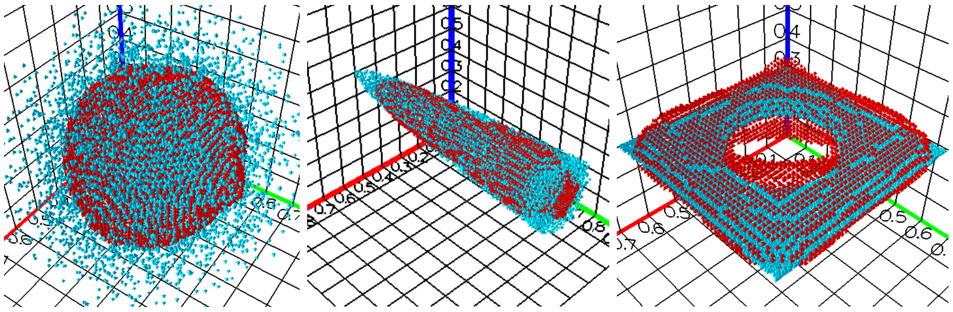
6.2 Artificial Data Sets

The `Datasets` class contains a number of functions to create artificial datasets. The currently available functions are listed in Code Fragment 6.2.1. The selection of functions is fairly arbitrary and was guided by specific experiments and data analyses that we performed during the development of the DS module. All functions have the number of points to be created as an input parameter. The `randomLine` function has several additional parameters to control the generation of a random polygonal line in the three-dimensional space:

- `nObs`. The number of observations that shall be generated.
- `breaks`. Number of vertexes in the polygonal line.
- `smoothness`. The smoothness of the polygonal line. Possible values vary between 0 and 1. The parameter controls the angle of the vertex. A low value produces a smooth polygonal line in space while a large value yields a nervous line.
- `var`. Controls the spreadness of the data points around the polygonal line. The parameter must be positive.
- `noiseRate`. Controls the noise rate in the dataset.

Code Fragment 6.2.1 *The list of the public functions of the* `Datasets` *class*

```
 vector<float *> *spiralLine( int nObs );
 vector<float *> *spiralSphere( int nObs );
 vector<float *> *fourClusters( int nObs );
 vector<float *> *oneCluster( int nObs );
 vector<float *> *randomLine(
    int nObs,
    int breaks,
    float smoothness,
    float var = 0.5,
    float noiseRate = 0.1 );
 vector<float *> *oneBigOneSmall( int nObs );
 vector<float *> *SLS( int nObs );
 vector<float *> *cone( int nObs );
 vector<float *> *stripe( int nObs );
 vector<float *> *cube( int nObs );
 vector<float *> *cubeSphere( int nObs );
 vector<float *> *cubeSeveralSpheres( int nObs );
 vector<float *> *plane( int nObs );
 vector<float *> *planeSphere2D( int nObs );
 vector<float *> *planeSphere3D( int nObs );
 vector<float *> *planeHole( int nObs );
 vector<float *> *torus( int nObs );
```



(a) `OneCluster` (b) `cone` (c) `planeHole`

Figure 6.1: Illustration of the Artificial Datasets

The `oneCluster`, `cone`, and `planeHole` datasets together with a corresponding density surface are illustrated in Figure 6.1.

The following code fragments demonstrate the use of these functions.

Example 6.2.1 Creating the plane dataset.

```
#include <3dvdm/ds/datasets.h>
#include <3dvdm/ds/util.h>

// 1) Instantiate a dataset object
Datasets d;

// 2) Simulate 100’000 of data points
vector<float *> plane = d.plane( 100000 );

// 3) Simulate another dataset of 100’000 of data points
vector<float *> ball = d.oneCluster( 100000 );

// 4) merge the datasets
vector<float *> merged = mergeTables( plane, ball );

// 5) mix the datasets
vector<float *> mixed = randomize( merged );

// 6) Rescale the data to be between 0 and 1
vector<float *> rescaled = rescaleTable( merged );

// 7) Save the data in a data file
saveTable( "plane-ball", rescaled );
```

Example 6.2.2 Creating a dataset with two balls.

```
#include <3dvdm/ds/rand.h>
#include <values.h>

// 1) A front-end to access random variables
Rand r;
```

```
// 2) Simulation of the three-dimensional vector
// simulated around (0,0,0) with variance (1.0,1.0,0.5)
vector<float *> ball1 = r.vec( 10000, 3,
    NORMAL, 0.0, 1.0,
    NORMAL, 0.0, 1.0,
    NORMAL, 0.0, 0.5 );

// 3) The second ball is
// simulated around (3,3,3) with variance (0.5,1.0,1.0)
vector<float *> ball2 = r.vec( 10000, 3,
    NORMAL, 0.0, 0.5,
    NORMAL, 0.0, 1.0,
    NORMAL, 0.0, 1.0 );

// 4) Merging the dataset
vector<float *> merge = mergeTables( ball1, ball2 );
```

Example 6.2.3 Simulation of a spiral. To create a dataset that is spread around a curve, the curve has to be given in parametric form: $(x, y, z) = (x(t), y(t), z(t))$. The simulation is done in two steps. First, a data object is created (step 2), and then the actual simulation is produced (step 3).

```
#include <3dvdm/ds/curve.h>
#include <values.h>

// 1) A curve in the parametric form:
// (x(t),y(t),z(t))=(sin(t),cos(t),t) (a spiral)
double spiralT( double t )  return t / 2;
double sinSpiral( double t )  return sin( t );
double cosSpiral( double t )  return cos( t );

// 2) Simulating a (N(0,1), N(0,1), N(0,1)) random variable
// around the spiral
Curve spiral(3, spiralT, NORMAL, 0.0, 1.0,
                cosSpiral, NORMAL, 0.0, 1.0,
                sinSpiral, NORMAL, 0.0, 1.0 );

// 3) ‘‘scanning’’ the curve as t=0..4*M_PI (two cycles).
// 100’000 observations are simulated
vector<float *> spiral = spiral.randCurve( 4*M_PI, 100000)
```

Part III

A Data Mining Case Study

The last part describes a data mining case study. We analyze the web log of a heavily used server that mirrors a large number of open source projects.

The analysis of the web log is challenging for a number of reasons. First, the data volume is huge and therefore provides a good yardstick for the scalability of a tool—both in terms of efficiency and effectiveness. Another characteristics is the highly skewed distribution of the data. While this is potentially valuable information it must still be possible to go beyond the big picture and analyze the data at the micro level, e.g., analyze domains that generate significantly less traffic than the top most two domains.

Clickstream data has attracted a lot of interest in the past. Among others, solutions have been proposed that turn a web log into a webhouse that is much better suited for data analysis. Such a data preparation step greatly enhances the chances for successful data analyses. We omit the data preparation step because our focus is the density surface method itself. Thus, our main goal is to illustrate how to use density surfaces rather than a concrete interpretation of the web log. If the concrete interpretation is the main interest it is necessary to go through the aforementioned data preparation step.

In our analysis we primarily investigate the dynamics of the number of visitors. We analyze several domains by comparing the shapes, orientation, and location of the structures. Since the clickstream contains very unbalanced structures (for example the number of clicks of the most frequent European domains (.dk and .fr) differs by an order of magnitude) it is essential to be able to equalize the structures. This greatly supports the visual comparison of the structures and ensures that interesting patterns do not drown in over-pronounced structures.

Chapter 7

Analyzing the Clickstream Data from sunsite.dk

7.1 The Dataset and the Analysis

The SunSite.dk (`http://www.sunsite.dk`) server hosts and mirrors many open source software projects. It is widely used in Scandinavia and Europe as a mirror for open source projects because of its high update rate (at least once a day all the packages are updated) and a fast and reliable connection. The server generates a web log of more than 4GB (more than 3.5 million clicks) per day.

The web log contains information about the clicks (requests) that the web server handled. For each click the web server logs information about the visitor (domain of the computer, browser info, etc), the date and time of the request, and the document retrieved (the actual path to the file, its size, etc). We mapped the main dimensions (the IP, path to a file, time) to the spatial coordinates (X, Y, Z) of the visual world.

The data is coded according to the sequence in which it appears in the log (cf. Figure 7.1). For example, the first domain that is found gets code 0, the second domain code 1, etc. Such a mapping results in a

```
Domain, Document,   Time  Coded as   X, Y,   Z
dk       /          12:00    ->      0, 0, 12:00
com      /recipes/  12:01    ->      1, 1, 12:01
dk       /samba/    12:01    ->      0, 2, 12:01
```

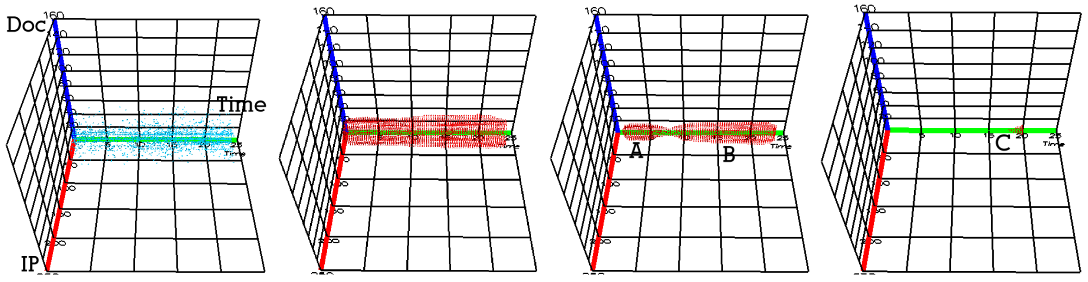
Figure 7.1: The Natural Ordering of the Data

visualization, where frequent items (e.g., .dk domain) get a low codes and rare items get a high code.

Figure 7.2(a) displays the clickstream data for one day. Each observation corresponds to a web-click recorded by the server. Because the coding preserved the natural ordering of the data all frequent domains (documents) are clustered at the beginning of the IP (Doc) axis.

Figure 7.2 shows an overall work-load of the web-server. The work-load is high during the very early morning hours and the local working hours (cf. A and B Figure 7.2(c)). The work-load peaks at 8PM (cf. C, Figure 7.2(d)).

As we will see later (cf. Sections 7.3, 7.4, and 7.5) the cluster (cf. Figure 7.2(b)) contains clicks from quite different domains.

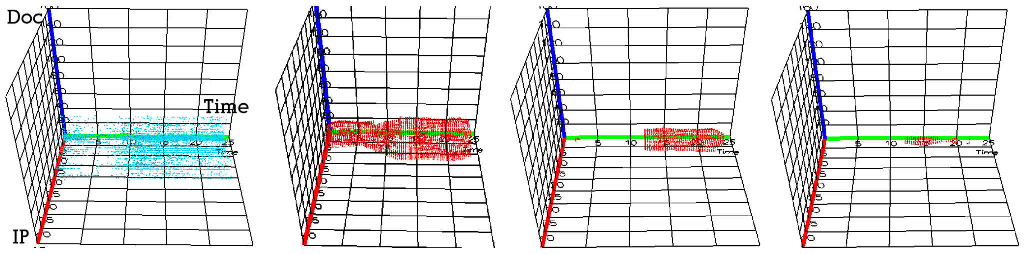


(a) The Scatter Plot (b) $\alpha = 0.2$ (c) $\alpha = 0.5$ (d) $\alpha = 0.9$

Figure 7.2: The Sunsite Dataset (Natural Ordering)

To get a better understanding it is necessary to decrease the granularity of the analysis. Towards this end the window functionality, conditional data sets, and the equalization of structures can be used. Below we make use of all these techniques. In addition we also investigate subsets of the data. These subsets were produced with the help of simple shell scripts. Essentially, the shell scripts were used in place of a more general window function.

Figure 7.3 shows the clicks received from European domains. The dataset is highly unbalanced: more than 55% of all clicks are received from Denmark, 8% of clicks are received from France, ∼ 4% are received from Germany, as well as Italy, the UK, and Spain. The huge difference in the number of clicks among the countries is reflected in the density surfaces (cf. Figure 7.3(b)–7.3(d)). The density surfaces only catch clicks from Denmark and omit the less pronounced domains. The solution to the problem is to use equalization and/or conditional datasets (cf. Sections 7.2 and 7.4).



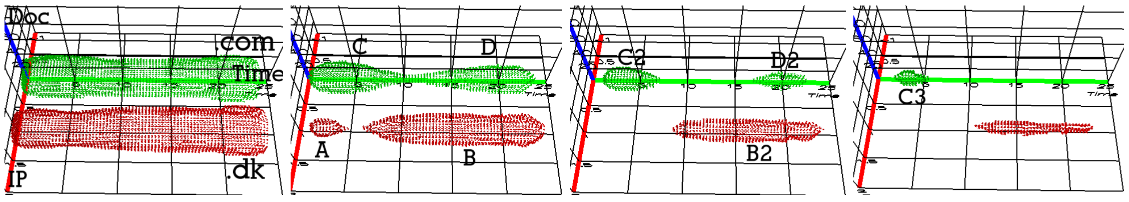
(a) The Scatter Plot (b) $\alpha = 0.2$ (c) $\alpha = 0.5$ (d) $\alpha = 0.9$

Figure 7.3: Europe

7.2 Comparing .com, .dk, and .fr

This section analyzes click received from the three most popular domains: .com (54% of the filtered dataset), Denmark (39%), and France (7%). Even for the top three domains the number of observations differs significantly. We use conditional dataset and equalization to compare the domains.

Figure 7.4 compares the density surfaces for .com and .dk. The Figure 7.4(b) shows that .com and .dk are very different in nature. .com covers computers from many time zones, mostly the US ones. This yields two main clusters (C and D in Figure 7.4(c)). Note that the web-server records the local time (Denmark time) of the actual clicks. Therefore cluster D corresponds to the beginning of the day and cluster C to the end of the day in the US. Flexible working hours and the distribution over many time zones results in a smooth increase and decrease of the number of visitors from .com. That is reflected by the cone-like surfaces. Cluster $C3$ (Figure 7.4(d)) shows that .com visitors peaks in the early morning hours.

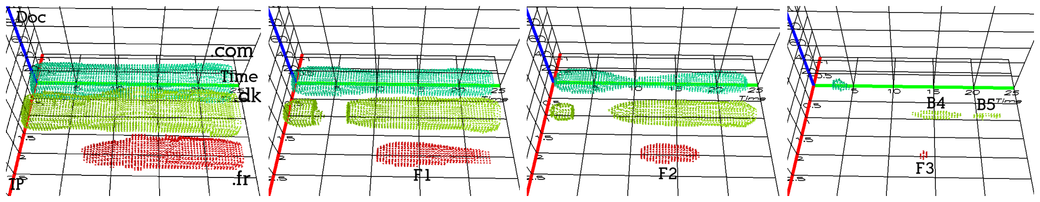


(a) $\alpha = 0.2$ (b) $\alpha = 0.5$ (c) $\alpha = 0.7$ (d) $\alpha = 0.9$

Figure 7.4: .com, .dk

Cluster A (Figure 7.4(b)) shows that there is a group of people visiting the web-site in the very early morning hours. Possibly these clicks represent the last e-mail checks at the end of the day. Cluster B is bound by the beginning of the working hours in Denmark and midnight. The shape of the .dk surfaces is much more rectangular than the shapes of the .com surfaces and indicates a more rapid change in the number of visitors. At the beginning of the working hours there is a fairly sharp increase (compare with Clusters C and D, Figure 7.4(b)) and at the end of the day a sharp decrease of the number of visitors. The Denmark domain peaks during the working hours (Cluster $B4$, Figure 7.5(d)) and in the evening (Cluster $B5$, Figure 7.5(d)).

The number of visitors from France increases relatively fast in the morning (cf. Cluster $F1$, Figure 7.5(b)), peaks around lunch (Cluster $F3$, Figure 7.5(d)), and decreases smoothly during the second part of the day (cf. Cluster $F1$, Figure 7.5(b)). The main work-load from France is bound by the working hours (cf. Cluster $F2$, Figure 7.5(c)).

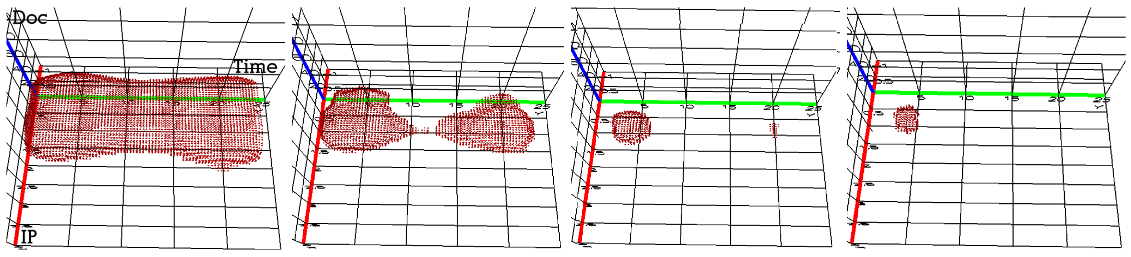


(a) $\alpha = 0.2$ (b) $\alpha = 0.3$ (c) $\alpha = 0.5$ (d) $\alpha = 0.95$

Figure 7.5: .com, .dk, .fr

7.3 Comparing .net, .com, .edu, and .us

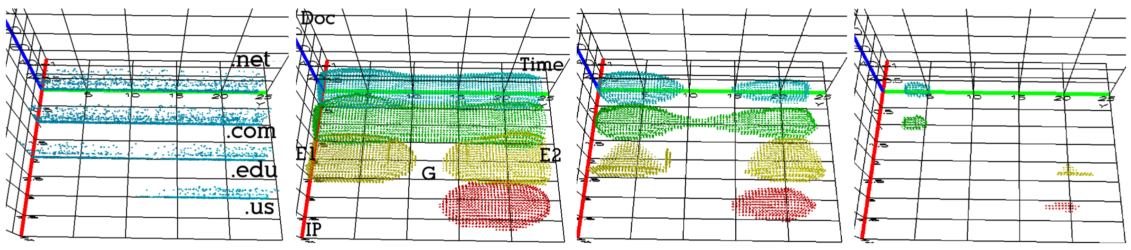
In this section we analyzes clicks received from the .net (36% of the filtered dataset), .com (53%), .edu (8%), and .us (3%) domains. Figure 7.6 shows the density surfaces for the unconditional dataset. Clearly, .com dominates and overwhelms the other domains.



(a) $\alpha = 0.2$ (b) $\alpha = 0.5$ (c) $\alpha = 0.8$ (d) $\alpha = 0.9$

Figure 7.6: .com .edu .net .us (unconditional)

Figure 7.7 shows the same data using equalized conditional density surfaces. .net and .com exhibit a similar pattern. They are quite diverse and the density surfaces resemble cone shapes. The pattern is more pronounced for .com.



(a) Scatter Plot (b) $\alpha = 0.2$ (c) $\alpha = 0.5$ (d) $\alpha = 0.9$

Figure 7.7: .com .edu .net .us (conditional)

The .edu domain consists of two large clusters $E1$ and $E2$. The clusters correspond to the working hours of the universities and research institutions in the US. Note that the split into two surfaces is artificial because we start the time line at midnight local time.

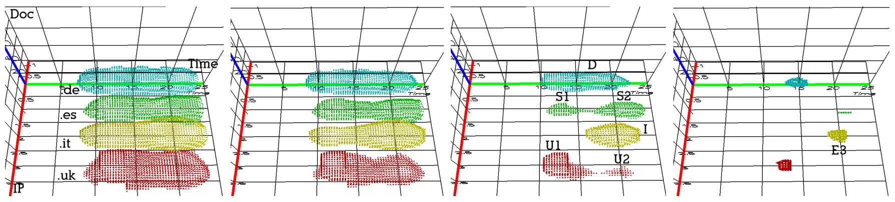
The clusters show a large flexibility in the working hours. There is a great variety in the working style of the research people (people who prefer to work in the morning, or in the evening). The accumulation of all the types of clicks yields a sharp increase and decrease in the number of visitors, gaping for the several hours at the late night hours in the US (Gap G, Figure 7.7(b)). The .edu domain peaks in the morning hours in the US (cluster $E3$, Figure 7.7(d)).

The .us domain corresponds to the working hours in the US. As expected this domain is more homogeneous and regulated.

7.4 Comparing .it, .de, .uk, .sp

Italy (30% of the filtered dataset), Germany (28%), the UK (22%), and Spain (20%) account for the second largest group of clicks in the click-stream data. Figure 7.8 shows the density surfaces for the filtered dataset.

All countries are very different in culture and working style. The density surfaces nicely reflect this. For example in Italy and Spain people have a long lunch break (siesta) in the middle of the day. The siesta



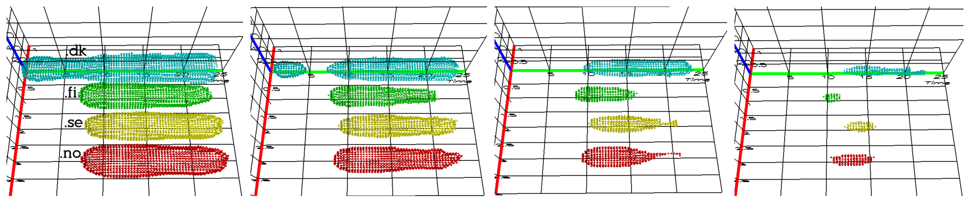
(a) $\alpha = 0.2$ (b) $\alpha = 0.3$ (c) $\alpha = 0.5$ (d) $\alpha = 0.9$

Figure 7.8: .it, .de, .uk, .es

divides the density surface for Spain into two similar sized surfaces (cf. $S1$ and $S2$, Figure 7.8(c)). Italy produces most of the clicks during the second part of the day (cf. surface I, Figure 7.8(c)). In contrast most of the UK clicks are received in the first part of the day (cf. surface $U1$, Figure 7.8(c)). The German domain is bound by the working hours, peaking in the afternoon. Figure 7.8(d) shows the peaks for the individual domains.

7.5 Scandinavian Countries

Figure 7.9 shows the density surfaces for the Scandinavian countries. The filtered dataset contains .dk (93% of the filtered dataset), .fi (2%), .se (3%), and .no (2%). The Scandinavian domains show very similar patterns in the overall behavior. The surfaces are shifted in the time dimension because of different time zones.

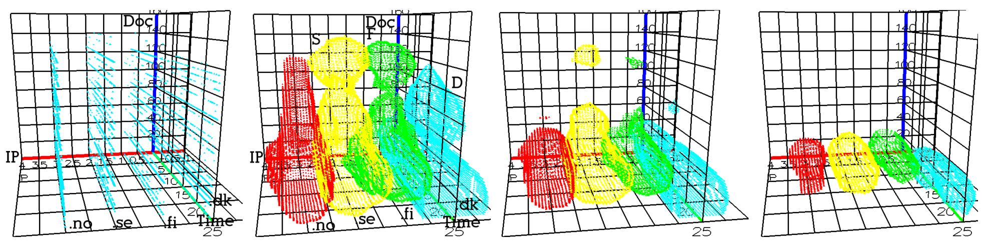


(a) $\alpha = 0.3$ (b) $\alpha = 0.5$ (c) $\alpha = 0.7$ (d) $\alpha = 0.9$

Figure 7.9: Scandinavia

Figure 7.10 shows the Scandinavian domains for the most popular documents only. In the previous analyses the document dimension did not contribute interesting information. The reason is that a few documents (e.g., the home page of sunsite.dk) are so heavily accessed that they dominate all other document accesses. One has to appropriately restrict the document dimension (similar to what we did for the domain) to analyze it. Doing so yields genuine surfaces in the 3D space as illustrated in Figure 7.10.

The analysis show different patterns with respect to both the time and the domain dimension. The dataset contains documents, which are visited during the working hours (cf. Clusters S, F, and D, Figure 7.10(b)). The shifts in the vertical alignment of the surfaces (cf. Figure 7.10(c) and 7.10(d)) indicate a difference in the most popular documents among countries.



(a) Scatter Plot (b) $\alpha = 0.1$ (c) $\alpha = 0.2$ (d) $\alpha = 0.5$

Figure 7.10: Scandinavia (the Most Popular Documents Only)

Bibliography

[1] *LAM / MPI Parallel Computing, MPI Tutorials: Getting started with LAM/MPI*, October 2001. http://www.lam-mpi.org.

[2] *VR Juggler Getting Started Guide*, September 2002. http://www.vrjuggler.org.

[3] H. R. Nagel. *The 3DVDM System*, to appear.

[4] H. R. Nagel. *Introduction to VR++, Getting Started Manual for version 0.5.1*, December 2002.

[5] H. R. Nagel. *VR++ – Reference Manual, Reference Manual for version 0.5.1*, December 2002.