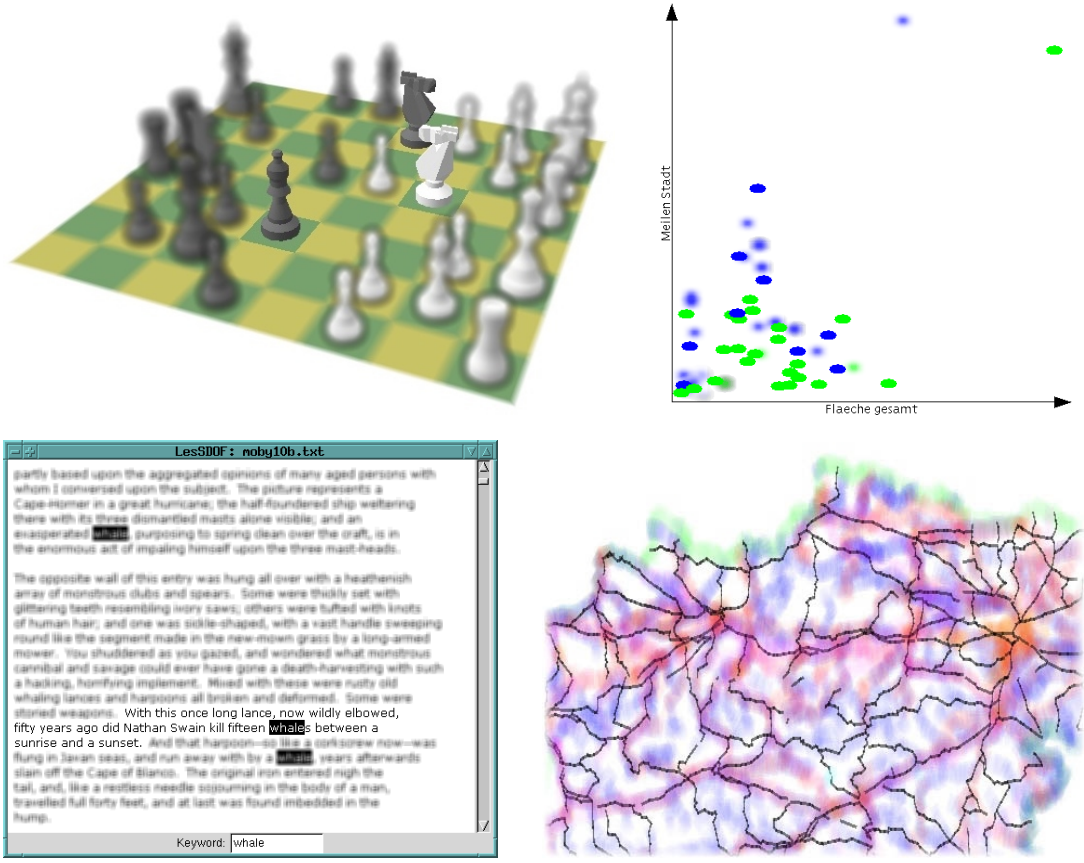
Robert Kosara

Semantic Depth of Field – Using Blur for Focus+Context Visualization

(PhD Thesis)



http://www.asgaard.tuwien.ac.at/~rkosara/
http://www.kosara.net/research/sdof/
mailto:rkosara@asgaard.tuwien.ac.at or mailto:rkosara@acm.org

Abstract

One central task in information visualization and related fields (like volume and flow visualization) is displaying information in a context that makes it easier for users to understand.

Blur is a visual cue that has been playing an important role in photography for over 150 years, but has been widely ignored in computer graphics. Sharp objects in photographs immediately attract the viewer's gaze – distinguishing between sharp and blurred objects therefore is very well suited for directing the viewer's attention to certain objects or parts of the image.

In this thesis, a method called *Semantic Depth of Field* (SDOF) is proposed, which blurs currently irrelevant objects and thus guides the viewer's attention. This method only requires one additional value for each data point: its relevance. This relevance is then translated into a blur level, which is used for drawing objects. The user has full control over the functions involved in this process. A number of applications is shown to demonstrate the usefulness of SDOF.

Because blur is known to be slow, a method for fast blurring of objects is also presented, which makes it possible to use SDOF in interactive applications.

The results of a user study are also presented, which showed that SDOF is a preattentive feature, i.e., can be perceived within 200 ms, and does not require serial search. SDOF is also not significantly slower than color when used in search tasks; and it does not decrease performance when combined with another feature, as is usually the case.

Zusammenfassung

Ein zentrales Thema in der Informationsvisualisierung, wie auch in anderen Bereichen (etwa Volumens- und Strömungsvisualisierung), ist die Darstellung von Information in einem Kontext, der die Daten leichter erfassbar macht.

Unschärfe ist eine visuelle Eigenschaft, die in der Fotografie seit über 150 Jahren eine wichtige Rolle spielt, dies in der Computergrafik aber so gut wie gar nicht tut. Scharfe Objekte oder Bereiche in einem Foto erregen aber sofort die Aufmerksamkeit des Betrachters / der Betrachterin – der Unterschied zwischen scharfen und unscharfen Objekten eignet sich daher ausgezeichnet, die Aufmerksamkeit auf bestimmte Dinge zu lenken.

In dieser Arbeit wird eine Methode namens *Semantic Depth of Field* (SDOF) vorgeschlagen, die durch Unschärfe weniger wichtige Objekte weniger sichtbar macht, und damit die Aufmerksamkeit des Benutzers lenkt. Diese Methode benötigt nur eine zusätzliche Information pro Datenpunkt: dessen Relevanz. Diese wird in eine Unschärfe übersetzt, die dann zum Zeichnen von Objekten benutzt wird. Der / die BenuterIn hat volle Kontrolle über die Funktionen, die in dieser Übersetzung verwendet werden. Eine Reihe von Applikationen, die SDOF einsetzen, wird besprochen um dessen Nutzen zu demonstrieren.

Weil Unschärfe in der Computergrafik als langsam bekannt ist, wird eine Technik vorgestellt, die es möglich macht, SDOF in interaktiven Applikationen einzusetzen.

Außerdem wird von den Ergebnissen einer Studie berichtet, in der gezeigt werden konnte, dass SDOF ein präattentives Merkmal ist, also innerhalb von 200 ms und ohne serielle Suche wahrgenommen wird. SDOF ist auch nich signifikant langsamer als Farbe in Suchaufgaben, und es verlängert die Suchzeit nicht signifikant, wenn nach der Kombination von SDOF und einem anderen Merkmal gesucht wird – wie das sonst der Fall ist.

Contents

Chapter 1

Introduction

> *Why should we be interested in visualization? Because the human visual system is a pattern seeker of enormous power and subtlety. The eye and the visual cortex of the brain form a massively parallel processor that provides the highest-bandwidth channel into human cognitive centers.*
>
> — Colin Ware [60]

Information visualization produces images from abstract data. Its goal is to make efficient use of our visual system to convey information and to provide the user with the means to gain insight into data. For this purpose, data not only has to be displayed effectively, but it must also be possible to explore and to analyze it, and eventually to present the results of this process to others.

When large amounts of data have to be displayed, it becomes difficult to show all objects with enough detail on the available amount of screen space. Focus+context techniques make it possible to display more data by different means. Most of these methods are *user-driven*, i.e., the user has to select which parts of the display are shown in more detail. There are several ways to do this: by distorting the screen geometry in different ways, by showing different information for the items the user focuses on, etc. – these techniques are discussed in Chapter 2.

But when exploring and ana lysing data, the user often wants to query the data for objects with certain properties; and when presenting results, the user wants to stress certain features of the data, or guide the audience's attention to a particular feature. For this purpose, *data-driven* methods are needed, that point out the results of queries to the user. Different visual cues can be used to do this – color, saturation, etc. –, which have different effects on the display, and are more or less effective. These cues also must allow for very fine-grained control of the display of objects – which is not the case with distortion-oriented methods, for example: they often enlarge objects which are close to relevant ones, but are not relevant themselves (because relevance does not have to coincide with the physical layout of the information).

The idea proposed in this thesis is to blur objects that are of little relevance, and to display relevant objects sharply, thus making them stand out. This method, called *Semantic Depth of Field* (SDOF), was inspired by the depth of field (DOF) effect known from photography. In contrast to DOF, SDOF blurs objects based on their relevance, not their distance from the lens.

Photography and cinematography deal with tasks that are very similar to the ones described above. In an image, the photographer wants to make clear what the subject is, and in what context that subject is put. The viewer (usually) does not have to search for the main parts of a photograph, but is guided through it. This is done with different means, one of which is depth of field (Chapter 3). A lens does not depict all objects equally sharp, but creates blurred depictions for those that are outside the focus plane. A viewer's attention is immediately drawn to the sharp parts of an image – this has been known in photography for about 150 years.

SDOF requires an additional piece of information from the data that is to be visualized: its relevance. Each object or data point is assigned a number from 0 to 1, where 0 means a totally irrelevant object, and 1 stands for a maximally relevant one. How this is done is very specific to the application, and relevance can be used in very different ways: data can be queried by the user and the results shown, objects can be pointed out in a tutorial-like application, and relevance can be used to navigate between layers of information.

Relevance is then translated into a blur level by means of a function that the user can control. It is thus possible to get an overview of the structure of the data by changing blur function properties, or to define a different query by changing the relevance function.

The model behind SDOF, including parameterization and applicability is described in Chapter 4.

A number of applications were developed in the course of this research. Among them is a text viewer which allows the user to search for a keyword, and points out not only the found word, but also the sentence it appears in, by displaying that sentence sharply, and the rest of the text blurred. Other examples include a scatter-plot program which allows the user to differentiate groups of objects through blur; a file system viewer, that can point out files with certain properties through blur; and a chess board that can show different connections between chessmen through selective focus. These applications are described in detail in Chapter 6.

The SDOF model allows the use of existing rendering mechanisms for generating images, but these are generally too slow for interactive applications. A method was therefore developed that can very efficiently blur objects using hardware features present in current low-cost consumer graphics cards, such as texture mapping. This method is described in Chapter 5.

Because of its use in photography, blur promises to be quite effective and intuitive. But it is not necessary to rely on anecdotal evidence: Perceptual psychology investigates the properties and mechanisms of our visual system (among others), and provides means to measure the effectiveness and efficiency of a method. One important property of a visual feature is preattentiveness. Preattentive features are recognized within a very short time (approximately 200 ms) after the exposure to the stimulus, and do not require serial search. They therefore provide a very efficient means of conveying information to the user.

A user study was performed to find out if SDOF was a preattentive feature. We tested 16 participants for different aspects of preattentivity, such as being able to detect and locate a sharp object, or to estimate the percentage of sharp objects among blurred ones. The results clearly showed that SDOF is, in fact, a preattentive feature, and that it can be perceived very quickly. It also turned out that SDOF is not significantly slower than color, and that the combination of blur with orientation does not make perception slower (which a combination usually does). The user study is described in detail and its results are discussed in Chapter 7.

1.1 Conventions and Legal Stuff

Throughout this thesis, the author will generously refer to himself as “we”. This is meant to make reading easier, and to make the thesis sound less presumptuous (because this work was not done strictly on my own without any help from others – but I got help from my supervisors and colleagues).

In this thesis, trademarks, registered names, etc. are not marked as such – it is not my obligation as a scientist to do trademark research. All photographs (except those in Figure 3.10 on page 22) and other images (except those in Chapter 2) in this thesis are copyrighted by the author, and may not be used without explicit permission in any way. The photographs in Figure 3.10 are copyrighted by Dr. Heinrich Tauscher, and are used with his kind permission. All statistical diagrams in Chapter 7 are copyrighted by the Center for Usability Research and Engineering (CURE). The results of the user study presented in Chapter 7 are copyrighted by VRVis Research Center and the Institute of Software Technology and Interactive Systems of the Vienna University of Technology. All methods used in this study are copyrighted by the Center for Usability Research and Engineering (CURE) in Vienna.

Chapter 2

Related Work in Visualization

Whenever large amounts of data are to be investigated, visualization potentially becomes a useful solution to provide insight into user data. Especially for exploration and analysis (but also for presentation) of very large data-sets, visualization not only needs to provide an easy-to-read visual metaphor, but also should enable the user to efficiently navigate the display, allowing for flexible investigation of arbitrary details.

Focus+Context (F+C) techniques enable the user to investigate specific details of the data while at the same time also providing an overview over the embedding of the data under investigation within the entire dataset. But F+C encompasses a number of very different techniques that achieve similar goals in very different ways.

This chapter presents an overview of existing F+C techniques (Section 2.1), as well as a classification of them (Section 2.2). Ways of presenting tree-structured data in a useful way are described in Section 2.3. We also list some existing uses of blur in visualization (Section 2.4) and present a short introduction to perceptual psychology, especially preattentive processing (Section 2.5).

2.1 Focus+Context Techniques

An important factor in visualization is not only to visualize information, but also to provide a context in which the information is put. This is especially important in information visualization, where there are no (or hardly any) natural mappings of data dimensions to spatial dimensions, or to visualization properties.

Once the user has built a mental map [44] of the data, the mappings should only be changed if there is no other way of displaying the requested information. A change of the mapping means that the user must build a new map, which takes time and effort.

But not only in information visualization, but also in other areas, F+C is important and is gaining interest. It is useful, for example, to be able to see the outline of the skin when looking at a volume rendering of the bones and blood vessels in a human hand [16].

An important concept in this regard is the *degree of interest* (DOI) function [11]: it defines which parts are how important to the user right now. This function can be very different depending on the method and the user settings, and can also change during a visualization session.

The function as it is defined by Furnas [11] relies very much on the spatial layout of the information. Its *a posteriori* component is a simple distance metric of objects from the current

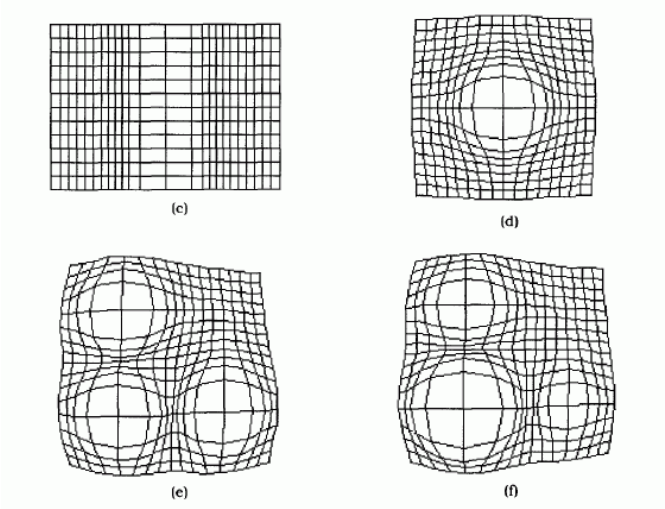


Figure 2.1: An illustration of distortion techniques (Leung and Apperley [32])

point of focus (the *a priori* component deals with structural information that is contained in the data anyway, like directory levels). While this is a useful metaphor for user interaction in a purely distortion-oriented display, it also imposes restrictions on the complexity of the DOI function. Most importantly, it does not allow for a data-driven approach (Section 2.2), where the data is queried and the results pointed out to the user.

2.1.1 Distortion Techniques

The richest and most important class of methods providing F+C are distortion techniques [32]. They mostly work on 2D visualizations, magnifying important parts while compressing less important ones (Figure 2.1). This leads to the screen space being partitioned according to importance, not according to space used by the different objects.

Space-Scale Diagrams [12] are a way of explicitly modeling the different scales that are available in a distorted display, as well as how they can be combined and navigated. The concept of geometric distortion has also been extended to 3D [6], where objects are scaled due to their importance, pushing other (smaller) objects aside. The use of this method does however make use of size as a visual variable impossible, which is not necessarily the case in 2D (2D distortions are easier recognized as such).

Fisheye Views

Fisheye Views [11, 52, 53] are a focus+context metaphor that is also based on a concept from photography. A fisheye lens is an ultra-wideangle lens that has a field of view close to or even above 180°, and that has not been corrected for barrel distortion (i.e., straight lines off-center appear to be bent). In addition to the perspective effects that depict nearer objects larger, a fisheye lens also has an uneven distribution of magnification over the lens, so that objects at the same distance appear larger if they are in the middle, and smaller if nearer to the edge of the lens.

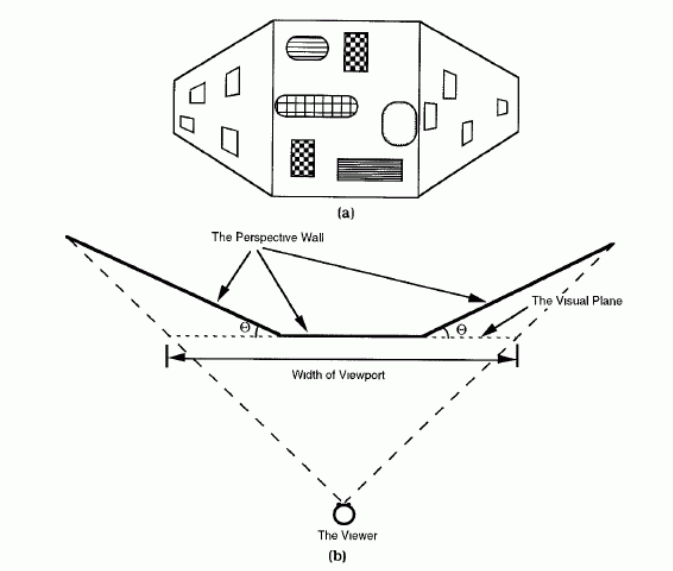


Figure 2.2: An illustration of the perspective wall (by Leung and Apperley [32])

A fisheye view distorts the image in such a way that the region of interest (no matter where on the screen it is) is magnified much more than the other parts of the screen.

Perspective Wall

Distortion techniques are meant to enhance a visualization, especially navigation. This requires them to be fast, so that the user can easily move between parts of the visualization of different magnification levels. One technique that is especially easy to implement on hardware is the Perspective Wall [39]. The visualization is mapped onto a "wall" that consists of three segments: A middle segment that is parallel to the screen and nearest to the viewer, and two segments on either of its sides that are folded back slightly, so that they are distorted by perspective. Because of simple perspective effects, the middle segment appears larger, so that it shows the current focus, while the side segments show the context in less magnification the further away it is from the current focus (Figure 2.2).

A similar idea is the document lens [49], which provides context not only on two, but four sides. A big advantage of this technique is not only that it is fast, but that it also provides a center that is completely undistorted (in contrast to fisheye views, for example), which in this case is necessary for being able to read the text.

Seamless Multi-Level Views

Most distortion techniques only change the size of objects depending on their importance. Multi-Level Views [24] distort the image, but they also include a semantic level, in that they can show different images at different levels of magnification. Thus, a semantic zoom can be integrated by showing images at different levels that differ not only in their magnification, but also in the amount of details that are drawn at all, or that are shown as icons in lower

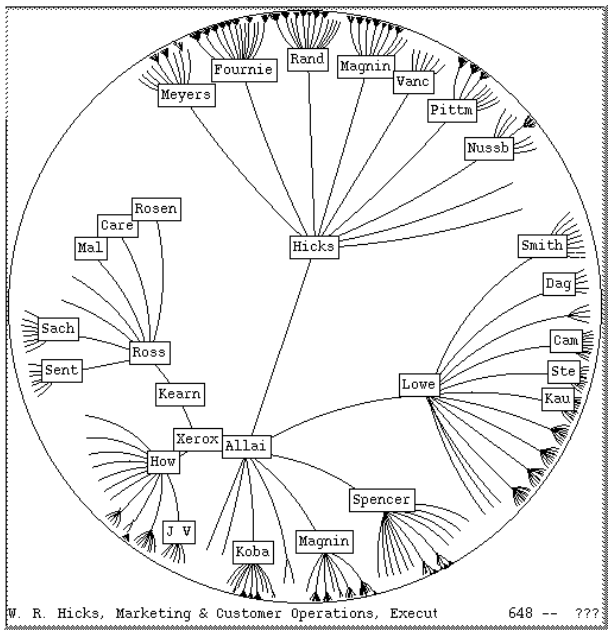


Figure 2.3: A hyperbolic tree (Lamping et al. [30])

magnifications and as photographs (as in the example of landmarks) in higher magnifications.

The different levels are smoothly blended into each other, which can also be accelerated by hardware if done using MIP-mapping. Fog is used as an additional visual cue to enhance depth perception.

Hyperbolic Space

Hyperbolic space [29, 30, 45] is similar to a projection of a hemisphere onto the Euclidean plane (Figure 2.3). The center of the hyperbolic plane is closest to the Euclidean plane, and is least curved. It therefore presents an almost undistorted image. The farther away from the center something is, the smaller it is due to the steeper angle of the "projecting" area – similar to fisheye views. Other than with fisheye views, the projection is constant, but the objects are moved around on the visible area, and are magnified depending on their location.

Stretchable Rubber Sheets

A similar idea are stretchable rubber sheets [54], which allow more general distortions and provide a slightly different metaphor. The user can have several focal points, and the DOI function can be different than with fisheye views or hyperbolic trees.

2.1.2 The Macroscope

The Macroscope [34] does not distort the image, but rather puts several layers of images at different zoom levels over each other. The "lowest" level (or background) of the image is the

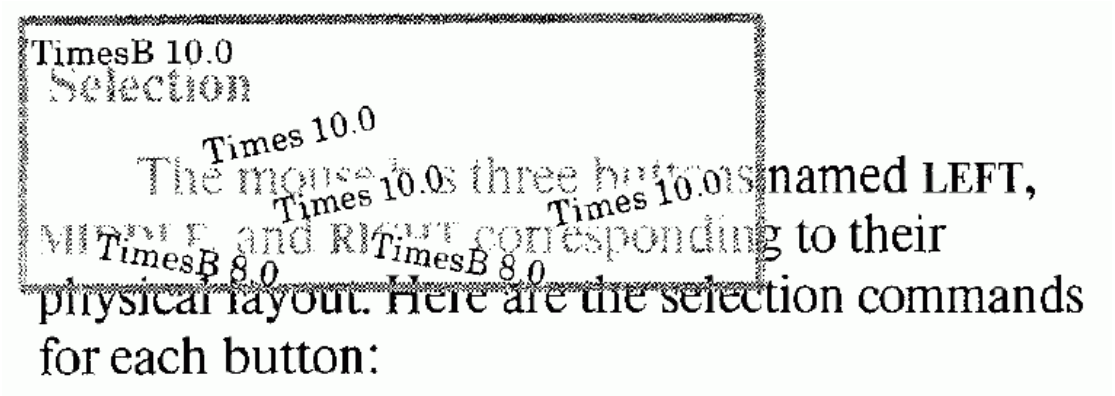


Figure 2.4: A magic lens showing text fonts (Stone et al. [58]).

whole information space (e.g., a map), which serves as the context. A handle (like a rectangle) is displayed on this background to show the part that is magnified. This magnified area is drawn translucently, so that it covers the whole screen (or window), but the background (context) can still be seen. Objects in the magnified view are drawn cruder (i.e., with larger "pixels"), which makes the levels easier to discriminate.

But because of the hard contrasts, the levels in this method are hard to distinguish. It is also rather counter-intuitive to have the more detailed information displayed in a translucent layer (and with larger pixels), when there is a solid background. Therefore, blur is used in the more recent version of the method [35], as mentioned in Section 2.4. Another problem is that details and context from completely different parts of the visualization will be projected onto the same location without any information about whether or not this means that they are in any way related.

2.1.3 Magic Lenses

The Magic Lens [4, 58] displays more detail or different information in a small window without changing the magnification level. The metaphor used here is that of a lens that is moved over a map, for example (which is quite interesting, because only one example in the cited paper shows a different magnification inside the lens, all other examples show purely semantic differences). This reduces clutter, and can be used to display additional information, that would be impossible to show all the time (many different types of information at many points).

With magic lenses, the user has to actively move the focus over the visualization, and is not shown the most relevant data automatically. This is useful for exploration, but for analysis and presentation of data, a more data-driven (rather than user-driven) approach would be useful.

2.1.4 GeoSpace

GeoSpace is a Geographical Information System (GIS) that makes it possible to display crime data, certain cities, or hospitals [38]. This data is displayed in the same context as the whole map, but the relevant parts of the display have a higher color saturation and opacity than

the rest. This leads the viewer's attention to the relevant objects easily without removing context information.

2.2 Classification of F+C Methods

This section presents a classification of the F+C methods discussed above. Usually, only distortion-oriented (or, as we call them, *spatial*) methods are treated as different from the rest. We believe, however, that there are three different classes in F+C visualization that deserve differentiated treatment.

2.2.1 Spatial Methods

The most prominent group of F+C methods are *distortion-oriented* [32] or *spatial methods*. This class encompasses all the techniques listed in Section 2.1.1, like fisheye views, the perspective wall, stretchable rubber sheets, and seamless multi-level views.

Distortion-oriented techniques are usually used in an explicit way, by actively bringing the interesting objects into focus, e.g. by clicking on objects or dragging them around. These methods do not allow for very fine-grained control, because objects that are close to important ones are often enlarged even though they may not be relevant at all. They work best when they are used to reinforce the already existing spatial layout, but not to try out a completely different view on the data.

2.2.2 Dimensional Methods

For smaller numbers of objects that have a lot of data associated with them, a visualization method is useful that shows just a limited number of data dimensions, and allows the user to select which of the objects are to be shown in more detail – we call these *dimensional methods*. The context in this case are not only the other objects, but also the remaining data dimensions. This type of method also shows more detail, but in terms of data dimensions, not screen size.

Dimensional methods are also *user-driven*, because the user has to move the focus to see the additional dimensions. They do allow for fine-grained control, because magic lenses can take on any shape. But they are not very suitable for pointing out information, because they require user interaction to discover the information.

2.2.3 Cue Methods

The third type of focus+context techniques allows the user to select objects in terms of their features, not their spatial relations; usually by assigning a certain visual cue to them – we therefore call these methods *cue methods*. They make it possible to query the data for information which is not immediately visible in the initial visualization, while keeping the original layout, and thus not destroying the user's mental map [44].

Examples for this type of technique are GeoSpace (as discussed above) and the geographical visualization presented in Section 2.4, which uses blur.

In contrast to distortion-oriented techniques and magic lenses, with this type of method, the user first selects the criteria, and then is shown all the objects fulfilling them. So these

methods are *data-driven* rather than *user-driven*, and therefore make it possible for the program to point out information, and guide the user. They also allow very fine-grained control over which objects are pointed out, because they generally do not have (strong side-effects).

According to this classification, SDOF is a cue method.

2.3 Trees

Trees contain a natural hierarchy, much more than many other data structures. They are therefore especially well-suited for being used for F+C. Navigation in a tree can also be hard, which also makes a good visualization important.

2.3.1 Cone Trees and Cam Trees

Cone Trees [50] display trees in 3D, with the children of each node layed out along the base of a cone, the apex of which is the parent node. The root node is placed at the very top, and the size of cones is determined so that the tree best fits the available "room" (which is delimited by screen space). The cones are translucent, so that they structure the data, but do not obstruct the view on nodes that are farther away from the viewer. The tree casts shadows onto "walls" and the "floor" of the display, thus also showing projections of its structure.

When a node is selected, it is moved to the front by simultaneously rotating all tree levels, following a shortest rotational path. This animation is important, because it would otherwise be very hard to understand how the different tree levels came to be moved to the new state.

The labels of nodes are displayed on small "cards" in the cone tree. This can be problematic when the labels are too long: changing the label size or aspect so that the text fits would obstruct parts of the tree in the background. So for this case, the entire tree is drawn from left to right instead of top to bottom. The labels of this tree (which is called a *cam tree*) can extend farther in the horizontal direction.

One problem of cone/cam trees is that they are very inefficient in their use of screen space.

2.3.2 Hyperbolic Trees

A hyperbolic tree [30] is laid out not in Euclidean space, but in a geometry where the fifth Euclidean axiom — that any line has only one parallel that passes through a given point (which does not lie on the first line) — does not hold. If that geometry is projected to 2D Euclidean space (e.g., a computer screen), the appearance is similar to an image taken by a fisheye: the whole space is projected into a circle, with objects near the edge taking up exponentially less space (another paper [21] contains a nice introduction to hyperbolic geometry).

Hyperbolic trees are first drawn with their root in the middle of the circle, and the child nodes spread out all around the root. The different branches get different amounts of space, depending on the number of children they have on different levels (Figure 2.3 on page 7). The focus is changed by moving the nodes in Euclidean space. This does not influence their layout, so this step does not have to be repeated. But the different location on the projected circle yields a different magnification of the node.

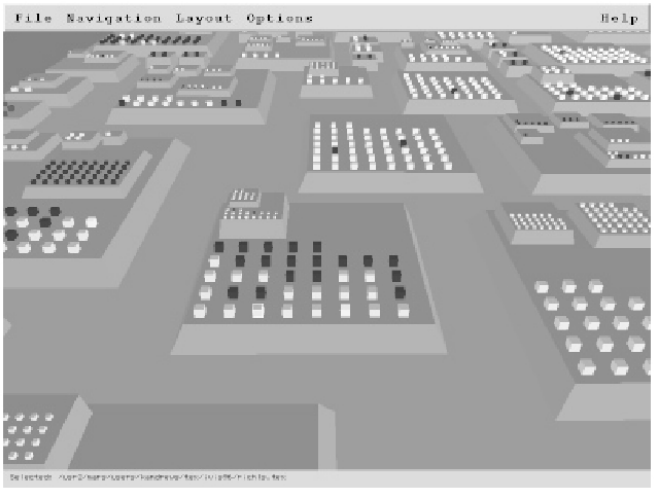


Figure 2.5: Information pyramid (Andrews et al [2])

2.3.3 Treemaps, Space Filling Trees

While cone trees are quite inefficient in their usage of screen space, tree maps [57] fill the entire space with a representation of the tree to depict. The space is partitioned in one direction, which is turned 90 degrees for every tree level. The partition sizes are proportional to the sizes of the subtrees (which are of course simply the sum of their children, so the leaves have to provide some kind of size information; in the example of a file system, this could be the file size).

This method is very space-efficient, but is hard to read, especially for trees that are deeply nested. A notion of recursion levels can be present, but is equally hard to understand for a "deep" tree. Very small objects are hard to see at all, and are certainly hard to label. Therefore, in the sample implementation, labels pop up if the mouse is moved over an area, so that the text does not clutter the image.

A similar method are information pyramids [2], which show the tree levels as stacked pyramid frustrums (Figure 2.5). That view can be navigated by "flying" over the pyramid, and "cropping" it to a certain height, in order to limit the displayed detail.

2.4 Uses of Blur in Visualization

There have been surprisingly few attempts to use DOF or blur in visualization at all; the ones relevant to this work are shortly summarized here.

In a system for the display of time-dependent cardio-vascular data [63], a stereoscopic 3D display is included that is controlled by the viewer's eyes. Like a microscope, only one thin slice through the data appears sharp, all others are blurred and therefore almost invisible. Eye tracking equipment determines what the user is looking at, and that point is brought into focus. This makes it possible to concentrate on one detail without the surrounding structures confusing the viewer. Later work [64] describes "non-linear depth cues", which means displaying structures that currently are of interest (like single organs) in focus, and other

objects out of focus, not based on their distance from the camera, but on their importance. This amounts to a semantic use of depth of field.

The Macroscope [35] is a system for displaying several zoom levels of information in the same display space. For this purpose, the images on all levels are drawn over each other, with the more detailed ones drawn "in front", i.e., drawn over the less magnified layers. The layers' transparency can be changed so that the background (context) can be more or less visible. In order to make the background less distracting, blur is used for the front-most images that show the whole image.

The most interesting existing approach for this work is a display of geographic information [7]. In this system, up to 26 layers of information can be displayed at the same time. Each layer has an interest level associated with it that the user can change. The interest level is a combination of blur and transparency, making less interesting layers more blurred and more transparent at the same time. This work does not seem to have been followed up on recently.

Also interesting in comparison to this work is GeoSpace (Section 2.1.4), which uses a different cue to guide the viewer and present context.

F+C technique, which is a system for visualizing geographical data [38] that uses color saturation to show different types of data for the same geographical area. Different cities, hospitals, pharmacies, etc. can be viewed by "lightening them up" with brighter and more saturated colors than other parts of the image. Here also preattentive processing is exploited for the purpose of fast perception.

A system that is not a visualization system at all, but that is quite interesting, is the Restricted Focus Viewer (RFV) [5]. The RFV is a software solution for eye-tracking in psychological trials, which tends (or at least tended) to be expensive and inaccurate. The participant in the study sees a blurred image, and can move a focus around in which the unblurred image is shown (all movements of the focus are logged of course, and can be precisely reproduced). The blur serves two purposes here: It makes it necessary for the participant to move the focus to the parts of the image he or she wants to look at, and it avoids distractions from other parts of the image.

All the described approaches only used blur in a very limited way. None of them presented a thorough model or linked their work to perceptual psychology, nor showed the vast field of applicability of SDOF.

2.5 Preattentive Processing

Visualization is so effective and useful because it utilizes one of the channels to our brain that have the most bandwidth: our eyes. But even this channel can be used more or less efficiently. It is therefore very important that we know about the different properties of visual cues, and processing of visual information in the brain [13].

The visual system can perform certain tasks without the person having to focus their attention on the objects involved, and in a very short time (typically below 200 ms). This is called preattentive processing, examples for it include detecting filled objects among outlines (Figure 2.6 on the following page, hue differences, orientation, and motion [59].

It is desirable to use preattentive features simply because they are the ones that are perceived fastest, thus optimizing the human-computer interface. Preattentive features also require less concentration and effort, and so are the logical choice for the most important

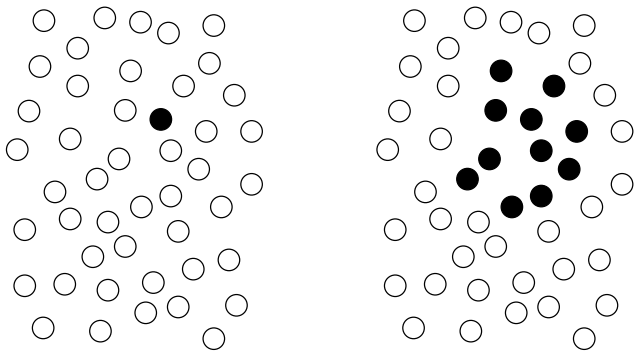


Figure 2.6: An example of the preattentive feature color. The filled circle immediately stands out on the left side, and so does the compound object on the right. The fact that two groups of objects (left and right) are seen at all is due to proximity being another preattentive feature.

data to convey over a visualization. This is also an important factor in animations, where one wants the viewer to be able to easily follow what is going on. Preattentively perceived objects and constellations "pop out" by themselves.

A kind of visualization hierarchy is needed that assigns more important information to preattentive visual features, and less important ones to others. Visualizations for multivariate data have already been developed based on preattentive processing [17, 18, 19].

SDOF is a preattentive method, which is shown in Chapter 7. But even without that proof, it is easy to imagine why this would be the case: depth of field is an intrinsic property of the human eye. Our eyes are not very similar to a camera in many respects, but they are similar at least insofar as both use a lens and an aperture to project an image to a receptive surface (which is where the differences start ...). The eye does not have unlimited depth of field, but we hardly perceive that. Objects that are blurred (because they are too far away from the focus plane or because they are in the peripheral parts of the field of view, where resolution decreases) are either "invisible" (i.e., not perceived as important, at least as long as they do not move) or simply "interpolated", so that we do not perceive the change in sharpness. The effectiveness of DOF in photography also strongly suggests that depth of field is, in fact, preattentively perceived.

Perceptual psychology seems to be getting more popular with researchers in visualization at the moment [19, 21, 60].

Chapter 3

Related Work in Photography

One important difference between photography and drawing is depth of field [42]. It is a natural phenomenon that can be found in any lens system, and even with real pinhole cameras.

SDOF is based on this effect, that is quite well known from photography and cinematography. The basics for this effect as well as its uses in practice are described in this chapter: In Section 3.1, different camera models are discussed; and Section 3.2 describes the use of DOF in photography.

3.1 Camera Models

Camera Models are the basis for all depictions that are calculated by a computer rather than taken with a real camera. A camera model describes the way light rays (or, more common, "sight rays") find their way from the object to the film (or – in the case of sight rays – from the film to the object).

3.1.1 The Pinhole Camera

The traditional camera model in computer graphics is the pinhole camera (Figure 3.2, left). In this model, the film is contained in a light-tight box that has an infinitesimally small hole on one side. Through this hole, light rays can fall in and cause an image to be formed. Because the hole is infinitesimally small, any point on the film can only be hit by a light ray from exactly one direction. This causes a perfectly sharp image with infinite depth of field – at least in theory. A real pinhole camera has a finite hole diameter which causes objects that are extremely close to lose some detail. This effect is hardly noticeable, however.

Pinhole cameras are not just a model but really exist and are used (Figure 3.1 on the next page shows an example). They do show hardly any depth-of-field effects, but due to the finite size of a real hole (and also the fact that hole size and exposure time have to be balanced somehow), sharpness is not very good. This limits the actual use of pinhole cameras to artistic purposes and experiments.

Figure 3.1: A pinhole image taken in Amsterdam. Everything from the rain drops few centimeters from the pinhole to houses hundreds of meters away is equally sharp.

3.1.2 The Thin Lens

A real camera uses a much more complicated (and expensive) thing to form the image than a hole: a lens system[1] (Figure 3.2, right). A lens system normally contains a number of simple lenses, together with the aperture (also called the stop) and possibly the shutter and auto-focus mechanics, etc. forming a very complex device.

The geometric camera model (which, more accurately, is called the *thin lens model* [23, 31]) is described in this section. It consists of a single, infinitesimally thin, simple lens that projects the image onto the film plane (Figure 3.3) – in this model, the curved surface and the physical width of the lens do not play a role. The distances between object and lens, and between lens and image satisfy the following equation, which is called the lens law [31]:

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{f} \tag{3.1}$$

In this formula, u is the distance from the (infinitely thin) lens to the object, and v is the distance from the lens to the image. A lens focuses all rays that are parallel to its axis to a point that is at a certain distance from its center (Figure 3.4 on page 17). This is called the focal length, represented by the letter f.

[1] In English, both the simple glass lens (i.e., a single body of glass that refracts light) and the lens system (the object that is mounted to a camera, usually containing several glass lenses) are called "lens". Therefore, the term "simple lens" will be used here to refer to the former meaning, and "lens system" to refer to the latter, if the meaning is not clear from the context.

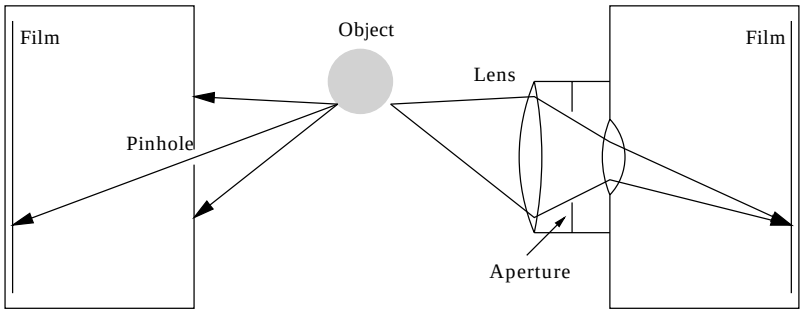


Figure 3.2: Pinhole camera (left) and camera with lens system (right). The rays in the right part are only schematically drawn, and not modeled after their real phyical properties.

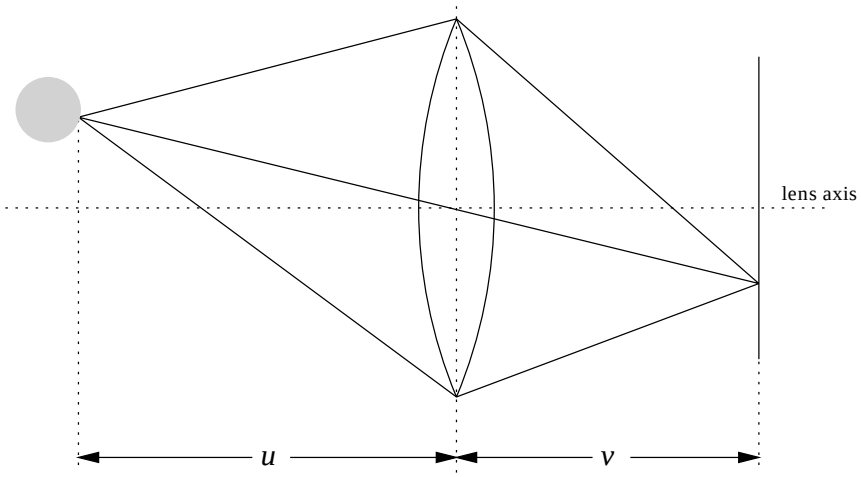


Figure 3.3: The thin lens model, with the object in focus.

If the film plane is not at distance v from the lens, the image gets blurred. In such a case, a point of the object is not projected to a point in the image, but rather to a circle, the so-called *circle of confusion* (CoC). The diameter of this circle, C, can be calculated using similar triangles (Figure 3.5 on page 18):

$$C = D\frac{v}{v - v'} \tag{3.2}$$

D is the diameter of the lens. In practice, one never uses the lens diameter, but the f-stop (or aperture setting) a (also called k in the literature), which defines the ratio between focal length and effective lens diameter: $a = \frac{f}{D}$. Closing the aperture by one stop ("stopping down") makes the lens diameter smaller, causing a more acute angled triangle that also leads to a smaller increase of CoC diameter with distance from the focus plane — and thus, an image with more depth of field.

Any point whose CoC diameter is smaller than a certain maximum (which depends on viewing parameters, see Section 3.2.4) is perceived as in focus. It is possible to calculate a distance that, if the lens is focused at it, will project points at infinity at exactly the maximum

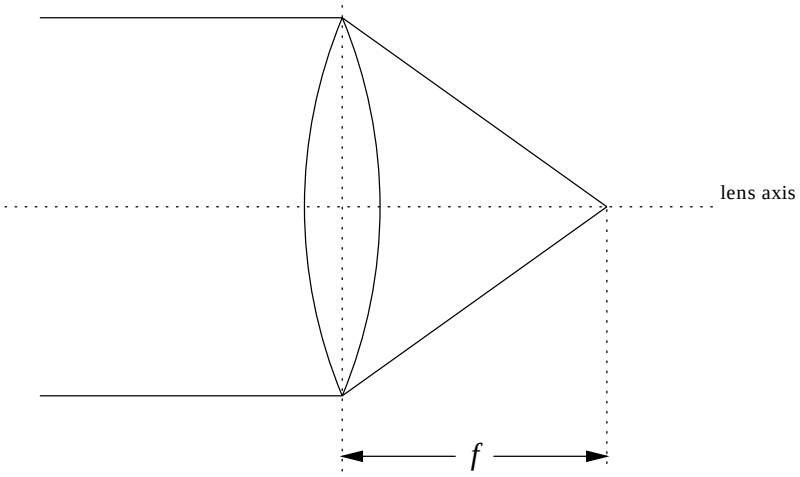


Figure 3.4: Illustration of the focal length f

possible CoC radius, $C_{\max}$. This is called the hyperfocal distance H, which can be calculated using this formula [23]:

$$H = \frac{f^2}{C_{\max} k} \tag{3.3}$$

This is an interesting number, not only because it is useful for landscape photography (where one usually wants maximum depth of field), but also because the near and far planes u_{near} and u_{far} that delimit the area that appears sharp in the image, can be easily expressed by it. If the lens is focused at $u = \frac{H}{x}$, then [23]

$$u_{\text{near}} = \frac{H}{x+1}, u_{\text{far}} = \frac{H}{x-1}$$

3.1.3 Other Camera Models

Complex lens models can be based on physics [31], or on geometry [25] (like the *thick lens model*), physical models that use geometry, or even vector field analysis, etc. For many purposes, however, the simple geometric thin lens model is sufficient.

It does not account for a number of effects, like diffraction, geometric distortion (straight lines are bent if off-center), chromatic and achromatic aberrations, etc. These do not play a role in this thesis, and therefore are ignored.

Diffraction on the aperture does in theory play a role in depth of field (if the aperture is closed below a certain minimum diameter, diffraction causes the CoC to grow again), as do a few other effects. In a real lens, the fact that the aperture is not round but rather a regular polygon with seven or eight vertices, plays a much bigger role (coma effect in night photography, aperture reflections when a bright light source is in the image, etc) than diffraction. Also, the fact that diffraction patterns are only visible for monochromatic light, and blend into each other for light with more than one wavelength [56], makes this effect negligible.

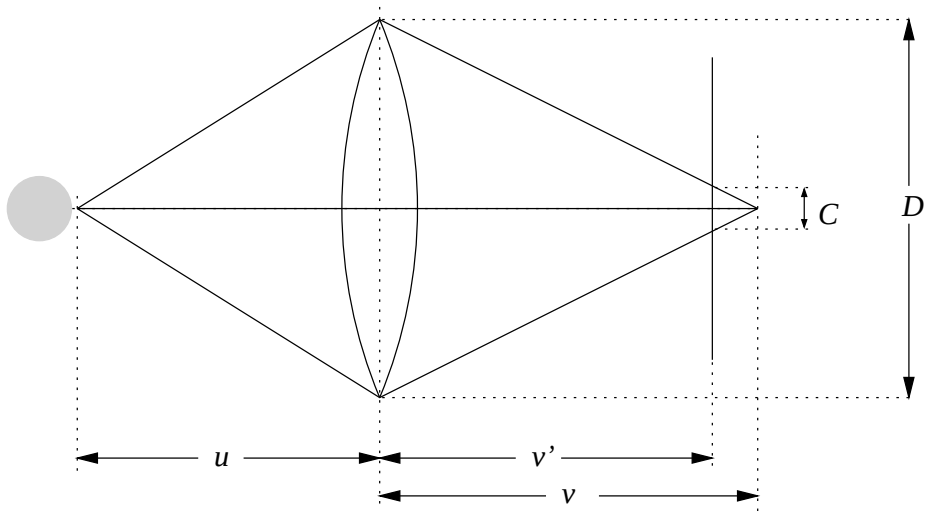


Figure 3.5: The geometric lens model, with the object out of focus.

3.2 Depth-of-Field in Photography

In Photography, blur can have two reasons: motion or optical effects [1]. Motion can be either motion of the camera (camera shake, which one usually tries to avoid) or motion of the object. Motion blur of an object is used to depict the motion, especially when an object is moving very fast. It is also possible to track the object with the camera and thus get the object sharp and everything else blurred through motion (these examples are illustrated in Figure 3.6 on the following page).

Because of limitations in the resolution of the human eye, points up to a certain diameter appear sharp, so that not only one infinitely thin plane appears to be in focus, but all points between the nearest and farthest planes whose points still are projected to circles with a diameter less than or equal to the acceptable circle of confusion. The distance between these two planes is called depth of field (DOF).

3.2.1 Uses of DOF

Depth-of-field is a very important means of directing the viewer's attention in a photograph. Focusing on a person in a crowd, for example, will guide the view to that person immediately, but will also allow the viewer to look at the other persons in the crowd and perceive the surroundings and atmosphere – or context – of the person.

The technique is also used for portraits, where the background is just not important, and is therefore blurred. Using the right parameters, a photographer can create a very homogenous background that will be virtually invisible to the viewers, thus concentrating on the person depicted. Using greater depth of field, it is possible to show the person in her working environment, for example, but without the objects in the environment distracting the viewer. But still, the viewer is able to see and identify the objects.

Depth of field can also be a vehicle of suggesting that there are many similar objects: by focusing on one of them, and showing the others out of focus, the viewer gets the impression

Figure 3.6: Examples of blur: Depth of field (left) and motion blur, with the camera tracking the object (right)

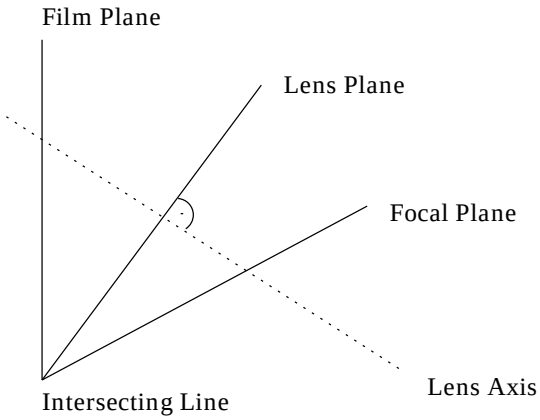


Figure 3.7: The Scheimpflug rule. The film, lens, and focal planes intersect in one line.

that there are many more objects than appear in the frame.

In motion pictures, focus change is a technique that is used quite often, especially in dialogues: two persons are visible, usually near the edges of the frame. The camera focuses on the person who is talking, and when the other person answers, the focus changes, too. Depth of field is also used to guide the person's attention, or to show where a character is looking. A good example is a scene from *The Sixth Sense* (Hollywood Pictures, directed by M. Night Shyamalan), where the main character reads text from a piece of paper, which is shown from the character's point of view with very little depth of field, and where the focus moves between key words as the character reads these parts.

Another photographic technique is double-exposure: By exposing the same negative two or more times, several images are projected onto each other. This can be achieved at exposure time, by not advancing the film between the exposures (this is not possible with all cameras), in the lab by printing two or more negatives on the same piece of paper, or by "sandwiching" two or more slides into the same frame. When double-exposing, one usually only uses one image that has fine detail, like a person, and other images that are either very faint or out of focus so that they are still identifyable, but allow the viewer to make out the main subject without distracting too much. This is a use of the same perceptual mechanisms as depth of field, but on a semantic level, rather than based on physics.

3.2.2 Advanced Uses

With a view camera (large format camera), the focus plane can be tilted against the image plane, thus making it possible to take pictures that have more freedom in what is in focus. When the film plane is tilted against the lens, the plane of focus intersects both the film plane and the plane that is orthogonal to the lens axis in a line (Figure 3.7) — this is called the *Scheimpflug rule* [43].

Physical cameras are still limited to blurring objects depending on their distance, even if that distance is measured along an direction that is not parallel to the lens axis. But tilting the lens already makes it easier to go to a more semantic use of depth of field.

When one is not limited to physically existing cameras, it is possible to go even beyond

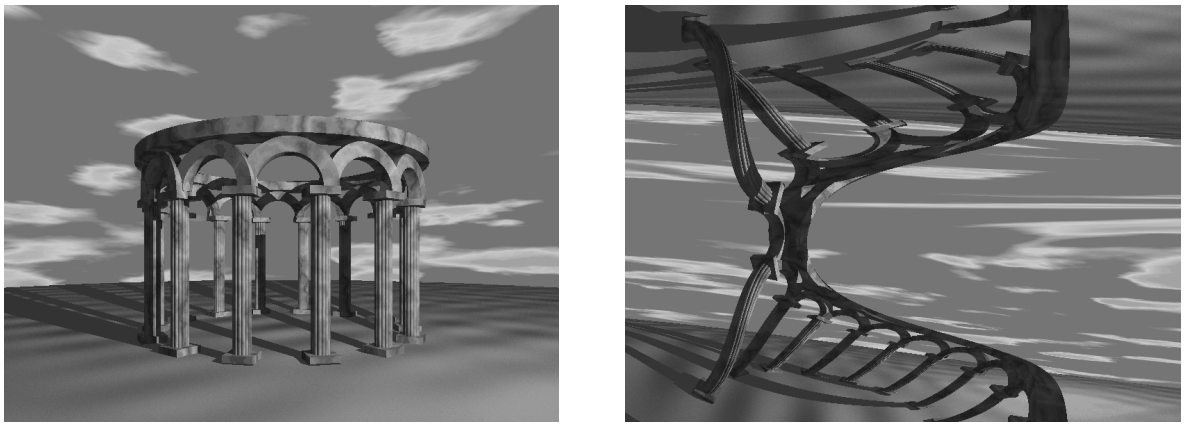


Figure 3.8: An extended camera showing an object from the outside (left) and from the inside and the outside simultaneously (right) (Löffelmann [36]).

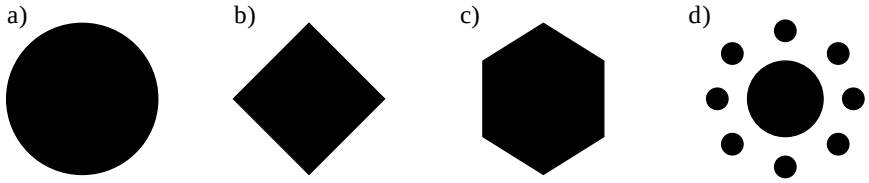


Figure 3.9: Different aperture shapes

a flat film plane or a round lens. Extended cameras [37, 48] can create images that are not possible with a real camera, such as seeing an object from the inside and the outside simultaneously (Figure 3.8). They are harder to understand, but allow the user very fine control over the appearance of objects.

The method proposed in this thesis is an application of a very similar idea.

3.2.3 Aperture Shape and Bokeh

Not just the size of the aperture influences the appearance of an image, but also its shape. The theoretical shape of the aperture is a perfect circle (Figure 3.9a), but that is never achieved in a real lens. Real lenses have polygonal shapes, similar to Figure 3.9b and c; some lenses also have curved elements for forming the aperture, thus creating a shape closer to a circle. Some lenses have even more complex shapes for special uses – like the one in Figure 3.9d, which creates a very soft, unreal image (Figure 3.10 on the next page).

The appearance of blurred parts in an image is called *bokeh* [42], which is the Japanese word for this quality of an image. It is difficult to give general rules about the bokeh of a lens, because it is not only dependent on the aperture shape, but also on other parameters of the lens (which mostly influence if the circle of confusion is really equally bright, or has a light or dark rim, etc.). But it is still generally true, that it is nicer the rounder the aperture is.

One effect where the aperture shape is visble very clearly is lens flare. A bright object creates visible reflections in the lens system, which have the shape of the aperture. This has

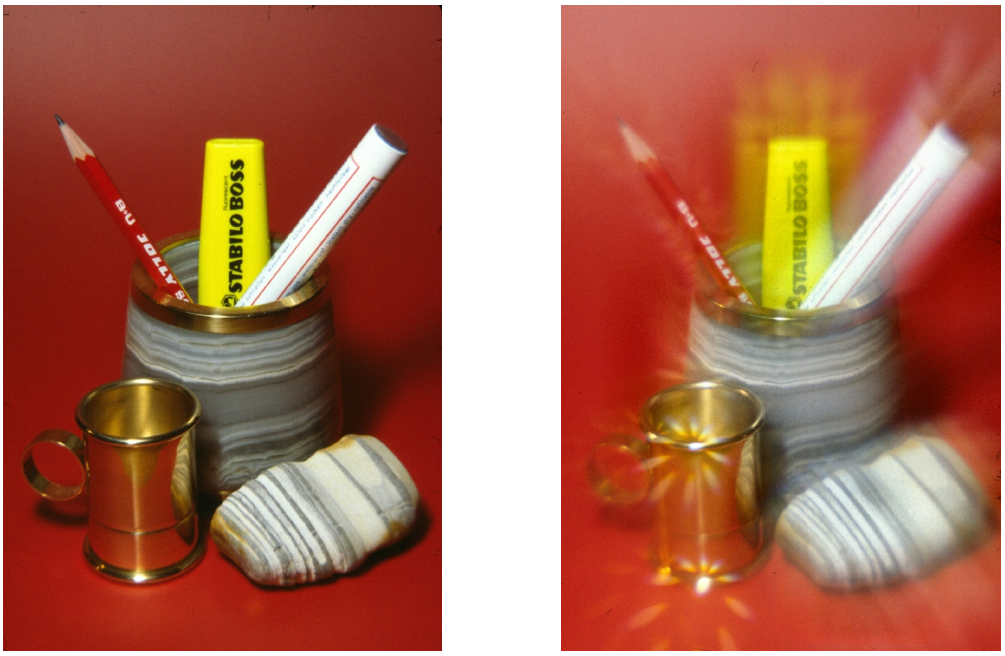


Figure 3.10: Example images for aperture shapes. An image taken with a round stop (left), and one taken with a stop similar to the one depicted in Figure 3.9d (right). Images used with kind permission from Dr. Heinrich Tauscher.

nothing to do with bokeh, but illustrates the effect of the lens shape – this effect gives the viewer quite some information on the lens: the number of lenses (which is half the number of reflections), their spacing, and the aperture shape.

3.2.4 Parameters

Depth of field depends on the perception of sharpness. A point (not in the mathematical sense) is perceived as sharp when it appears smaller than the resolution of the human eye. As a rule of thumb, this is the case for points with a diameter smaller than about $\frac{1}{1000}$ of the distance between the eye and the image.

In photography, the maximum CoC diameter $C_{\max}$ depends on the magnification of the final image (i.e., how much the slide or negative has to be magnified to be printed or projected) and the viewing distance. As a standard, a $C_{\max}$ of 0.03mm on the film is generally considered sufficient for 35mm photography.

In computer graphics, the $C_{\max}$ can be set to 1 Pixel, if the image is to be displayed on a standard screen. For projections, or when larger or higher resolution screens are used, that value would have to be adapted accordingly.

Chapter 4

Semantic Depth of Field

This chapter describes the main contribution of this thesis: Semantic Depth of Field (SDOF). SDOF allows the user to select relevant parts of a visualization that are then pointed out by deemphasizing all the rest through blur.

The building blocks of SDOF are discussed in the following sections, and are summarized in Figure 4.1 on the following page as well as Table 4.1 on page 28 and Table 4.2 on page 30.

4.1 Spatial Arrangement

In information visualization, usually some kind of layout algorithm is used to arrange objects in the visualization space (typically 2D or 3D). The special challenge of information visualization is the fact that data often does not have any inherent structure that naturally translates to a layout. Mapping functions are a very important part of visualization because they determine how well the user can build a mental map that he or she can use to understand and navigate the visualization. Changing the layout often means having to learn a new layout, and thus losing one's ability to navigate easily.

In our model, the spatial mapping function is called place; it translates from the original data domain (DD) to an intermediate two- or three-dimensional visualization space (VS_{2D} or VS_{3D}).

One input SDOF requires from the application in use is a certain spatial arrangement of data items. As we will show in the remainder of this section, both 2D and 3D arrangements are possible with SDOF. In cases where data items inherently exhibit spatial locations anyhow, this part of SDOF becomes trivial.

However, in many cases, especially in information visualization, the data to be depicted does not have any inherent spatial structure and therefore, in principle, there is a significant freedom to place data items in visualization space. In database visualization, for example, usually no inherent spatial sorting of rows and columns exist – how to arrange data items, instead, is an integral part of the visualization procedure. Usually, the spatial arrangement which is chosen by a visualization algorithm tries to reflect the distances between data items with regard to a certain similarity metric. Automatic layout algorithms are used to optimize, e.g., the drawing of the nodes of a graph [9]. Of course, one option of arranging data items can be used to reflect their relevance. An example would be to pan all the visualization space such that the object of interest finally resides right in the center of the projection. More generally, the distance metric in use for automatically laying out the data items can be defined

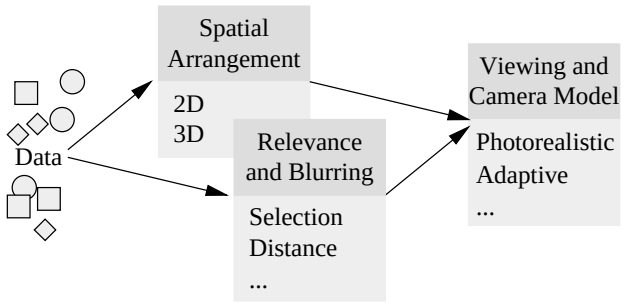


Figure 4.1: SDOF Building Blocks.

such that it reflects the relevance of data items. Consequently, the most relevant data items would be automatically placed near the center of the projection, thus being easily perceived in a quick manner.

However, there is a major problem with this approach: In cases where data items do not have any inherent spatial structure and, therefore, some synthetic layout has to be chosen for visualization, the user needs to form a mental map of the visualization (see Section 2.1). The user needs to learn the visualization layout in order to be able to work with it. As a consequence, it is necessary to avoid major changes to the data layout as much as possible. Therefore, in cases where relevance of data items changes during a visualization session (which is the usual case), other techniques for enhancing objects of interest, like SDOF, are required.

For providing a separable model of building blocks we model the SDOF procedure as described above in two steps:

$$\text{in 2D:} \qquad \mathrm{DD} \xrightarrow{\mathbf{place}_{2\mathrm{D}}} \mathrm{VS}_{2D} \xrightarrow{\mathbf{view}_{2\mathrm{D}}} \mathrm{CC} \tag{4.1}$$

$$\text{in 3D:} \qquad \mathrm{DD} \xrightarrow{\mathbf{place}_{3\mathrm{D}}} \mathrm{VS}_{3D} \xrightarrow{\mathbf{view}_{3\mathrm{D}}} \mathrm{CC} \tag{4.2}$$

where DD denotes the domain where the data items reside; VS is the intermediate visualization space, in which the spatial arrangement takes place; and CC are 3D camera coordinates – the view direction coincides with the positive z-axis, x and y correspond to the orientation of the projection. The function **place** arranges objects either in 2D or 3D visualization space, whereas **view** allows to specify the projection of the visualization layout. Of course, if no SDOF is used, the third dimension of CC is not needed in the 2D case. However, for applying a general camera model later on, it is useful to use a joint notation here.

4.2 Relevance

Independently of the spatial arrangement, the blur level of each object is determined. This is done in two steps: First, each object is assigned a relevance value r by the relevance function rel. The value of r is in the interval $[0; 1]$, where 1 means the object is maximally relevant, and 0 means the object is completely irrelevant.

$$\mathrm{DD} \xrightarrow{\mathsf{rel}} \mathrm{RI}, \qquad \text{“relevance interval” } \mathrm{RI} = [0, 1] \tag{4.3}$$

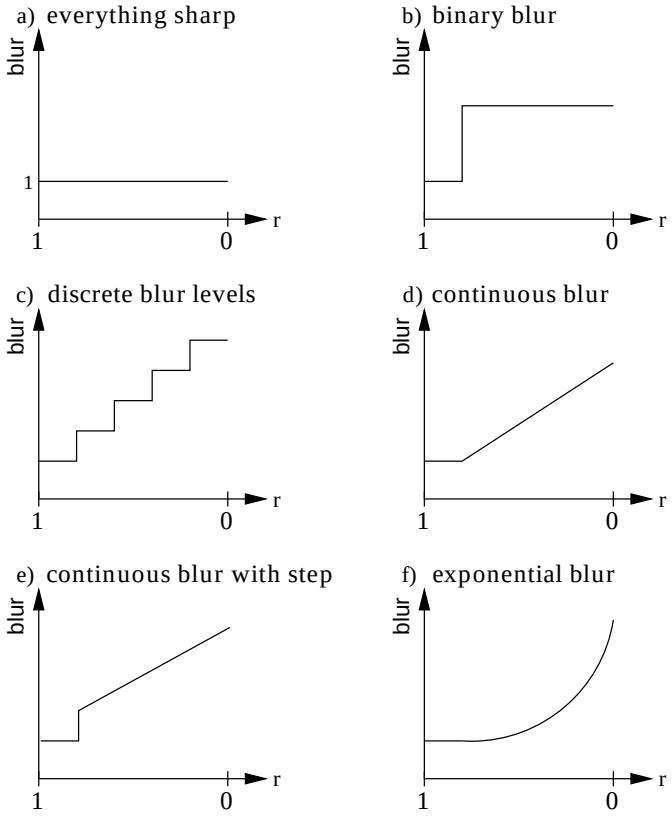


Figure 4.2: Some possible blur functions.

This relevance value is translated into a blur value b through the blur function blur later on.

We distinguish three types of relevance function: binary, discrete, and continuous ones. A binary function only classifies objects of data values into two categories: relevant and irrelevant; a discrete function yields a number of different classes, e.g., 0, 0.5, and 1; while a continuous function uses the whole range between 0 and 1.

The relevance function is application-specific and thus can be very different between applications (see Section 4.6.2 for examples). It can be changed almost continuously during a visualization session to get different views on the data. This is in contrast to the blur function, which will usually not change, but only its parameters will be adjusted.

Different relevance metrics for objects have to be offered by the application, that have to deal with the specific information and tasks the application is made for. Examples for binary relevance measures are the set of chessmen that threaten a specific piece in a chess tutoring system, the layer containing roads in a GIS application, or all incidents related to high blood glucose in a graphical patient record. Continuous functions could express the age of files in a file system viewer, the recent performance of stocks in a stock market browser, or the distance of cities from a specified city in terms of flight hours.

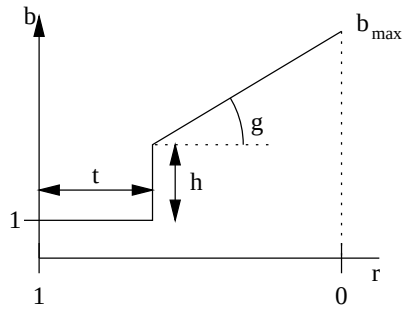


Figure 4.3: The standard blur function.

4.3 Blur

The function blur translates a relevance value into a blur level that can be used for drawing an object.

$$\mathrm{RI} \xrightarrow{\text{blur}} \mathrm{BL}, \qquad \text{blur levels } \mathrm{BL} = [0, \infty) \tag{4.4}$$

For our purposes, blur levels are always measured in pixel units. Therefore, a value of 1 or below denotes a perfectly sharp depiction of an object, any larger value makes the image more and more blurred.

The blur function can theoretically take on any shape (like the relevance function) to best suit the application. Some examples are given in Figure 4.2 on the page before: a constant function, a simple step function, a "staircase" function consisting of several steps, functions that consist of a step and a linear or exponential part, etc.

For practical purposes, we have found the function depicted in Figure 4.3 (which we call the *standard blur function*) to be sufficient for our current applications, however. We also believe that it is easier for users to work with a consistent blur function rather than having to adjust to a different one for every application – even at the cost of slightly less control.

In the standard blur function, the user can specify the threshold value t, the step height h, and the maximum blur diameter $b_{\max}$. The gradient g is then calculated by the application. Some details on the parameters of this function are given in Section 4.6.1)

It would be possible, of course, to map data values directly to blur levels. However, separating the mapping from data to screen space from visualization parameters gives the user more direct and intuitive control (see Figure 4.4 on the next page). This is important for several reasons: a) the relevance mapping can be changed without changing the parameters for blurring (e.g., to show different data dimensions); b) different blur functions can be used for the same relevance mapping (even though the use of the standard blur function appears to be the most useful default); c) the parameters to the blur function can be changed for different output media, like screens, printers, etc. without affecting the relevance mapping, or for looking at different sets of information.

Blur levels can be seen as a one-dimensional extension to visualization space, together acting as an interface between visualization design on the one hand, and rendering on the

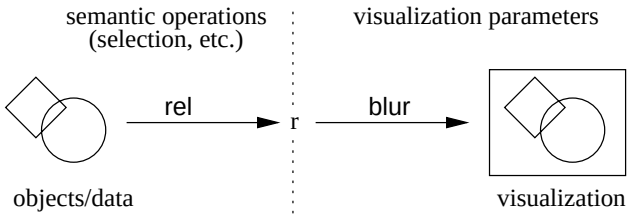


Figure 4.4: Two functions are used to map objects to blur diameters. This makes independent control of semantic and technical visualization parameters possible.

other:

$$\text{in 2D:} \qquad \mathrm{DD} \left\{ \begin{array}{c} \xrightarrow{\mathbf{place}_{2\mathrm{D}}} \\ \xrightarrow{\mathsf{blur} \circ \mathsf{rel}} \end{array} \right\} \widehat{\mathrm{VS}}_{3\mathrm{D}} = \mathrm{VS}_{2D} \times \mathrm{BL} \tag{4.5}$$

$$\text{in 3D:} \qquad \mathrm{DD} \left\{ \begin{array}{c} \xrightarrow{\mathbf{place}_{3\mathrm{D}}} \\ \xrightarrow{\mathsf{blur} \circ \mathsf{rel}} \end{array} \right\} \widehat{\mathrm{VS}}_{4\mathrm{D}} = \mathrm{VS}_{3D} \times \mathrm{BL} \tag{4.6}$$

4.4 Viewing and Camera Models

In order to provide a consistent model, and to embed the idea of SDOF in existing work in computer graphics, we discuss camera models for generating images with SDOF. Depending on whether the visualization space is two- or three-dimensional, different camera models can be used to finally achieve the SDOF effect. The camera provides two functions: $\mathbf{camera}$ projects data values from an intermediate space (where the information was laid out by the $\mathbf{place}$ function) to screen space; and dof, which calculates the blur level of each data item depending on its z coordinate and the z_{focus} value the camera is currently focused at.

In the following, we describe two camera models: a regular photo-realistic camera ($\mathbf{camera}_{\mathrm{p}}$) that can be used in the 2D case; for 3D, we present the *adaptive camera* ($\mathbf{camera}_{\mathrm{a}}$).

4.4.1 2D SDOF and the Photo-realistic Camera

In the 2D case, objects get a third coordinate in addition to their x and y values. This additional z value depends on the intended blur diameter b of the object: If the camera is focused at z_{focus}, an object with intended blur b has to be moved to a distance of z from the lens of the camera (see Figure 4.5 on the following page): where D is the effective lens diameter as defined in the thin lens model [31], and f is the focal length of the lens in use.

The above equations apply to camera models such as distribution ray tracing [8], linear post-filtering [47], etc. (see Section 5.1 for a discussion).

In the 2D case (2D spatial arrangement), vectors from $\widehat{\mathrm{VS}}_{3\mathrm{D}}$ are three-dimensional, i.e., containing a 2D location and a blur level b. As a next step in the SDOF procedure, a viewing

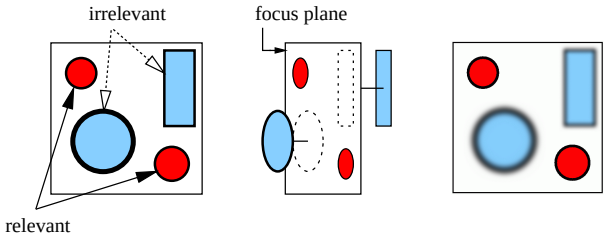


Figure 4.5: The photo-realistic camera and 2D SDOF. Objects from the original scene (left) are moved in the Z direction (middle) to create the right blur. The final image (right) is then rendered with a photo-realistic camera.

$\mathbf{data}[i]$	$\xrightarrow{\mathbf{place}_{2D}}$ $\xrightarrow[\text{rel}]{}$ r $\xrightarrow{\text{blur}}$	$\begin{pmatrix}\hat{x}\\ \hat{y}\\ b\end{pmatrix}$	$\xrightarrow{\mathbf{view}_{2D}}$ $\xrightarrow{\text{dof}_p^{-1}}$	$\begin{pmatrix}x\\ y\\ z\end{pmatrix}$	$\xrightarrow{\mathbf{camera}_p}$	$\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix}$
$\text{DD} \subseteq \text{R}^n$	$\text{RI} = [0,1]$	$\widehat{\text{VS}}_{3D}$		CC	(z_{focus} fixed)	IS

Table 4.1: 2D SDOF: from data items to blurred images

transformation is needed as well as a mapping to convert blur levels into depth values:

$$\widehat{\text{VS}}_{3D} \left\{ \begin{array}{c} \xrightarrow{\mathbf{view}_{2D}} \\ \xrightarrow{\text{dof}_p^{-1}} \end{array} \right\} \text{CC} \xrightarrow{\mathbf{camera}_p} \text{IS} \tag{4.7}$$

where CC are 3D camera coordinates as described in Section 4.1, ready to be rendered into image space (IS) using the standard photo-realistic camera (thin lens model, z_{focus} fixed). Note that this case directly corresponds to the basic idea, illustrated in Figure 4.5.

The function dof_p^{-1}, which is responsible for the computation of the third camera coordinate (z) from blur levels, is the inverse of dof_p, which calculates a blur level b from the distance of an object from the lens, z. As an additional parameter, it needs the distance at which the camera is currently focused, z_{focus}. If the thin lens model is used (see Equation 3.1 on page 15), dof_p and dof_p^{-1} are defined as follows:

$$b = \text{dof}_p(z, z_{\text{focus}}) = \left| D \frac{f(z_{\text{focus}} - z)}{z_{\text{focus}}(z - f)} \right| \tag{4.8}$$

$$z = \text{dof}_p^{-1}(b, z_{\text{focus}}) = \frac{D + b}{\frac{D}{z_{\text{focus}}} + \frac{b}{f}} \tag{4.9}$$

$$D = \frac{f}{a} \tag{4.10}$$

where D is the effective lens diameter (expressed as a ratio of the focal length f and the

aperture number a, see Equation 4.10 on the page before) and f is the focal length of the lens in use.

The above equations apply to camera models such as the ones described in Chapter 3, and can be used with some of the implementations presented in Chapter 5.

If the camera uses perspective projection, the objects will have to be scaled and moved (if off-center) to compensate for depth effects that are not desirable in this case.

All the steps necessary for 2D SDOF that were presented in this section are summarized in Table 4.1 on the preceding page.

In contrast to this "backwards compatible" model, fast and efficient implementations of SDOF are described in chapter 5, which use entirely different methods.

4.4.2 3D SDOF and the Adaptive Camera

In the 3D case, of course, it is not possible to directly map blur levels to depth values, because the spatial arrangement of data items already contains a third dimension. However, using a simple extension of the photo-realistic camera, it is possible to also handle the 3D case.

In this case, $\widehat{\mathrm{VS}}_{4\mathrm{D}}$-vectors (containing the three space coordinates and the blur level) are mapped into extended camera coordinates:

$$\widehat{\mathrm{VS}}_{4\mathrm{D}} \left\{ \begin{array}{c} \xrightarrow{\mathbf{view}_{3\mathrm{D}}} \\ \xrightarrow{\mathsf{dof}_{\mathrm{a}}^{-1}} \end{array} \right\} \widehat{\mathrm{CC}} = \mathrm{CC} \times \mathrm{FV} \xrightarrow{\mathbf{camera}_{\mathrm{a}}} \mathrm{IS}, \tag{4.11}$$

where CC is extended by one additional dimension of focusing values (z_{focus}, FV), becoming four-dimensional.

The adaptive camera is a modification of a photo-realistic camera that can change its focus for every object point to be rendered. This is easily possible with object-order rendering, but can also be achieved when rendering in image order. In contrast to the photo-realistic camera, the adaptive camera can render SDOF in 2D and 3D scenes. The photo-realistic camera is, in fact, a special case of the adaptive camera (which simply stays focused at the same distance for the whole image).

Function $\mathsf{dof}_{\mathrm{a}}$ is defined like $\mathsf{dof}_{\mathrm{p}}$ in Equation 4.8 on the page before. Different to the 2D case, now the inversion of $\mathsf{dof}_{\mathrm{a}}$ must be resolved for z_{focus}-values:

$$b = \mathsf{dof}_{\mathrm{a}}(z, z_{\mathrm{focus}}) = \mathsf{dof}_{\mathrm{p}}(z, z_{\mathrm{focus}}) \tag{4.12}$$

$$z_{\mathrm{focus}} = \mathsf{dof}_{\mathrm{a}}^{-1}(b, z) = \frac{D}{\frac{D+b}{z} - \frac{b}{f}} \tag{4.13}$$

An example for an adaptive camera is splatting [22, 62], which is a volume rendering technique, but which also can be used for information visualization. By changing the size of the splat kernel depending on the b value of a data point, SDOF can be implemented easily.

Another possibility is to use pre-blurred billboards [41]. Objects are rendered into memory, the images are then blurred and mapped onto polygons in the scene.

Table 4.2 on the following page summarizes all necessary steps for 3D SDOF (and also makes it easier to compare with 2D).

$\mathbf{data}[i]$	$\xrightarrow{\mathbf{place}_{3D}}$ $\xrightarrow{\text{rel}}\ r\ \xrightarrow{\text{blur}}$	$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ b \end{pmatrix}$	$\xrightarrow{\mathbf{view}_{3D}}$ $\xrightarrow{\text{dof}_a^{-1}}$	$\begin{pmatrix} x \\ y \\ z \\ z_{\text{focus}} \end{pmatrix}$	$\xrightarrow{\mathbf{camera}_a}$ $\begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix}$
$\text{DD} \subseteq \text{R}^n$	$\text{RI} = [0, 1]$	$\widehat{\text{VS}}_{4D}$		$\widehat{\text{CC}}$	IS

Table 4.2: All steps necessary for visualizing data values $\mathbf{data}[i]$ with 3D SDOF.

4.5 Properties and Applicability

This section discusses some high-level properties of SDOF, how it can be principally applied, and what challenges it brings with it.

4.5.1 Properties

When discussing the properties of SDOF, we must first look at the features of blur.

- Blur removes high spatial frequencies and reduces contrast. These two effects cannot be separated when blurring, and they together create a very familiar impression of blur. How much each one of these effects contributes, and if they would be effective on their own, needs further investigation.
- By removing high spatial frequencies, blur also removes fine details, which can be problematic when using icons, for example. But this "price" that has to be paid is rather low considering that only irrelevant objects are blurred (see below).
- The borders of blurred objects become fuzzy and transparent, and if an object is small compared to the blur level, the whole object can become transparent. This helps reduce the effect of occlusions, and also reduces the distracting effects of large but irrelevant objects in the display.
- Blurred objects appear smaller, even though they get larger. The blurred border of objects extends beyond the original one, but because that border is now semi-transparent, the underlying object appears to be smaller. If this is desirable depends on the use. When size is chosen as a visual cue, it is certainly a disadvantage; but when size does not play a role, this effect makes the object even less interesting to the viewer, and is therefore desirable.

SDOF, being yet another F+C highlighting technique, has the following properties that make it an addition to the visualization toolbox:

- SDOF does not distort geometry. It is therefore usable when shapes of objects or parts of objects (glyphs) and positions are used as visual parameters. We also believe that it is easier to recognize blurred icons than distorted ones.

- SDOF does not alter colors. If color is used to convey meaning, or the visualization is to be used by color-blind people, SDOF can be used instead of color highlighting. This also means that SDOF is independent of color, and can therefore be used when only gray-scale is available (e.g., printed images).
- SDOF changes the irrelevant objects, rather than the relevant ones. It is therefore useful whenever the relevant objects contain a lot of information whose perception might be impeded by changes.

4.5.2 Applicability

SDOF requires concrete queries to the data (which can be simple, but have to be formulated nonetheless), and is therefore useful for analyzing and presenting data.

SDOF can serve as an additional aid to guide the viewer's attention, together with brighter colors, etc., or as a completely separate dimension of data display (this last idea has turned out to be only of limited use, see the user study in Chapter 7). Because blur is very naturally associated with importance (even more than color), we do not believe that it is suitable for true multi-variate data visualization. It can, however, add another dimension for a short time, when the displayed data is to be queried.

Blurring needs space, so when a lot of very small objects are depicted, it is only of little use. The application can deal with this problem by drawing the objects in such an order that sharp objects are drawn on top of blurred ones. But this can introduce artifacts, where parts of the display appear sharp only because of the contrast between sharp objects and the background.

4.5.3 Challenges

SDOF images depend on the output device (similar to tone mapping [40], for example). The reason for this is that blur is not an absolute measure, but depends on the viewing angle that the image covers – this is also the reason why small images look sharper than larger ones: the circles of confusion are not visible in the smaller version, or at least to a smaller extent. We use a calibration step at program startup (and suggest a default value) to account for this problem (see Section 4.6.1).

This problem also has to be taken into account when presenting information to a large audience, where people are at very different distances from the screen the image is shown on.

Images that contain SDOF effects are also problematic when lossy compression is used (like MPEG, JPEG, etc.). In this case, artifacts can be introduced that create a high contrast in a blurred area, and thus distracting the user. But SDOF is most useful in interactive applications, so this problem should play no big role in practice.

Another factor that must be considered is that people do not like looking at blurred parts of a display to gather information (this is one of the results of our user study, see Chapter 7). So care must be taken that only truly irrelevant information is blurred, and that the user has a quick way of changing the display back to a completely sharp depiction – we call this the *auto focus*.

One problem we have not yet been able to investigate quite enough is the interplay between blur from SDOF and other depth cues in 3D applications. While SDOF appeared to

be quite intuitive and very visually effective in the 3D sPGNViewer (Section 6.5), several people have expressed their concern about this (potential) problem.

4.6 Parameterization

Parameterization of SDOF consists of two parts: Adaptation to current viewing parameters and user interaction to change the relevance mapping.

4.6.1 Output Adaptation

We ask the user to select two blur levels on program startup: a) the minimal blur level that can be easily distinguished from a sharp depiction – this value translates to the step height h in Figure 4.3 on page 26; b) the minimal difference in blur that can be distinguished – this value can be used to calculate g, if the smallest difference between any two r values is given. Because this is generally not the case, the blur function is adapted for every image after examining the r values of all objects. These values can vary with the use of the generated image (printing out, projecting onto a wall, etc.), the use of different screens, etc.

In our user study (Chapter 7) have shown values between blur levels of 11 to 15 pixels as easily distinguishable from sharp objects, even when images were only shown for a very short time. The performance for 7 pixels was significantly worse. These values can be used as defaults for working on a 1024x768 17" screen at normal viewing distance (about 80cm).

4.6.2 User Interaction

Interaction is a key part of SDOF. Blurred objects are unnatural, and it is therefore important for the user to be able to change the relevance mapping and blur function quickly, and to return to a depiction that shows all objects in focus.

Depending on the application, there are different usage patterns. In many applications, it is useful to be able to point at an object and say "Show me all objects that are older than this", "Show me all chessmen that cover this piece", or "Show me the cities weighed by their railway distance from this city".

Another way is to select values independently of objects: "Show me all threatened chess pieces of my color", "Show me all files that were changed today", or "Show me all current patients weighed by their need for drug xyz".

Another feature of interaction is the auto focus (described in Section 4.5.3). This function can either be initiated by the user, or performed automatically after a certain time by the program. Especially if it is triggered by the program, the auto focus must be animated, so that the user can follow what is happening, rather than suddenly being presented with an entirely different image.

But also other transitions between different displays always have to be animated to enable the user to follow the change and immediately see which objects are relevant in the new display. This is also another reason for separating r and b (see Section 4.2): The animation is done between the old and the new b values, rather than the r values. This is because the blur function can contain discontinuities that can lead to jumps between blur levels of objects, and are therefore undesirable.

4.7 Usage Types and UI Metaphors

There are many different ways in which SDOF can be used in an application. SDOF can be an enhancing factor or be the basis of completely new interface metaphors (Section 4.7.2). This section describes some ideas for such usage types. Application examples are described in Chapter 6 in more detail.

4.7.1 2D SDOF

Displaying information in 2D, it is possible to use blur to show a selection or any other distance function. This is quite different from DOF as used in photography (which only exists if there is a third dimension), but still appears to be very effective and useful.

One example is a 2D chess board (Section 6.5) that shows which pieces threaten a specific piece, or how well a particular square on the board is guarded by other pieces of the same color. This program could be extended to a tutoring system, where SDOF is used to point out certain constellations or possible moves.

It is also possible to blur text that only serves as context in a keyword search, for example (Section 6.1). The keyword is marked with color, for example, the sentence containing it is displayed sharply, and the rest of the page is blurred. It is thus easier to find the immediate context of the word and to judge whether this hit is useful.

Instead of text, blurring objects in a scatter-plot program is also possible (Section 6.3). This makes it possible to analyze the data by querying it for different properties.

Another example is a window manager that blurs all screen areas that are currently not used. Thus, a window showing the output of a program, or tracking a communication channel could be blurred so that it does not interfere with other work currently done by the user. If new messages arrive, the scrolling would be visible. It would also be possible to bring the window to focus on such a case, and thus directly guide the user's attention to it, without popping up a window or otherwise interfering with the user's current task.

4.7.2 Layered 2D SDOF

When several 2D layers of information are put on top of each other, it is possible to provide the user with an intuitive way of choosing how much and which information to display crisply.

Independent Layers

A number of 2D depictions at the same level of detail are put one above the other, like floor plans in architecture drawn on tracing paper. The layers are translucent (the level of translucency can be changed), so that the other layers can be seen through the ones on top. Any subset of these layers can be rendered out of focus, so that the information on the in-focus layers becomes much more dominant.

For a user interface that is based on layering inspired by transparent paper [3], the idea of wiggling the different layers is presented, so that the different layers can be discriminated. We believe that SDOF can show this effect much more effectively, and also provide many other means of interaction.

Stacked Layers

Using the same topology as for independent layers, this mode is closer related to the photographic metaphor. Only a subset of neighboring layers is in focus here, all other layers are blurred according to their distance from the nearest in-focus layer. The rate of blurring can be selected by the user. Additionally, it is possible to limit the number of blurred layers that are shown on top of the first in-focus layer. Any layers beyond this threshold are simply not displayed. This avoids too much clutter when navigating between layers. Moving the focus through these layers is very similar to changing the focus on a camera, and is therefore relatively easy to understand.

An example for layered 2D SDOF is a map viewer that allows many layers of geographic information (streets, mountains, rivers, telephone lines, weather data, population data, etc.) to be displayed at the same time. The user can select what information is shown and how sharp, thus focusing on certain information while at the same time getting the context of the depiction.

Hierarchical Layers

If data from several levels (semantic or from different magnification levels) is put together into one image, the different levels can be included more easily if there is a smooth focus change between layers with different detail levels while the image is zoomed (Figure 4.6 on the following page). It is thus possible to immediately see the correspondences between objects on different layers, without having to switch back and forth between them. Magnification and blur can change simultaneously or independently of each other, depending on the user's needs. Unlike the Macroscope (see Section 2.4), the different magnification levels are not drawn in the same size over each other, but maintain their relative sizes.

Thick Layers

Layers as used above are 2D layers with no extension into the third dimension. A thick layer combines many thin layers into one, which then behaves like one thin layer in some respects. But with a thick layer, it is possible to navigate within its sub-layers (independently of the other layers), and also to tilt the focus plane, and so get an image that combines information from different sub-layers, thus showing a development over time, for example.

4.7.3 3D SDOF

The above uses of SDOF were only special cases of 3D SDOF. In 3D, it is possible to shift the focus between any objects, similar to the 2D case. Together with other interactions like navigation, pan, zoom, rotation, the user can have the system point to the objects that meet certain criteria, etc.

If more complex objects are displayed (like 3D glyphs), it is possible to assign a different b value to each vertex, thus making it possible to distinguish between different selection functions on the same object.

Any hierarchical data could be displayed using nested, translucent boxes; these boxes are blurred when the user focuses on their contents, while the contained objects are blurred (and the boxes drawn crisply) when the higher hierarchy level is of interest. This also solves

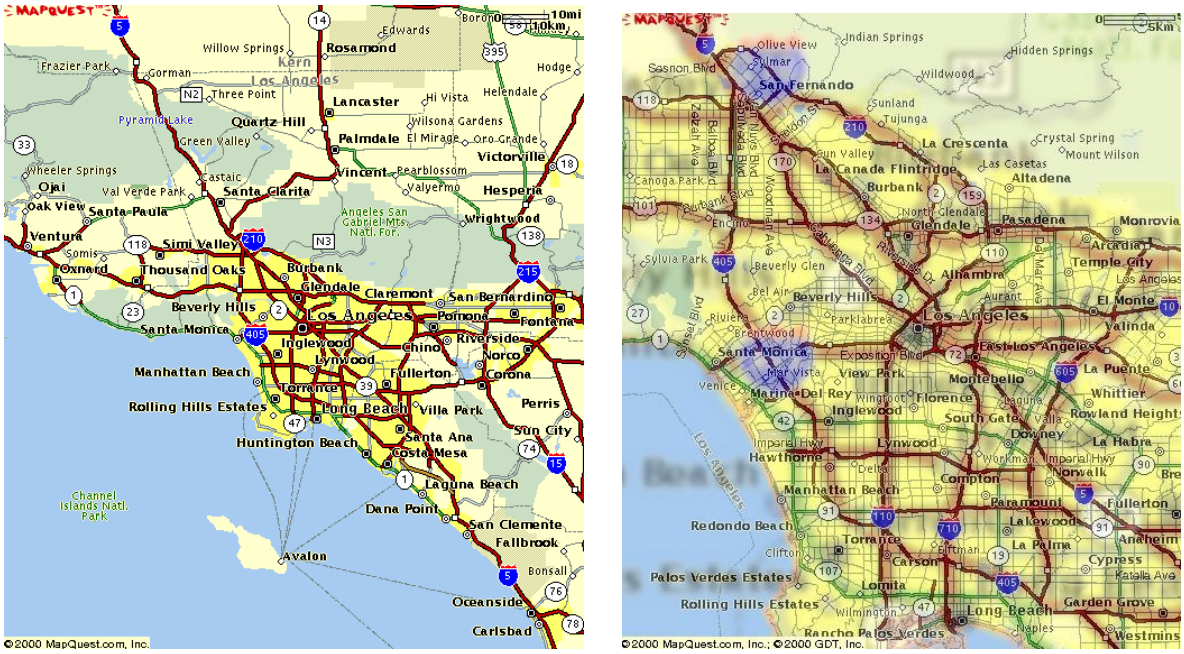


Figure 4.6: Hierarchical Layers. Overview (left), zoomed in view with blurred context (right).

the problem of how to draw these boxes so that they can be distinguished: because their contents are shown (but not in detail), they are recognizable.

Another example of 3D SDOF is a 3D file system viewer that displays files and directories as objects in 3D space, and that allows searches and selections, the results of which are displayed by blurring all objects that do not match the criteria. When looking for objects that have a certain age, it displays the difference from the searched age with continuous blur levels.

The chessboard example cited above in 2D can of course also be extended to 3D (Section 6.5).

Chapter 5

Implementation

For SDOF to be used as a method in real applications, rendering has to be fast enough to get interactive frame rates. This is a very important aspect, because users will not accept a program where they have to wait for the image to appear – especially not in information visualization. SDOF also is a quite dynamic feature that needs to be able to change quickly and to do so with intermediate steps (animation).

The low speed of existing blur methods seems to be one of the main reasons why blur is so little used in computer graphics and especially in visualization. Section 5.1 describes existing methods for blurring, which are mostly tied to depth-of-field quite strongly. Our own much faster implementation – appropriately named *FastSDOF* – is described in Section 5.2.2.

5.1 Depth-of-Field Methods

The simplest requirement that SDOF should fulfill is that there is a fast implementation of it. This is important simply because a method that will only run reasonably on very high-end hardware is unlikely to be widely accepted as a tool for visualization.

This section lists existing methods for rendering depth of field in computer graphics. They are compared to each other, and their applicability is analyzed in Section 5.1.9.

5.1.1 Distribution Ray Tracing

Maybe the most physically motivated approach (if the geometric model (Section 3.1.2) is considered to be sufficient) is to use ray tracing, and to sample many rays across the lens for every pixel and calculate the average of their colors. The lens diameter can be changed to get images with more or less depth of field. This is what is done in distribution ray tracing [8] (originally called distributed ray tracing). Using this technique, rays can also be distributed over an area in the image, as well as over time, making effects like motion blur and penumbras possible.

This method is of course only possible when the system uses ray tracing anyway, and it imposes a cost that depends on scene complexity and screen resolution. It has the advantage of being very close to the model, so that it is possible to directly use parameters from real lens system, and compare the results with photographs, or maybe include them in photographs or movie scenes. A principal disadvantage of ray tracing is its high computational cost, which makes this method extremely slow.

5.1.2 Linear Postfiltering

Given an image taken by a pinhole camera and depth information for every pixel (a so-called Z-Buffer), it is possible to simulate DOF. This can be done by calculating the CoC for every pixel depending on its depth, and accumulating all the CoCs in a new image, yielding an image with DOF [47].

This model does not take occlusion into account, which causes "bleeding" of out-of-focus points in the background into in-focus objects in the foreground. This effect is not existent in a real lens system[1], due to occlusion of the blurred parts of the more distant objects by the closer object.

The parameters for this method can be closely modeled after a real lens, but are independent of the parameters used in generating the image. This leads to a model that is somewhat less integrated than with distribution ray tracing. On the other hand, the complexity of this algorithm only depends on the image size, not the scene complexity. The scene does have an influence on the complexity, of course, because many pixels with large circles of confusion will cause more pixel accesses (for distributing the pixel's contribution).

5.1.3 Ray Distribution Buffer

Similar to linear postfiltering, the ray distribution buffer (RDB) [56] is a postprocessing method. It deals with the occlusion problem by introducing a Z-buffered color buffer of sub-pixels for every pixel. It calculates the CoC for every pixel in the original image, and distributes its color information to the RDBs of all the pixels that are inside that CoC. The cell of the RDB that the contribution is added to depends on the incident angle that the ray has on this pixel (this incident angle is simply calculated from the relative position of the two pixels and the z value). The RDB is Z-buffered, so that only nearer values can overwrite exiting ones for the RDB entry of the same pixel. The final image is calculated by first finding any occluding objects that would determine the color of the pixel (thus avoiding bleeding), and then averaging all the RDB cells of each pixel.

Also similar to postfiltering, the complexity of this method depends mainly on the image size, but also slightly on scene complexity.

5.1.4 Accumulation Buffer

The accumulation buffer was proposed in 1990 [15], and has in the meantime become a standard part of OpenGL. It allows the accumulation of several images, with a weight associated with each of them. Thus, it is possible to draw the same scene from different points sampled across the lens (similar to distribution ray tracing, Section 5.1.1), and combine the images. Depending on the number of images accumulated, this can yield very good quality images with depth of field.

The accumulation buffer is now also used in computer games for DOF effects. Because games require certain minimum frame rates, only a small number of images (typically eight) are accumulated. This leads to images that look very bad if objects are very far away from the focus plane and thus move for large distances so that the individual copies of the objects can be seen.

[1] However, depending on the contrast situation, this effect does exist to some extent in real lenses, but is probably due to reflections within the lens system.

The complexity of this method depends on the scene complexity as well as on the image size. In consumer graphics cards, the accumulation buffer is rather slow, so for larger scenes, alpha blending [46] can be used (which usually has lower accuracy, but is sufficient for many purposes).

5.1.5 Splatting

Depth of field can be understood as a 2D convolution operation, with the convolution kernel containing the contribution of the current pixel to all other pixels within its circle of confusion –this is in fact the same operation that can also be achieved with the accumulation buffer, if the scene is not redrawn from its 3D representation, but only the rendered image is moved and accumulated.

Thus, depth of field can be implemented by means of Splatting [61, 62]. Splatting is a direct volume rendering technique that works by convolving a kernel with a slice through a volume data set. This operation is needed to reconstruct the continuous volume from the sampled data (voxels). The size and shape of the footprint (the projected kernel) has a great influence on the appearance and usefulness of the generated image. By making the kernel dependent on the distance of the slice (which is always parallel to the plane which the image is projected onto) from the focus plane, it would be easy to implement depth of field with splatting.

There are two general directions in which slices can be drawn: back-to-front and front-to-back (as seen from the projection plane). In back-to-front splatting, a nearer slice is drawn over a slice that is farther away, thus creating correct visibility. Parts of the image can also be translucent (so that the inside of the object can be seen), which means that the nearer slice will be accumulated with the existing color values.

In front-to-back splatting, a new value is added only when the slice in front of it is still translucent enough so that it can be seen. This makes it possible to terminate the rendering process early (at full opacity).

Splatting is of course dependent on scene complexity, but also depends on the type of rendering and on the properties of the footprint.

5.1.6 $2\frac{1}{2}$D Method

A method similar to simple postfiltering is to draw objects into several planes depending on their Z position, blurring the planes separately, and then compositing them [55].

This method is of course limited to depictions where the background can be easily separated from the objects in the foreground. Objects that extend far in the Z direction are hard to depict this way. Another problem is that the partial translucency of blurred objects in the foreground is hard to implement, and has not been done in the cited paper. It is also difficult to control the blurring, which is very dependent on how well the objects fit into different planes and the distance between them.

5.1.7 Light Fields

Light fields [14, 33, 51] are four-dimensional representations of the illumination situation in a scene. A light field describes the amount of reflected light for every point in 3D space and in every direction (which is 2D, because only the direction in two planes — one parallel and

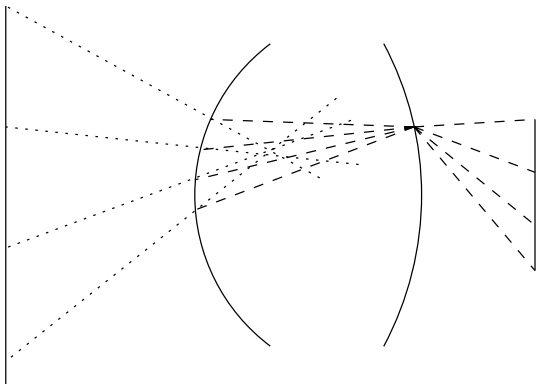


Figure 5.1: Illustration of thick lens model in light field rendering (based on Heidrich et al. [20]).

one perpendicular to the surface — plays a role), yielding a 5D description of the scene. If the model is restricted to solid objects, it can be reduced to four dimensions by projecting their surfaces onto cubes, and folding the sides of those cubes so that they end up in one plane.

Images of a scene can be calculated by projecting slices through a light field onto the screen. This can be done using texture mapping hardware and is therfore rather efficient.

It is also possible to do this while at the same time simulating lens effects like depth of field and distortions (barrel distortion, aberration) [20]. This method uses a thick lens model (i.e., one with a lens that consists of two sides with different curvature), which can model some simple lens systems as used at the beginning of the 20th century. First, a projection is done from the scene onto sample points on the lens using the outside lens curvature, and then projected to the image plane using the inner curvature (see Figure 5.1).

The lens surfaces are subdivided so as to minimize the error introduced by assuming a simple central projection for every point on the lens (which is generally not the case).

Like distribution ray tracing, this method is very much oriented on the physical basics of the simulated effects, and can therefore easily be parameterized for creating images that fit together with real images from a camera.

5.1.8 Importance Ordering

In the above methods, DOF is calculated for a single image, and therefore has to be recalculated for every image in a series, even when there is high correlation between the frames, like in a movie scene.

Therefore, importance ordering [10] remembers the Z position, circle of confusion diameter, and color of all pixels from the previous frame. Changes in these parameters are then classified into different importance classes, so that the most important pixels (the ones that have changed most) can be updated first. Thus, it is possible to interrupt the algorithm at any time and still get a usable image that already contains the most noticeable effects, or to set a limit on the amount of pixels updated for every frame.

The DOF method used is linear postfiltering (Section 5.1.2), with the additional rule that the circle of confusion of a pixel can only influence pixels with a Z value larger than that

Methods	Properties					
	Close to Physical Model	Postprocessing	Independence from Scene Complexity	No Bleeding	Correct Transparency	Speed
Linear Postfiltering		•	•			+
Ray Distribution Buffer		•	•	•		+
Importance Ordering		•	•	•		+
Light Fields	•			•		-
Accumulation Buffer	•			•	•	++
Distribution Ray Tracing	•			•	•	--
Splatting				•	•	-
$2\frac{1}{2}D$ Method				•		+

Table 5.1: Comparison of the properties of different visualization methods.

pixel (i.e., no bleeding into objects that are closer to the viewer). This seems to effectively lead to the same results as the ray distribution buffer (Section 5.1.3).

This is not really a new method (even though the adaptation of linear postfiltering is interesting), but it is a very useful extension to existing postprocessing methods if used for more than just static images.

5.1.9 Comparison of Methods and Discussion

The following list describes the key features that DOF methods should have. Table 5.1 shows which methods have which features.

Close to Physical Model. Is the rendering process closely modeled after the lens model? This usually leads to images that look more realistic, and is therefore generally useful.

Postprocessing. The method is applied to the finished picture as generated with a pinhole camera model.

Independence from Scene Complexity. Every method is at least somewhat dependent on scene complexity, because the 3D structure of the scene determines how many large and small circles of confusion have to be rendered. But if the method is independent of the 3D description of the scene, different materials, etc., this columns is ticked.

No Bleeding. If an object that is far away and out of focus can contribute to a pixel that is closer to the camera and in focus, the closer object appears to be translucent. This is called bleeding, and not a desired property.

Correct Transparency. An object that is out of focus becomes somewhat transparent. But only the outermost part does, where some rays can get past it and “see” the object behind.

Speed. Methods should be fast, of course. It is very hard to compare the speed of the published methods (which were published over a relatively long time, and often did not contain detailed information on their actual speed), so only a relatively crude idea of the relative speeds can be given in Table 5.1.

Most of the methods described can be used for Semantic Depth of Field. All of them

require some changes for this, because SDOF has different requirements than conventional DOF. Interestingly, a close connection to a physical model now is a disadvantage.

Considering its integration with OpenGL, the accumulation buffer would be the "canonical" solution for SDOF. By not drawing the entire scene from several viewpoints, but only the objects that are displayed as blurred, a very realistic image is attained with a rather simple method. Unfortunately, the accumulation buffer seems to be implemented in software only on consumer graphics cards[2].

Another method that is suitable for SDOF is linear postfiltering. Instead of using the real depth values of pixels to determine the size of the circle of confusion, a "Z2-Buffer" could be introduced that contains values that are based on the importance of objects. The big disadvantage of this method is that it is quite slow and cannot be implemented using hardware at all (except for rendering the original image, which would then have to be transferred to main memory and back to the frame buffer, which is slow).

Splatting would also be a possibility, but seems to be even slower than linear postfiltering, and is not suited well for information visualization. Depth of field (semantic or not) seems to be rather easy to add to an existing splatting program, however.

The $2\frac{1}{2}$D Method is in principle usable for SDOF. Rather than grouping objects by their position, they could be grouped by meaning, rendered into different planes, and blurred independently. But this does not answer the question *how* to blur them efficiently. One approach would be the accumulation buffer, which is however problematic for the reasons stated above. Another problem would be visibility of objects drawn into the same layer, but coming from very different depth positions.

Distribution ray tracing and light field rendering are ruled out by the fact that they are too tightly bound to the physical model of depth of field. It is certainly possible to modify these methods such that SDOF is supported, but this would amount to almost entirely redesigning (or working around) the original methods.

5.2 Fast Methods

In the end, none of the above methods was used, because they were too slow for interactive applications. Two new techniques were designed and implemented that make use of modern hardware, and thus are much faster than any purely software-based ones.

5.2.1 Polygonal SDOF

When a polygon with only one color is blurred, something peculiar happens: only its edges change, the interior stays the same. It is therefore possible to create the impression of a blurred polygon by simply modifying its edges. A very similar thing happens with a line: It simply gets spread out perpendicular to its direction. The distribution of the values is a convolution of the blur kernel (which, ideally, is round) with a constant function along the line (Figure 5.2 on the next page).

The result of the convolution is a structure that is identical for most of the length of the line, and only differs at its ends (Figure 5.3 on the following page). It is therefore possible to create a texture that contains one "end" of a line with a given blur diameter that is then drawn on a rectangle instead of that line – creating the impression of a blurred line. For this

[2] Thanks to Markus Hadwiger for pointing this out to me.

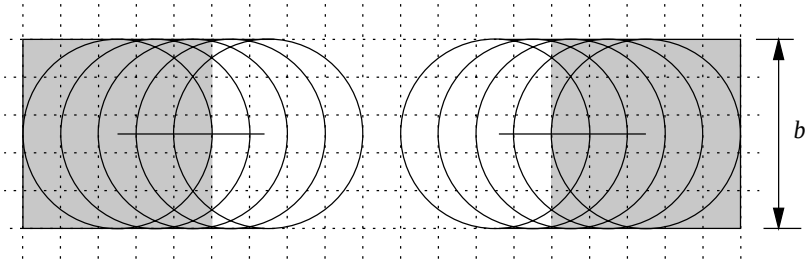


Figure 5.2: The convolution of a round blur kernel along a line.

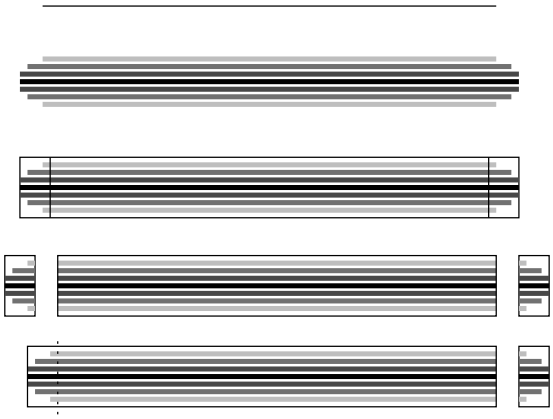


Figure 5.3: The textures used for the polygonal method.

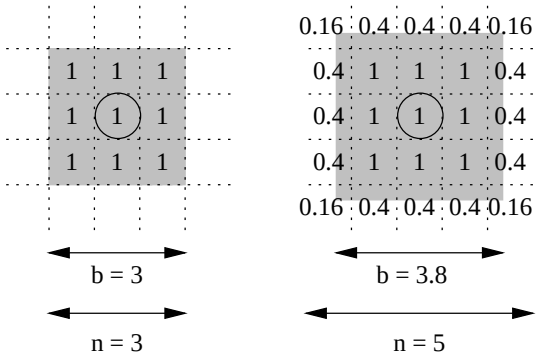


Figure 5.4: The Box Filter (left), and the generalized box filter for arbitrary sizes (right). It also shows the difference between the blur level b and the filter kernel size n.

purpose, two rectangles have to be drawn: One that covers one end and the "inner" part of the line; and a second one that contains the other end. The texture for the first rectangle is clamped at its edge, so that the constant inner part of the line is repeated along it.

This texture only has to be created once for every blur diameter, because it can be treated as a luminance / alpha texture that is modulated with the color of the line. This means very little storage space is needed, and if the blur kernels are precomputed, they can be made as complex as necessary, regardless of the computational cost.

For polygons, a "half line" is drawn along each edge, that only contains the outer part of the texture (split along the line). If a texture is mapped onto the polygon, that texture has to be blurred by another means (e.g., software filtering or FastSDOF).

The polygonal method is used in sav (Section 6.4 on page 50) for displaying blurred time annotation glyphs, and in the regular sMapViewer (Section 6.6 on page 52).

5.2.2 FastSDOF

Blur can be understood as a convolution operation of the image with a blur kernel[3]. In photography, this blur kernel ideally is round, but usually is a regular polygon with six to eight vertices, due to the shape of the aperture.

The more common (because of its simplicity) type of blur kernel in computer science is the box filter (Figure 5.4, left). It has the big advantage of being separable [41], which reduces its computational cost from $O(n^2)$ to $O(2n)$, where n is the filter width. It can also be generalized quite easily to arbitrary sizes (Figure 5.4, right) other than just odd integers. Even though b can take on any real value, the filter width n is always an odd integer: $n = 2\left\lfloor \frac{b}{2} \right\rfloor + 1$

In terms of bokeh (Section 3.2.3 on page 21), this does not give a very nice image. But it has turned out to be suitable for our purposes, and we were prepared to sacrifice some image quality for speed that would enable us to write applications fast enough for interactive work. Figure 5.5 on page 46 shows an object blurred with this method at different blur levels. Rendering time was below 1 ms for all levels.

[3]Such convolution operations are supported only on very high-end graphics hardware, but the goal for fastSDOF was a method that would work on standard PC graphics hardware

Using graphics hardware is different from a software implementation of a filter in that it does not sum up the color values of surrounding pixels for every single pixel. Rather, it adds the whole image to the frame buffer in one step by drawing it onto a textured polygon (this is done by blending with a special blend function). When the image is drawn in different positions (with one pixel distance between the images), several image pixels are added to the same frame buffer pixel. Because of the limited accuracy of the frame buffer (typically eight bits per color component), this can only be done for small values of n (we have found $n \leq 4$ to yield acceptable images).

For larger blur diameters, we use a two-step approach. First, we sum up four images into the frame buffer, with their color values scaled so that the sum uses the entire eight bits. We then transfer this image to texture memory (this is a fast operation) and use this auxiliary sum as the operand for further calculations. The auxiliary sum already contains the information from four addition steps, so when summing them up further, only one quarter of the addition steps is needed. Because all the values in the box filter (except for the border, which is treated separately) are equal, all auxiliary sums are equal – they are only displaced. This means, that the auxiliary sum only needs to be computed once (as well as another auxiliary for the borders). Summing up auxiliary sums is therefore not only more accurate, it is also faster.

For a box filter of size n, $n-1$ additions are usually needed in one direction. When an auxiliary sum of k values is used as the operand, only $k + \lfloor \frac{n}{k} \rfloor + l$ (with $l < k$) additions are needed – this is faster for any $n > k + 2$, where k is usually a small number ($k = 4$, in our case).

Instead of adding one copy of the image, m-fold multi-texturing allows adding m copies at the same time. For small m (usually 2 or 3 on current low-cost graphics cards), this means an improvement of almost a factor m.

For blur diameters larger than 20 pixels, we first scale the image to one quarter of its size, then blur with half the diameter, and then scale it back ("quarter method"). The current implementation is fast enough to suffer no noticeable speed difference for blur levels between 2 and 40. In both scaling operations, bilinear filtering is used, which adds to the blur effect quite nicely. This makes larger blur diameters faster than smaller ones, because the gain from only blurring one quarter of the image is larger than the additional scaling steps.

The resulting blurred image is then copied into a texture which is later applied to a billboard. When using the "quarter-method", the second scaling step is done when finally applying the image to the billboard, so there is very little overhead involved. Because the image never leaves the graphics card (provided the number of blurred objects is not too large in relation to available memory on the card), this is very efficient. Table 5.2 gives some numbers on the actual performance of the applications that exist so far.

Using the described method, it is possible to run applications – like the ones presented in Chapter 6 on page 47 – at interactive frame rates (at leat 5 frames per second) on cheap consumer graphics hardware. This number is likely to increase with some further optimizations as well as the use of multi-texturing (which is supported by more and more consumer graphics cards).

5.2.3 Comparison

Even though FastSDOF appears to be the more useful of the fast methods, the polygonal method also has its merits. When texture memory is scarce, or when a lot of polygonal,

Program	Size (Pixels)	Percentage Blurred	Framerate	Figure
LesSDOF	500x400	12%	167	Figure 6.1 on page 48
LesSDOF	500x400	92%	143	Figure 6.2 on page 48
sfsv	380x480	25.5%	31	Figure 6.3 on page 49
sscatter	600x600	37.2%	23	Figure 6.4 on page 51
sscatter	600x600	46.5%	19	Figure 6.5 on page 51

Table 5.2: Performance figures for the applications shown in this paper.

un-textured objects are to be drawn, the polygonal method is clearly superior. It also allows for more sophisticated blur kernels (e.g., a perfectly round one) and can thus create a nicer bokeh in the images. The polygonal method is also useful when implementing SDOF in a system where only a scene graph can be generated, but there is no access to the underlying mechanisms, so no textures can be copied, etc.

For most other applications, however, FastSDOF is clearly superior. When textured or very complex objects are drawn, it does not create such a high load on the geometry parts of the graphics card, but rather relies on raw texturing power. Texture mapping is very important in computer games, and is therefore likely to gain even more speed with new hardware. FastSDOF also makes it possible to use small increments in blur (below one pixel), which is not possible with the polygonal method – at least without full-screen anti-aliasing, which is very costly.

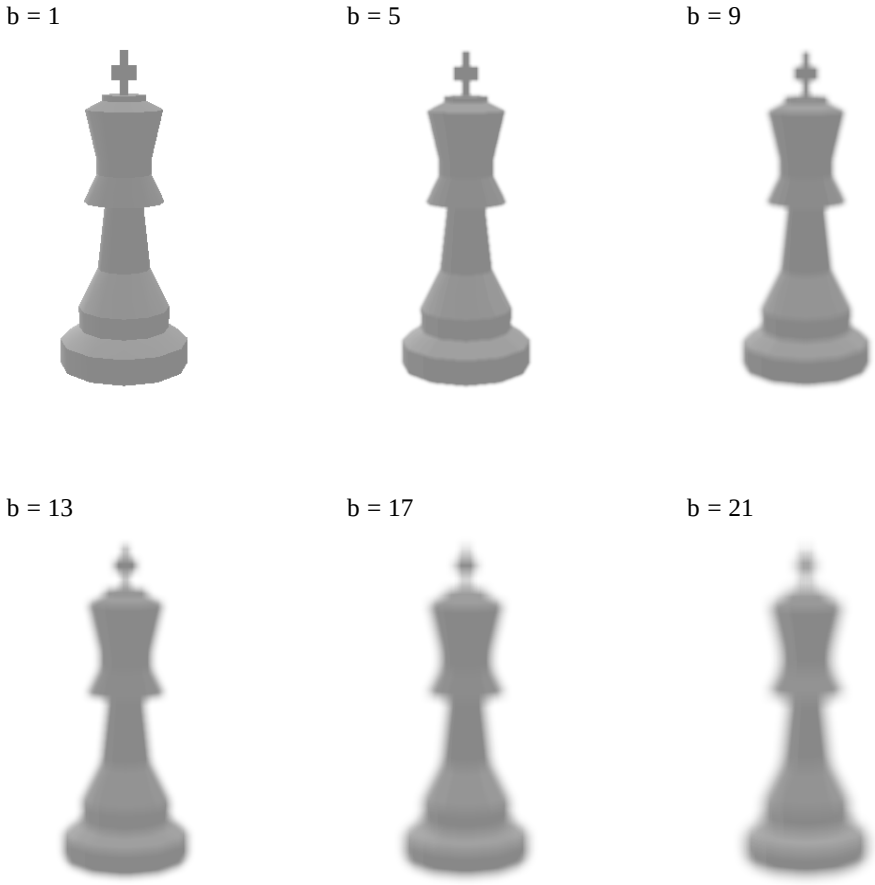


Figure 5.5: A gallery of blurred objects with different blur levels.

Chapter 6

Applications

This chapter presents several applications that use SDOF to solve real problems. The descriptions are structured into the problem, application overview, and SDOF aspects.

6.1 LesSDOF: Text Display and Keyword Search

Displaying text and being able to search for keywords is a very common application. It is not only used in word processors, but also in web browsers, help systems, etc. Most applications only show the found keyword (e.g. using color), but leave it to the user to understand the context. It would be helpful to be shown the whole sentence in order to more quickly make use of the search result.

6.1.1 The Application

LesSDOF [28] displays a text file and allows the user to scroll through it, much like the Unix program *less* (which is a pun on the older and less powerful program *more*). When scrolling a whole page, a few lines are displayed on both pages as context. Theses lines are slightly blurred so that user understands that this is context information (Figure 6.1 on the next page). When searching for a keyword, the found words are displayed with their fore- and background colors exchanged, and therefore clearly stand out. The sentence in which they appear is displayed sharply, while the rest of the page is blurred. It is possible to jump between hits, and so move the focused sentence, or to show all context sentences in focus (Figure 6.2 on the following page). Other hits for the keyword, which are visible on the page but not the current focus, are visible despite the blur – the inverse display of the keyword is easy to see.

6.1.2 SDOF Aspects

This application only uses a binary relevance classification. A text string is either a keyword or it is not, a line of text is either new or overlapping from the last page. Blur and other cues (like inverse display of the keywords) are used to reinforce each other in the case of the current keyword, and as orthogonal dimensions for other keywords. This example does not use any color, and is still very effective in guiding the viewer's attention.

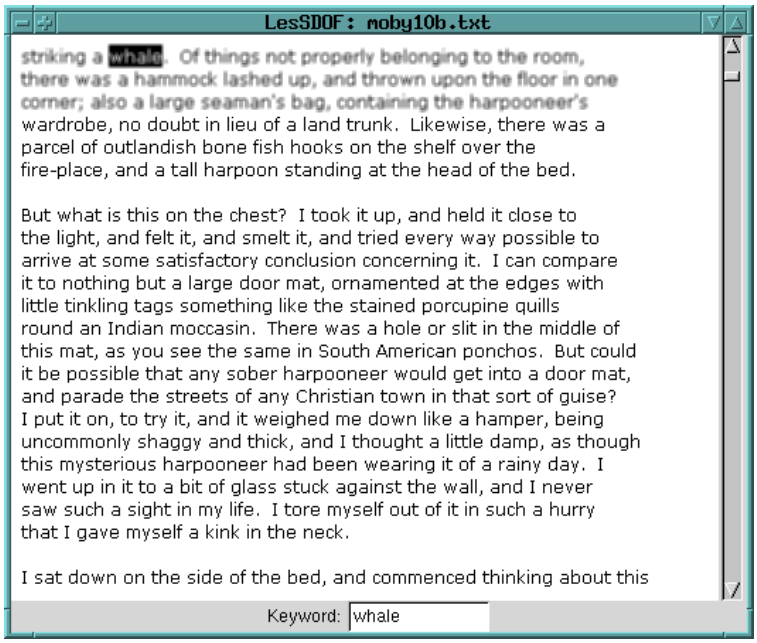


Figure 6.1: Scrolling in LesSDOF: The top three lines are context from the last page, and therefore blurred – but still readable.

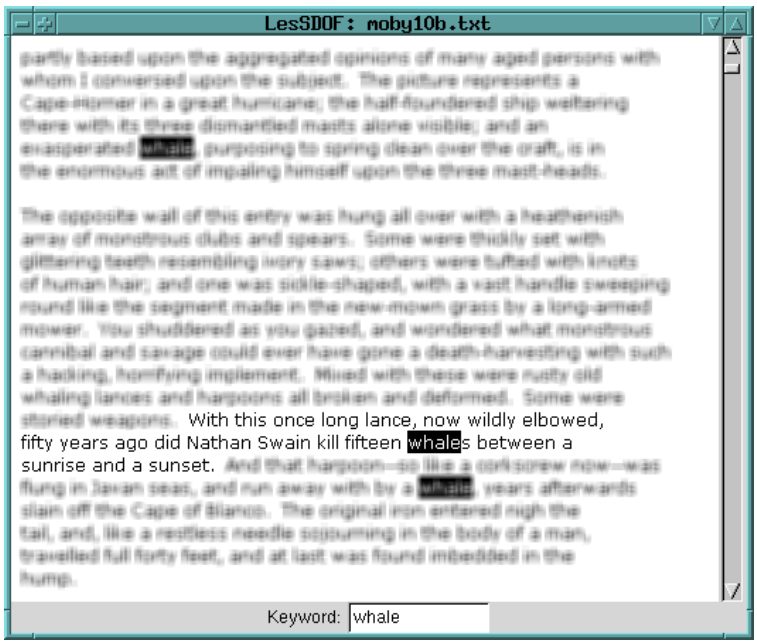


Figure 6.2: Finding a keyword in LesSDOF: There are three hits on this page, with the focus currently on the middle one. The sentence around the keyword is clearly visible, while the rest of the context is blurred.

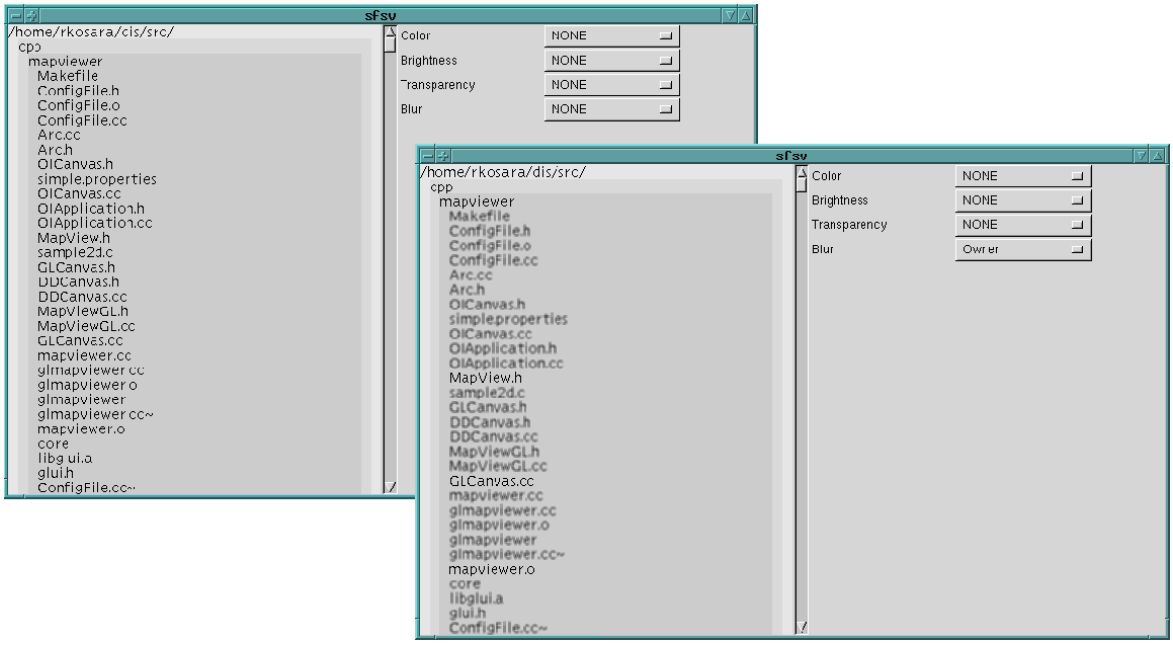


Figure 6.3: A file system viewer with all files in focus (top left) and one focusing on the files of one user (bottom right).

6.1.3 Interaction

In LesSDOF, the user cannot directly influence either the relevance or the blur function. When paging through a text, the “overlapping” lines are displayed using the minimum perceivable blur; when showing the results of a search, the irrelevant parts are displayed using the maximum acceptable blur.

6.2 sfsv: SDOF-Enhanced File System Viewer

File system viewers like the Windows explorer are among the most used applications on today’s personal computers. Some aspects of them are quite effective (like the tree view), while others are quite poor. One of the poorer aspects is the ability to quickly look for different information in a directory or directory structure without losing the context. Sorting the data according to a data dimension clearly is not a solution to this problem.

6.2.1 The Application

The *sfsv* application (SDOF-enhanced file system viewer) shows a directory structure in a slight variation of the well-known treeview (Figure 6.3). It is possible to do different queries on this data and show the results using different visual cues. One of these cues is blur. So if the user selects the file sizes to be shown blurred (Figure 6.3), the larger files are shown in focus, while the smaller ones are out of focus.

6.2.2 SDOF Aspects

Here, SDOF can be used both as an orthogonal cue and a reinforcement, depending on the user's needs. The combination of cues makes it possible to find files in their context, e.g., the ones that eat up all the hard disk space.

6.3 Sscatter: SDOF-Enhanced Scatterplot

Scatter plots are a very useful tool to get an overview over data and to test hypotheses. But scatter plots are only really useful for two data dimensions, others must be mapped to visual attributes of the displayed objects. A large number of easily distinguishable cues is therefore needed.

6.3.1 The Application

Sscatter can read data files in different formats whose structure (column delimiters, sizes, names, how many lines per data point, etc.) can be specified in a configuration file. It displays the data in a scatter plot, where the user can select which data dimensions are mapped to which visual features. When used on data of car models from 1993, one can see that more expensive cars have lower fuel efficiency, and that American and other cars are available over the whole price range (Figure 6.4 on the next page). It is also possible to find out that the availability of manual transmission is generally a feature of more expensive cars (Figure 6.5 on the following page).

6.3.2 SDOF Aspects

Because the user is free to choose data dimensions, a combination of binary (e.g., availability of manual transmission), discrete (e.g., number of cylinders) or continuous (e.g., price, engine size, etc.) relevance measure. What exactly is needed depends on what the user wants his or her new car to be or do.

6.4 SDOF-Enhanced AsbruView: sav

In a visualization of clinical therapy plans called AsbruView [26], there are several different knowledge roles. For each of these roles, information items can be defined together with a rather complex time annotation. These bits of information should be visible in a way that makes it possible to understand their relations with each other – both in time and with respect to criteria like which definitions use the same parameters, etc.

What makes comparisons especially difficult in this case is the fact that the plans themselves are structured (tree-like and possibly quite large), and the number of dimensions is high (there are five knowledge roles).

6.4.1 The Application

sav (SDOF-enhanced AsbruView) is a display of time annotation glyphs that are displayed in layers that are stacked in the direction orthogonal to the viewer. The user can select which layer he or she wants to see as the first (sharp) layer by selecting the "tab" of that layer. All

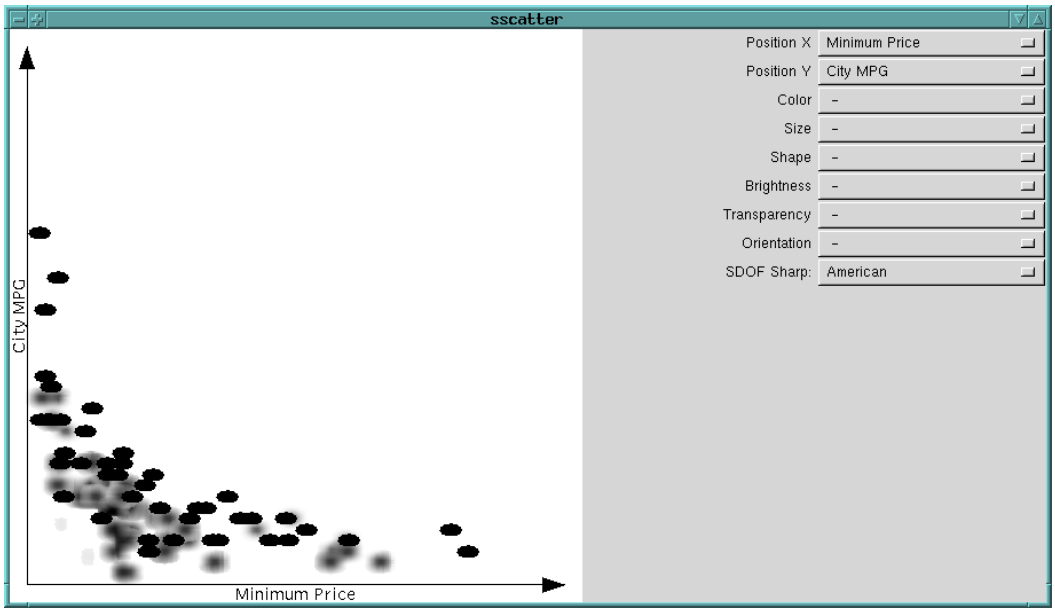


Figure 6.4: A scatterplot of car data showing that more expensive cars have a lower miles per gallon (MPG) number, and that American and other cars are available over the full price and MPG range.

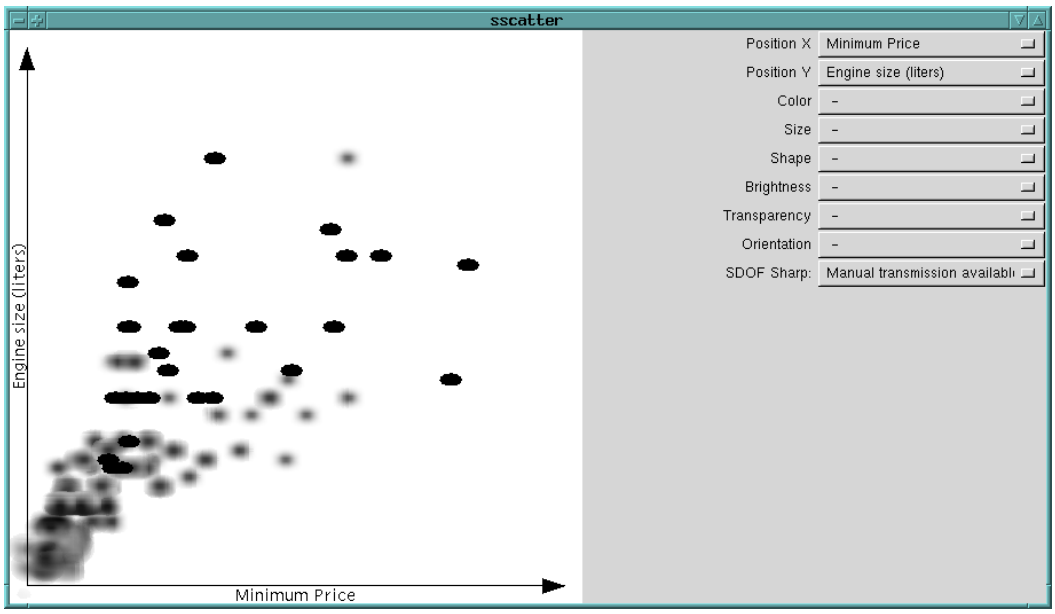


Figure 6.5: A scatterplot of car data showing that more expensive cars have larger engines, and that the availability of manual transmission is generally a feature of more expensive cars.

actions within a layer (folding of plans, scrolling) are, of course, synchronized between all the layers.

The user can also choose to select objects not based on the layer they are in, but by conditions such as the occurrence of a parameter or variable in expressions.

6.4.2 SDOF Aspects

This application demonstrates the use of layers in a more abstract view than the mapviewer (Section 6.6). It only uses a binary relevance function, but provides several different ways to apply them to layers and to objects within layers.

6.5 Chess Boards: sPGNViewer

Pointing out information in a display without hiding context and without changing that context too much can be quite challenging. At the same time, chess players, for example, "see" certain configurations and relations between chessmen. It would be interesting to mimic this, and thus be able to show the user information (in a tutoring system, for example).

6.5.1 The Application

sPGNViewer [27] reads descriptions of chess games in the Portable Game Notation (PGN), which is a simple text format using the algebraic notation used in chess. It is then possible to select the move after which the position of the chessmen on the board is displayed. Individual figures can be marked as relevant or irrelevant, and the program can show the user all chessmen that threaten or cover a certain figure, for example (Figure 6.6 on the next page).

The board can be displayed in 2D and 3D, and the 3D display can be turned and tilted.

6.5.2 Interaction

The user can select which move to show the board after, and also change between 2D and 3D display. It is also possible to select single fields on the board and change their relevance manually. Pointing to a figure makes it possible to select the relevance to be mapped to chessmen threatening or covering that figure.

6.5.3 SDOF Aspects

The most useful relevance function for this program is a binary one. But it is also possible to show those chessman that would be threatened if the figure of current interest was removed. In this program, the "auto focus" feature (Section 4.5.3) is also implemented.

6.6 sMapViewer: Layered Maps

When displaying a large number of information layers in a geographical visualization, the user has to decide whether to be distracted by too much information, or to have less – and possibly too little – information visible at the same time.

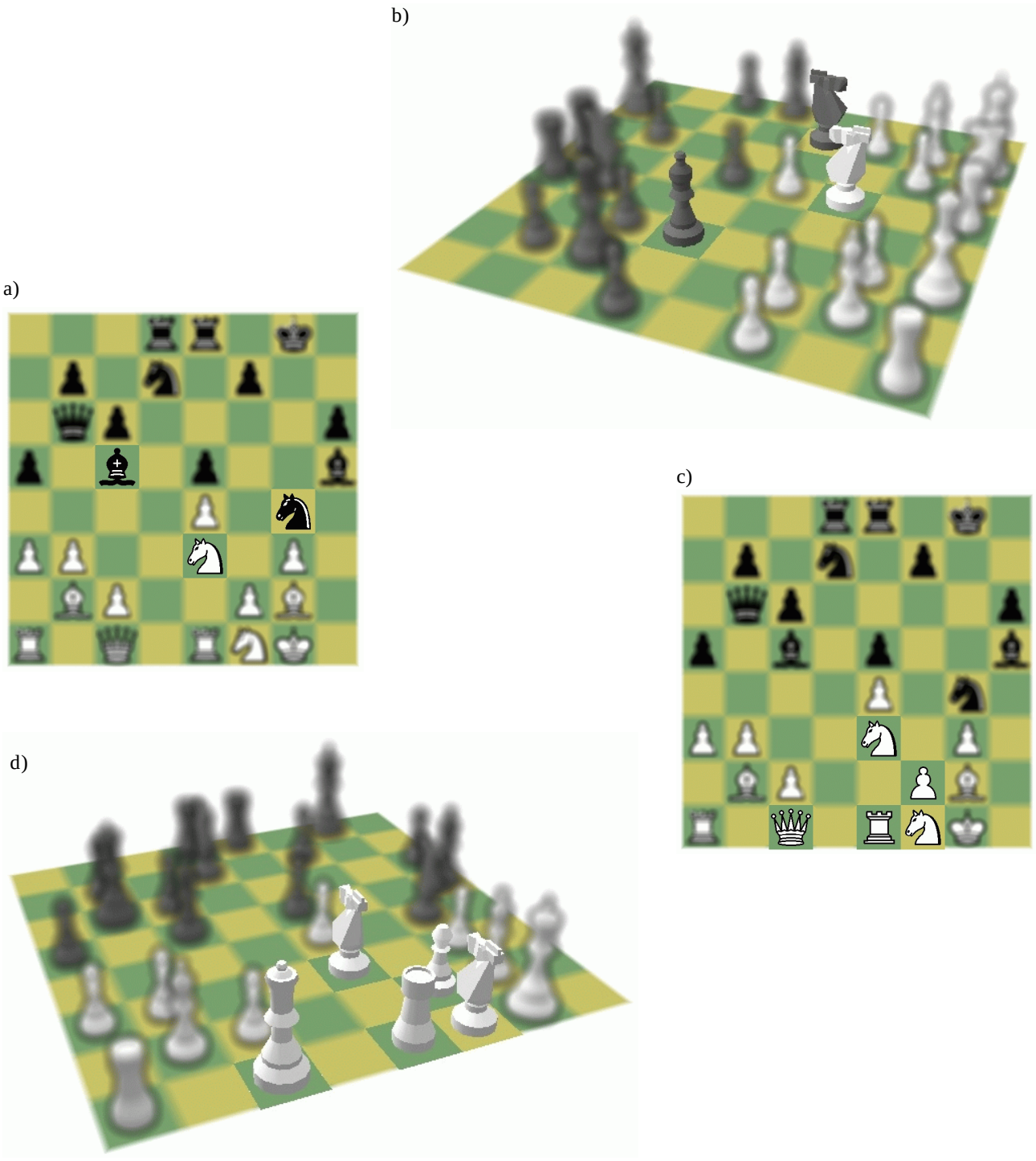


Figure 6.6: sPGNViewer: **a)** and **b)** showing the chessmen in focus that threaten the knight on e3; **c)** and **d)** focusing on the chessmen that cover the knight on e3.

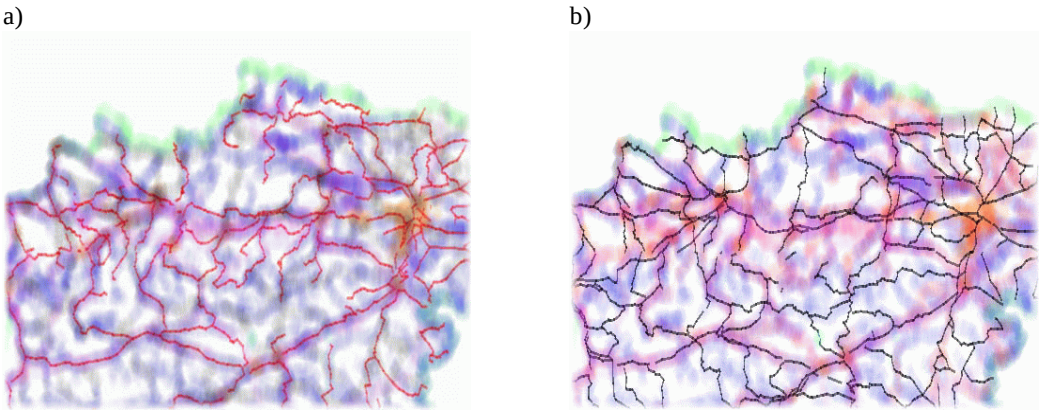


Figure 6.7: sMapViewer: The left part shows railroads in focus, while the right one focuses on rivers.

6.6.1 The Application

sMapViewer allows the user to select which layers to focus on by changing their blur level. This way, the user is provided with context information while not being distracted by it.

There are two version of this program that differ in their display style and the ways the user interacts with them. In the "regular" version (Figure 6.7), the map is read from a vector file and the relevance of each layer can be directly and independently controlled.

In the user study version (Figure 7.10 on page 67), there are three different display modes and also different interactions (which are described below). The display modes are *semi-transparent*, *SDOF*, and *opaque* mode, which are described in more detail in Section 7.3.10.

6.6.2 Interaction.

In the "regular" version, the user can directly change the relevance for each layer, and thus has complete control over every detail. It is also possible to zoom and pan the image. It is not possible, however, to change the order of the layers.

In the user study version, the user can only select a layer to be put "on top" of the stack of layers. In the opaque version, it is also possible to switch individual layers on and off.

6.6.3 SDOF Aspects

The regular version of this program contains the possibility of defining a continuous relevance function. The user study version uses a discrete function in SDOF mode.

Chapter 7

Evaluation

Visualization methods need evaluation. We therefore performed a user study to test some of the hypotheses we had about the nature of SDOF (such as its preattentiveness).

The study was prepared in cooperation with the Center for Usability Research and Engineering (CURE) and was performed in the CURE Usability Labs in August 2001. It was financed by the VRVis Research Center and the Institute of Software Technology (IFS), Vienna University of Technology.

This section describes the hypotheses we tested (Section 7.1), the test design (Section 7.3) and the results we got (Section 7.4). These results are discussed and conclusions are drawn in Section 7.5.

7.1 Hypotheses

The following hypotheses were tested in the study. Each of them is shortly presented and described here. The details of how they were tested are given in Sections 7.3.4 to 7.3.11.

Sharp objects can be detected and located preattentively among blurred ones. This hypothesis is the core of the whole study. It states that it is possible to detect the presence of a sharp object and to locate it (at least the quadrant it is in), even if the image is only shown for 200 ms. This is important in practice, if SDOF is to be used as a tool for guiding the viewer. This hypothesis was tested in block 1.

The percentage of sharp objects among blurred ones can be estimated preattentively. Being able to estimate the ratio of the number of objects with and without a certain feature is another preattentive task. The block that tests this hypothesis is block 2.

Sharp objects can be found faster than rotated ones. This is a comparison of search speed with orientation, another preattentive feature – tested in block 3.

The combination of blur with other features does not impact search time. Combinations of features generally make search tasks slower, but we wanted to know if this was also the case for SDOF. This hypothesis was also tested in block 3.

Blur is perceived in a way that makes it useful as a full visualization dimension. Can blur act as a full addition to color, etc? And if, how is it perceived (linear, logarithmic, etc.)? This was tested in block 4.

SDOF makes finding keywords easier. The efficiency of the LesSDOF application was tested with this hypothesis in block 5.

SDOF makes it possible to judge scatter-plots faster. Scatter-plots and similar applications are quite common applications, so we wanted to know the usefulness of SDOF in such a setting – tested in block 6.

SDOF makes reading layered maps faster. The layer metaphor is at the core of the SDOF idea, and its most natural application are maps. We therefore wanted to know if it can be understood and worked with effectively. This last hypothesis was tested in block 7.

7.2 Sample

To rule out large differences in perception between test participants, and to allow for a rather small sample size due to financial and time constraints, we selected a rather narrow group of participants who all fulfilled the following requirements:

- male
- aged 18–25
- very good vision (no contact lenses or glasses)
- students at university
- basic computer knowledge

The sample size was 16 individuals, which we recruited from different universities in Vienna. Each participant was paid a small amount of money for taking part.

7.3 Test Design

This section describes the hardware and software environment of the test, as well as the design of the individual blocks.

7.3.1 Hardware Setup

We used a Dual Celeron 433MHz PC with 128MB RAM and an nVidia Geforce2 GTS graphics card. The screen was a Philips 150B TFT LCD screen with a resolution of 1024x768 pixels which was able to display 16.8 million colors. The screen was run at 75Hz – this is important for the maximum error in the preattentiveness trials.

For the first two blocks of the test, it was necessary to display images for 200 ms. The screen refresh was synchronized with the vertical refresh. At 75 Hz, a screen refresh takes 13 ms, which is the maximum delay between drawing and displaying the image; and also between clearing the screen and the display of the empty screen. This introduces a maximum error of 6.5%, which is not significant for these trials.

7.3.2 Software

All software used in this trial was developed specifically for it. The programs were written in C++ using OpenGL and wxWindows, and ran on GNU/Linux 2.4.7 using XFree86 4.0.2.

Block Number	Description
Block 1	Preattentive Detection & Location
Block 2	Preattentive Count Estimation
Block 3	Disjunctive and Conjunctive Search
Block 4	Relation of Blur Levels
Block 5	Application: LesSDOF
Block 6	Application: sScatter
Block 7	Application: sMapViewer
Block Q	Qualitative Questions

Table 7.1: An overview of the structure of the user study.

For blocks 1 (target detection and location), 2 (count estimation), 3 (feature interplay), and 6 (Sscatter), a program was used during the test that only displayed images and an answer screen. This program pre-loaded the next image while waiting for the answer by the participant. This ensured predictable display times that were independent of image loading and decoding (the image was already in memory when it was needed).

The images for blocks 4 (relations), 5 (LesSDOF), and 7 (sMapViewer) were drawn directly during the trial.

All programs produced log files, which were simple tab-delimited ASCII text files. Each program first wrote a line containing the number of the participant, the block number and a complete time stamp to the file. After a blank line, a line with column headings was written. All lines after that contained the actual log information from the test. Each of these lines contained at least the number of the participant, a line number, the block number and a time offset in milliseconds from the start of logging.

7.3.3 Test Layout

The test consisted of eight blocks (seven quantitative blocks and one qualitative one), as described in table 7.1. The order of blocks and the order of images or questions inside of every block were changed for every participant to cancel out learning effects (except for the questions in block Q, which were always asked last).

In the first three blocks, we used images created by a program to test for preattentiveness. The objects shown in these images were put in an 8x8 grid, and were displaced randomly with an equal distribution of angles, and a Gaußian distribution of the distance to the center of each field. All images were 512x512 pixels in size. For the objects, we chose ellipses, because they have the advantage of keeping their shape even when blurred quite strongly (in contrast to rectangles, which become ellipses ...), and they can be rotated which makes comparison between color, blur and orientation in the later blocks easier.

The images were manually checked for overlapping ellipses and ellipses too close to the borders between quadrants.

7.3.4 Block 1: Preattentive Detection and Location

The first block was designed to test the first hypothesis (see Section 7.1): Sharp objects can be detected and located preattentively among blurred ones.

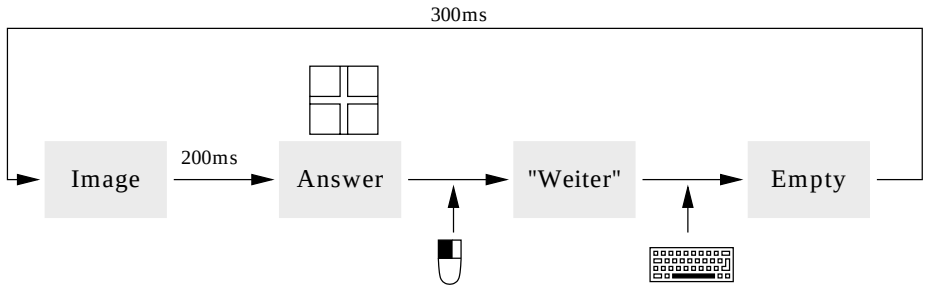


Figure 7.1: Structure of Block 1

For this purpose, we showed the participants images similar to Figure 7.2 on the following page for 200 ms. All objects in the images of this block were black and had their principal axis parallel to the x axis (i.e., horizontal). Half of the images in this block contained exactly one sharp target, the other half did not. There were 3, 32, or 63 distractors, i.e., blurred ellipses (64 in the case where no target was present vs. 63 plus one target). Images were created with each combination of three different blur levels (blur diameters of 7, 11 and 15 pixels). For each combination of parameters, 30 images were created, resulting in 1260 images (30 images, 2 target cases (present or not), 3 numbers of distractors (3, 32, 63), 7 selections of 3 blur levels).

For every participant, five images were chosen per parameter combination (resulting in 210 images per participant), and put into a random sequence.

Before the start of the main part of the test, participants were shown a sample image from the data set and then had 10 trial images. During the test, the participants were first shown the test image for 200 ms, and then an answer screen which consisted of frames for all four quadrants that they could click as well as two buttons, one for “no target” and for “target not locatable”. The buttons as well as the quadrant frames reacted by being drawn with a lighter background when the mouse was moved over them.

After the participant had clicked on one of the quadrants or one button for his answer, an empty screen with the word “Weiter” (“next”) was shown. To get to the next image, the participants then had to press the space bar. This was done to give the subjects control over when the next image would appear, and also to enable them to take short breaks when they would get tired. After the key-press, a blank screen was shown for 300 ms, and then the next image appeared (Figure 7.1).

After every 30 images, the participant was shown a screen telling him to take a short break. But because participants were able to take breaks at any time anyway, these were seldomly taken.

7.3.5 Block 2: Preattentive Count Estimation

Another aspect of preattentive processing is the ability to estimate the ratio of target objects in an image, described in the second hypothesis: The percentage of sharp objects among blurred ones can be estimated preattentively.

To test this, we showed the participants images in which there was an object in every position of the grid, with different numbers of targets and different blur levels. In each image, there were between 5% and 95% targets (in 10% steps, which translated to 3, 9, 16,

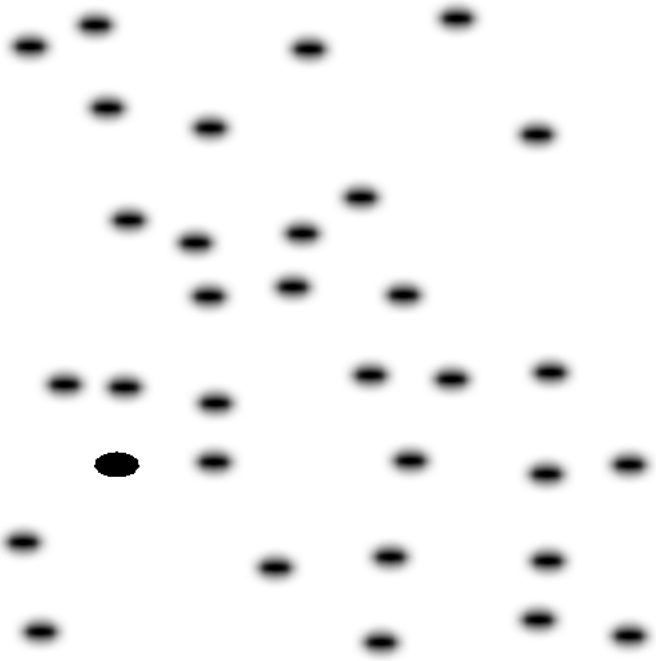


Figure 7.2: Example image for block 1 (32 distractors, 1 target present)

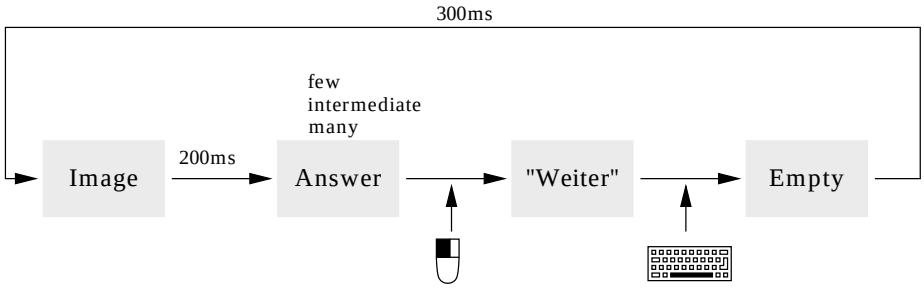


Figure 7.3: Structure of Block 2

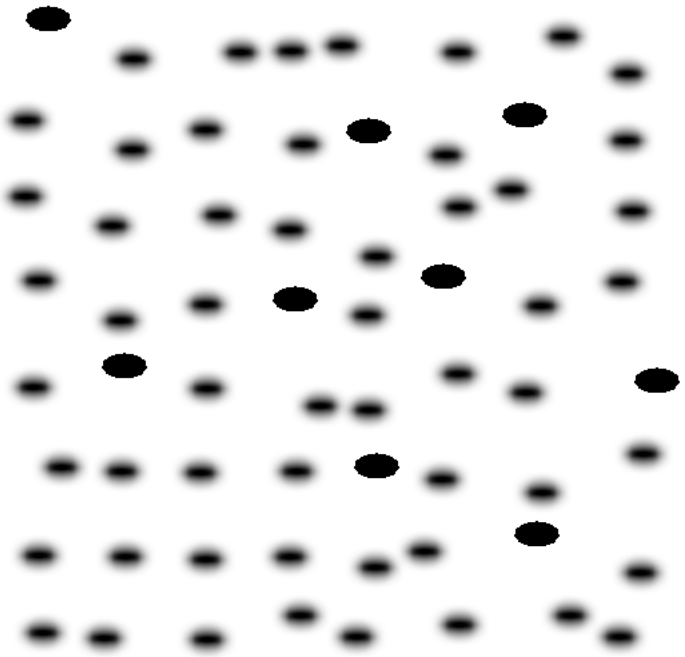


Figure 7.4: Example image from block 2 (count estimation) with nine targets and 55 distractors.

22, 28, 35, 41, 48, 54 targets), and all distractors were of the same blur level per image. Five images were created for every combination of parameters, and for 3, 9, and 54 targets, also an "ungrouped" version was created (see below). This resulted in 180 images, five images were selected from every parameter combination and put into a random order for every participant.

The sequence of screens for this block (Figure 7.3 on page 59) is similar to block 1. But instead of click-able quadrants, the participants were given three buttons to choose whether there were "few", "intermediate", or "many" targets (where "few" were from 1 to 19 objects, "intermediate" from 20 to 45, and "many" from 45 to 64 objects – these numbers did not coincide with the numbers in images so that every image was clearly in one class (and not on the border between two)).

We also wanted to differentiate between images where objects formed groups and images with ungrouped objects, but it turned out to be impossible to have completely ungrouped objects in an image where more than a quarter of the objects are either targets or distractors (we wanted to at least have ungrouped distractors for images where "ungrouping" targets was not possible. But even this was not possible).

7.3.6 Block 3: Interplay

In the third block, we wanted to test how well the combination of blur and color as well as blur and orientation compared to each of the cues on its own, and how problematic ignoring each of them would be.

This block consisted of two parts: one for *disjunctive search*, and one for *conjunctive search*.

Disjunctive search means looking for an object with one specific feature, ignoring the others, even though they are different in the targets and the distractors. For example to find the sharp object independently of its color (which could be black or red), with blurred distractors 50% of which were black, and 50% red. There was also a control test for the same task with all distractors black and a black target, as well as one with all distractors red and a red target.

The second task was to look for the red object, independently of its blur level, among black distractors that could be blurred and sharp (again 50% blurred, 50% sharp). Again, there was a control test with all objects sharp, and one with all objects blurred.

For disjunctive search, 540 images were created (30 for each combination of parameters), and 90 were selected for every participant (5 from every parameter combination).

Conjunctive search means looking for a combination of features to find the target. Table 7.2 on the next page gives the combinations of features to look for, and which distractors were present in the images (Figure 7.5 on the following page gives an example).

The screen sequence for this block is similar to the previous two (Figure 7.6 on the next page). But instead of a fixed time that the images were displayed, the participants could look at them as long as they needed to find the answer to the question – they were of course told to answer as quickly as possible.

7.3.7 Block 4: Relations of Blur Levels

The goal of this block was to find out how blur levels are perceived and if blur could be used as a separate, fully-fletched visualization dimension.

Sub-part	Target	Distractors
a	black and sharp	red sharp, red blurred, black blurred
b	red and sharp	black sharp, red blurred, black blurred
c	black and rotated	red rotated, red horizontal, black horizontal
d	black and horizontal	red rotated, red horizontal, black rotated
e	horizontal and sharp	rotated sharp, rotated blurred, horizontal blurred
f	rotated and sharp	horizontal sharp, rotated blurred, horizontal blurred

Table 7.2: The tasks and distractors for conjunctive search in block 3.

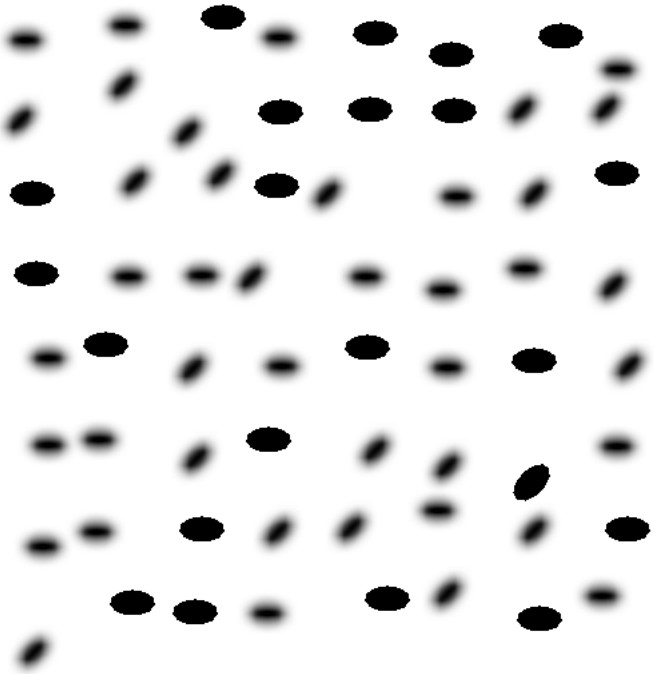


Figure 7.5: Sample image from block 3: Find the rotated sharp object.

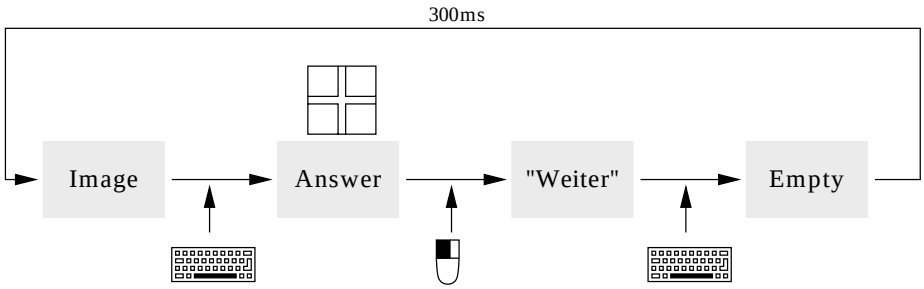


Figure 7.6: Structure of Block 3

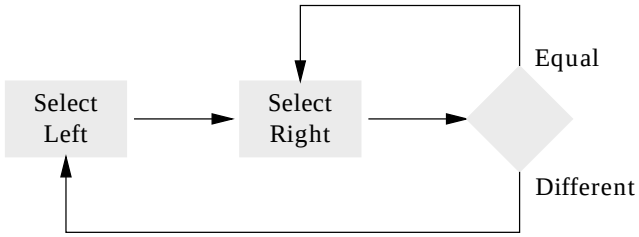


Figure 7.7: Block 4 (relations of blur levels)

This block consisted of five parts. The task in parts a to c was to judge the difference in blur between two objects. For this purpose, two ellipses were shown close to each other. The participant would compare the blur levels and then click on one of the buttons “equal” or “different”. Depending on the part, the objects were then changed (and also sometimes left the same in the “equal” case to make errors from expectations less likely): in part a, the right blur level was increased if the user clicked “equal”, and a new starting point (equal for both objects) was selected in the case of “unequal”. In part b, the blur level of the right object was decreased in the case of “unequal”, and the objects started at a high blur level. In part c, a random level and a random direction was chosen, and the user had the choice between “left object sharper”, “equal”, “right object sharper”. Depending on his answer, the object was changed (similar to parts a and b, depending on the direction).

Part d was designed to find the absolute threshold for blur perception. For this block, only one object was shown, first starting from with a sharp one, and then starting with a clearly blurred one (both cycles were repeated three times). The user had the choice of two buttons, “sharp” and “blurred”. When starting with the sharp object, the blur level was increased (or left equal) when the answer “sharp” was given, until the participant answered “blurred”.

In parts a to d, an empty screen was shown for 300 ms before the next image was displayed, to make changes less obvious to the user, and to make the exercise less tiring when no change happened for several screens (so the user would not believe the mouse click to have had no effect).

In part e, two objects were shown to the participants that were blurred at different levels. The participant had to give a numerical estimation of the blur factor of these two objects. These numbers were given orally, and noted by the test supervisor.

7.3.8 Block 5: LesSDOF

LesSDOF (Section 6.1) is an application that can show texts and search for keywords in them.

There are three modes in LesSDOF for this study: a) only the keyword is displayed with inverted colors (e.g., white text on black background); b) keyword is inverted, primary context (sentence) is displayed sharply, everything else (second-level context) is blurred (Figure 6.2 on page 48); c) keyword is inverted, context is drawn on a gray background, the rest of the page is displayed sharply (Figure 7.8 on the next page).

It was possible to step from one hit to the next by pressing the arrow down key, or to go back to the previous hit by pressing the arrow up key. At the beginning and end of the text, the navigation would simply wrap around, and this would be visible because of the position

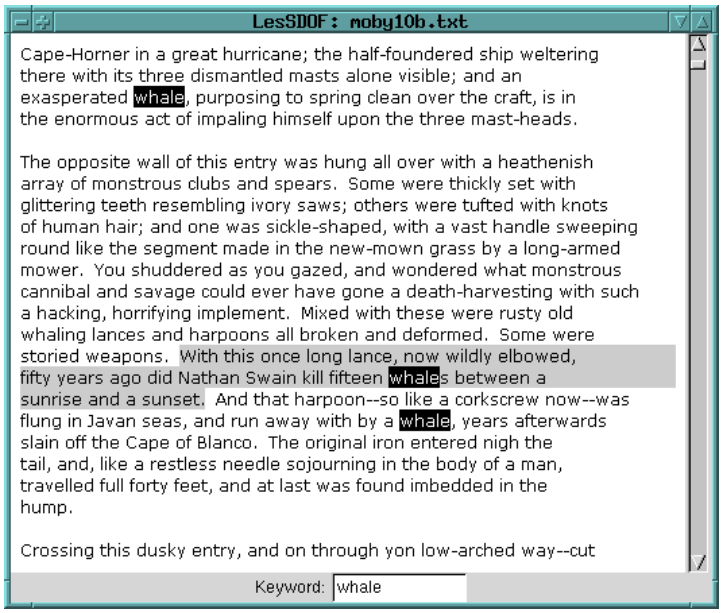


Figure 7.8: LesSDOF in *grey* mode.

of the scroll bar.

The participants got twelve different texts with different tasks (specific to each text) that they should solve by entering a keyword (which was predefined as part of the task) and then going from hit to hit, trying to find the answer in the context of the keyword. As soon as they knew the answer, they would press the space bar, which would cause the search time to get logged. The answer was then given orally.

7.3.9 Block 6: Sscatter

The subjects were shown images from the Sscatter application. They were asked to point to certain objects which were defined by the values they stood for. The data set contained census data and election results from the 50 states of the USA from the 2000 presidential elections. In all images, the color encoding showed the average income in that state (blue for high income, green for low), and the election results were encoded to show states with a majority for George W. Bush in focus or rotated, and the others blurred or not rotated (Figure 7.9 on the following page).

The tasks for each image were given to the participants orally and in written form. In the first part, they consisted of clicking on object fulfilling two or three criteria (position along one of the axes, sharpness / orientation, and, in some images, also color). The participants had to click on the object they thought was the correct one.

In the second part, participants had to click on the center of gravity of a set of objects – defined by color, blur, or orientation.

In both parts, the location of the correct object or point (center of gravity), the location of the mouse click, and its euclidian distance from the correct point were logged.

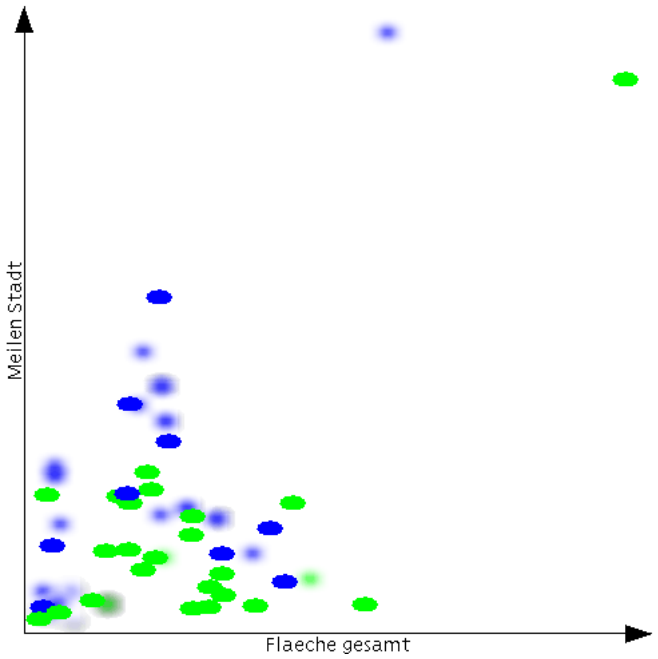


Figure 7.9: An example image from block 6. The image contains census data and election results from the 2000 presidential elections in the USA. The axes encode total area (x axis) and miles city (y axis), blue objects stand for richer states, and sharp objects represent states with a majority for Bush.

7.3.10 Block 7: sMapViewer

Similar to block 5, there were three different modes in sMapViewer for this study: *semi-transparent*, *SDOF*, and *opaque* mode. The map was a hand-drawn imaginary map with nine layers of information (rivers, railroads, highways, cities, raw materials, vantage points, low real estate costs, nature reserves, and industrial areas).

In *semi-transparent* mode (Figure 7.10a), all layers were drawn semi-transparently (with 50% transparency). Their depth order could be changed by clicking on the name of one layer, which was then brought to the top.

In *SDOF* mode (Figure 7.10b), the top-most layer was displayed sharp, while all others were more and more blurred (with exponentially increasing blur). Similar to semi-transparent mode, the user could click on one layer's name to bring it to the top.

In *opaque* mode (Figure 7.10c), the layers were drawn on top of each other without transparency. The user could not only change their depth order, but also hide layers completely by clicking a little check-box next to their names. When a layer was brought to the top of the stack, it was also automatically made visible.

The program logged the mode and the order of layers (and which layers were visible) after every interaction.

7.3.11 Block Q: Qualitative Questions

This block was always the last one, and consisted of oral questions to the participant. The following questions were asked (English translations given here):

- What is your overall impression of the use of sharpness / blur in user interfaces?
- Have you noticed anything special about your visual perception during or after the test?
- Which pros and cons do you see for the use of blur in applications?
- Are there any applications which this way of visualizing information seems especially suited for?
- Are there any other remarks you would like to make?

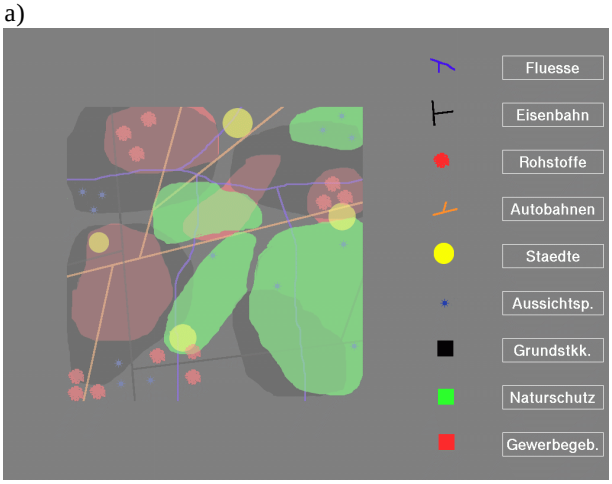
In addition to these questions, the participants were also asked several times during the test (mostly between blocks) if they had any remarks or if they noticed anything special.

7.4 Results

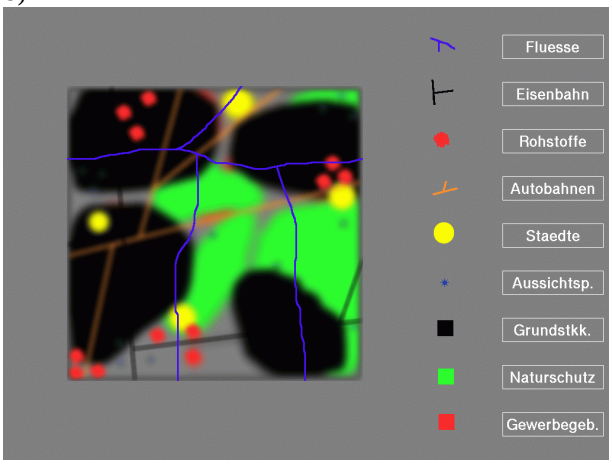
In this section the results are presented for every block, and then discussed in Section 7.5 on page 77.

Due to some technical problems and one participant misunderstanding a task, we were not able to use data from all 16 participants. We had the data of the full set of 16 participants for blocks 1, 3, 4, and 5. For block 2, we only had data from 14 subjects; for block 6, we had data from 12 subjects; and for block 7, there were 13 usable data sets.

a)



b)



c)

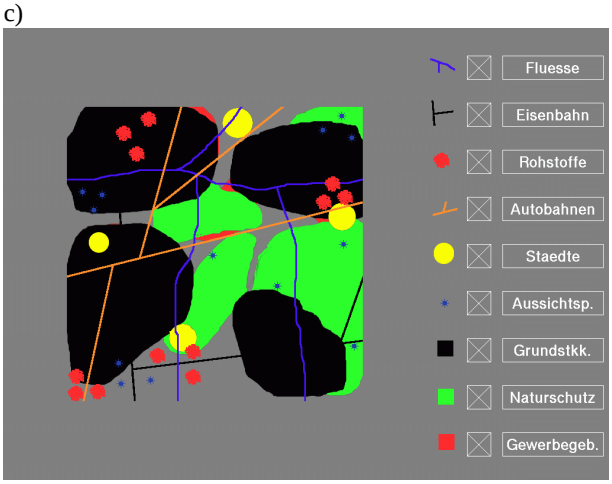
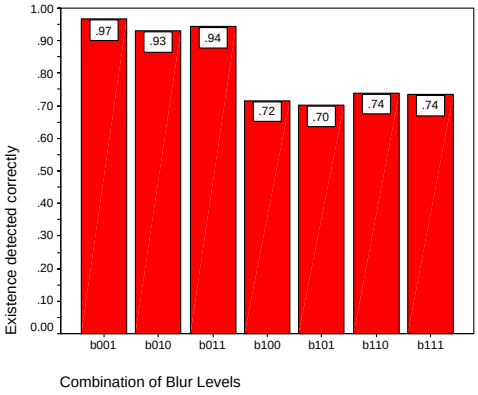
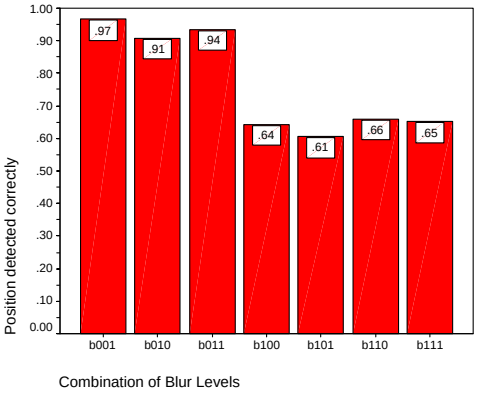


Figure 7.10: The MapViewer used in the user study. a) *semi-transparent* mode, b) *blur* mode, c) *opaque* mode.

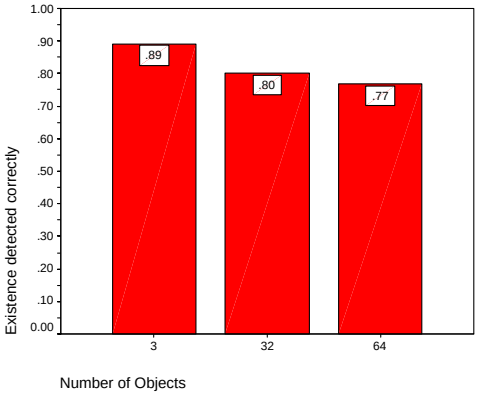
a) Target detection by distractor blur levels



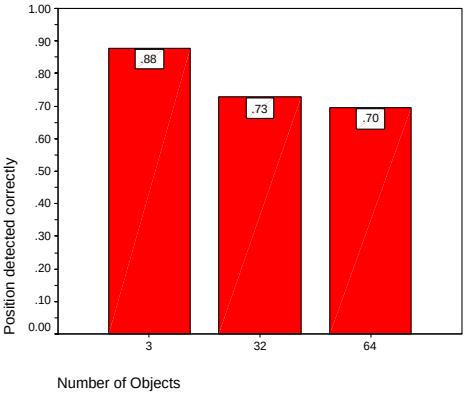
b) Target location by distractor blur levels



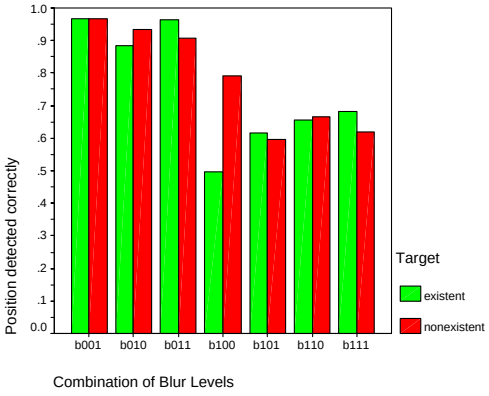
c) Target detection by number of objects



d) Target location by number of objects



e) Target location by blur and existence



f) Target location by blur and number

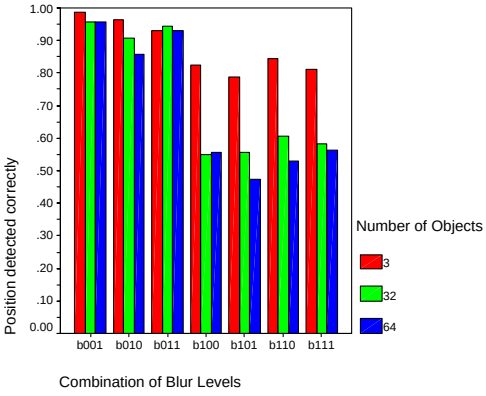


Figure 7.11: An overview of the results of block 1 (target detection and location). The percentage of correct answers is given for different combinations of parameters. Encoding of distractor blur levels: After the b, three digits represent the three blur levels. If the digit is 1, that blur level was present; if it is 0, it was not.

7.4.1 Block 1: Preattentive Detection and Location

The subjects were able to detect targets and also to point to the correct quadrant with a very high probability. Figure 7.11a shows these probabilities in dependence of the blur levels used in the images. The names of the bars are encoded as follows: the three digits after the b stand for the three blur levels, the first one for the smallest blur, the second one for the intermediate blur level, and the third one for the strongest blur used. If the number is 1, objects with the corresponding blur level were present in the image, if it is 0, that blur level was not used.

It is quite obvious that the smallest blur level strongly influences the accuracy for target detection and location – the difference between the first three and the remaining four cases in Figures 7.11a and b is significant.

There is a slight difference in being able to detect the sharp object and being able to locate it. The reason for this could be that objects near the quadrant border were detected but then associated with the wrong quadrant. Another explanation is that for those images where the smallest blur level was present, cases where blurred objects were mistaken for sharp ones would add to the accuracy in cases where the target was present, but not less in cases with no target.

There is also a significant difference between the accuracy in the cases with three objects and those with 32 or 64 (Figure 7.11c and d). But the values are still very good even for many objects (77% for detection with 63 distractors, 70% for location).

There are no significant differences in accuracy when comparing images where a target was present with images where there was no target (Figure 7.11e) – except in the b100 case, where the accuracy was much lower when a target was present. This is most likely due to the fact that participants were not able to distinguish between the (slightly) blurred distractors and the target.

If one compares the accuracy depending on the number of objects *and* the blur levels (Figure 7.11f), an interesting difference between the images with the first blur level present and those without can be seen: For a small number of objects (3 in our case), the accuracy does not diminish as fast as with more objects. But for images with many objects, the smallest blur level makes them useless (accuracies close to 50%, which is not better than chance).

7.4.2 Block 2: Preattentive Count Estimation

When comparing the accuracy for images by number of targets (Figure 7.12a), the accuracy is higher for numbers farther away from category borders. This was to be expected, since participants make more mistakes near these borders. To substantiate this claim, a chi-square test was performed to compare the answers to random ones. For all three blur levels, a significant difference from chance was found (b001: $\chi^2 = 782$, b010: $\chi^2 = 706$, b100: $\chi^2 = 439$, all for $p < 0.001$). With absolute numbers of around 95% for the peaks in each category, the result is very good.

Looking at the blur level (Figure 7.12b), there is a significant difference between the accuracy when using the smallest blur level and the other two levels. The reason for this is most likely that objects with a very low blur are perceived as sharp, and therefore the estimate is too high. This was to be expected, given the results of the first block. But even the lowest accuracy of 74% is quite good, the other values are close to 90%, which is very good.

The above results can be analyzed further by splitting them up in two dimensions: By category and by blur level (Figure 7.12c). Higher blur levels show significantly better accu-

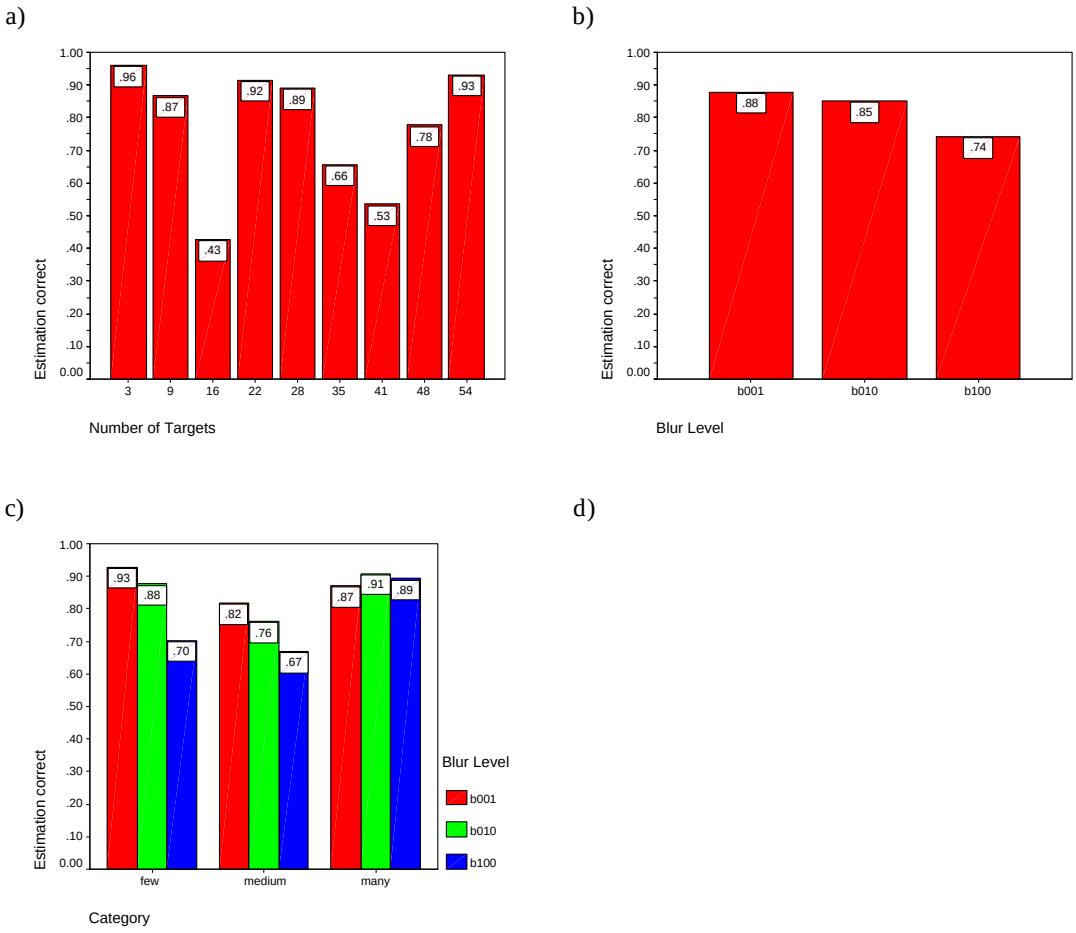


Figure 7.12: An overview of the results of block 2 (count estimation). **a)** average ratio of correct answers for count estimation by number of targets actually present in the image; **b)** average ratio of correct answers for count estimation by distractor blur levels; and **c)** correct answers for target detection by number of objects and distractor blur levels.

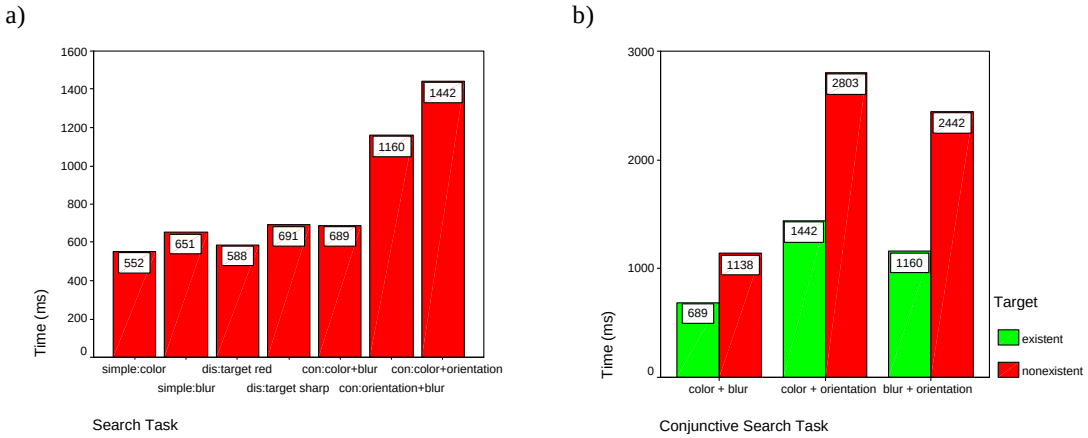


Figure 7.13: An overview of the results of block 3 (interplay). **a)** time needed for search by search task ("simple": only look for one feature, with no other feature present; "dis": disjunctive search for one feature with two distractor features present; "con": conjunctive search for combination of features; **b)** search times for conjunctive search by search task and existence of target.

racies, and the fact that the lowest blur level has its highest accuracy with the most objects (other than the other two, which have high accuracies at both "ends") substantiates the claim made above that objects with a small blur level are perceived as sharp.

7.4.3 Block 3: Interplay

This block may be the most interesting one in this study. It compares blur to color (Figure 7.13a), one of the most widely used and strong visual cues. And it shows that there is no significant difference between a simple search for color or for sharp objects! The conjunctive searches (for color and blur, color and orientation, as well as blur and orientation) also differ significantly from each other. But the conjunctive search for color and blur does not significantly differ from the simple searches for color or sharpness. This is surprising, because the combination of features usually means longer search times [59].

The differences between color and blur are not significant, which is also a very interesting result.

As Figure 7.13b shows, the search times were longer when no target was present. This is not surprising, because this trend can be found in most other tasks and is also supported by theory [13]. It simply takes the subjects longer to make sure there is no target and that they haven't missed it.

The error rate was very small in this block: for the combination of color and blur, there was only one error; for blur and orientation, there were five errors; and for color and orientation, there were four errors – these are the number over all images shown to participants in this block, which were 90 per participant, and 1440 total (this means an error rate of less than 0.7%). The difference between the three tasks are not significant, and this low rate was to be expected since subjects had as much time as they needed for every image. But it shows that they took the study seriously, and really tried to answer correctly – and that they did

Blur Level	Value
1	1
5	1.49
9	2.19
11	2.6
13	3.52

Table 7.3: Comparison of real and perceived blur values.

not sacrifice accuracy for speed.

7.4.4 Block 4: Relations and Blur Levels

When looking at the simplest task of this block, judging whether the blur levels of two objects that are side by side are identical, subjects more often mistook different blur levels for the same one (Figure 7.14a). When splitting this result up by the blur level of the left object (which was never very different from the right one, see below), it becomes clear that people are not able to identify blur levels as equal for larger values (Figure 7.14b) – there is a rather clear downward trend in the (green) "yes" answers for the identical cases. This is much less pronounced than for different blur levels, which are quite well discriminable.

As Figure 7.14c shows, there is virtually no difference between the different modes that were used to measure the relative threshold. All three modes yielded the same results for both cases.

The results for the distance needed to discriminate between blur levels fit quite well with the image painted with the above results (Figure 7.14d). There is no clear trend in the needed distance, the values seem to be quite random – they are all between 1 and 1.8, though, and their average is quite small (about 1.3).

The absolute thresholds needed to discriminate between sharp and blurred (and not different levels of blur) are more precisely measurable (Figure 7.14e). Participants perceived an object to be sharp when its blur diameter was around 3.27 when starting with a clearly blurred object and decreasing its blur level; and started seeing objects as blurred at an average b of 1.46 when starting with the sharp object. The last part of this block was to find out the ratio between real and perceived blur. Table 7.3 and Figure 7.14f show quite clearly, that the perception of blur is very weak, and is quite different between subjects. The participants only gave a perceived ratio of 3.52 (on average) for a real ratio of 1:13, which is practically useless for a continuous blur.

7.4.5 Block 5: LesSDOF

The results of LesSDOF and the other application blocks were quite disappointing – not because they showed that SDOF was a bad method, but because they were not very clear. Even though most of these results are not significant, they shall still be presented here.

While the search times were shorter for color and SDOF highlighting (Figure 7.15a), they were not significantly better than without any context information. Participants made hardly any mistakes (most of which could be traced to misunderstanding the question), so there was also no difference in the accuracy of the answers.

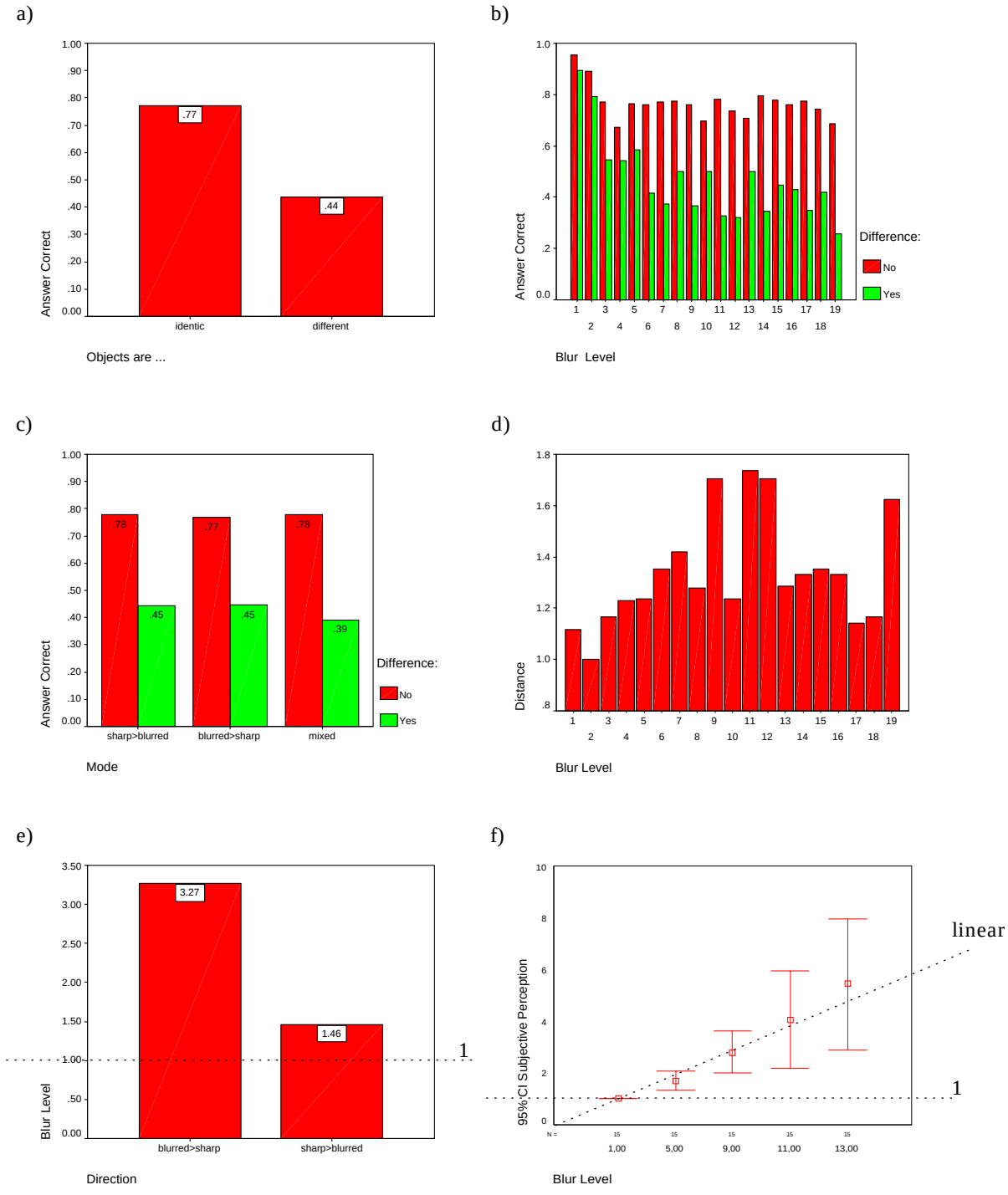


Figure 7.14: An overview of the results of block 4 (Relations and Blur Levels). **a)** Correct answers for identical and different objects; **b)** Correct answers for identical ("no") and different ("yes") objects, by blur level used for the less blurred object; **c)** Correct answers depending on direction of blur; **d)** Distance needed to detect difference, by blur level of less blurred object; **e)** Absolute blur level needed to detect sharp or blurred object, when only one object was present; **f)** Numerical answer to perception of absolute blur value, by displayed blur value.

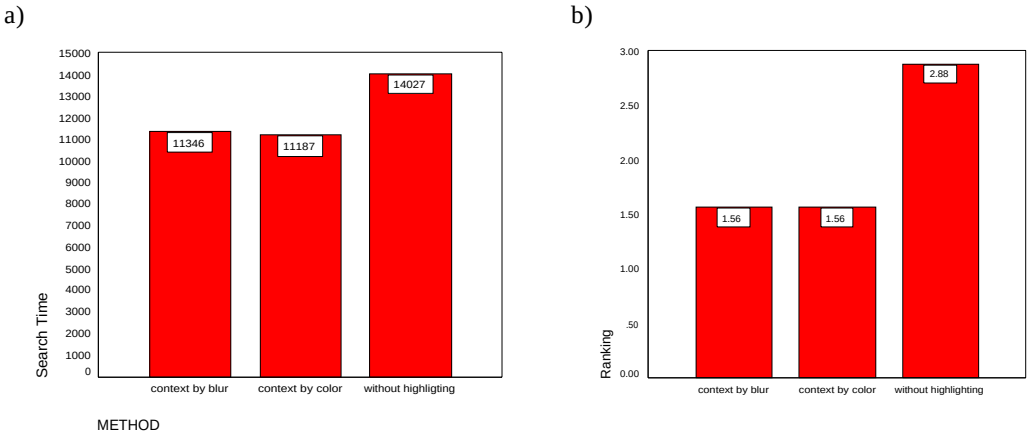


Figure 7.15: Results of block 5 (LesSDOF). a) shows the mean search times for the different methods; b) shows the ranking of the three methods by the users (lower values mean more useful method).

When ranking the three methods, however, blur and color were equal (at 1.56, Figure 7.15b), while no context was ranked significantly worse ($\chi^2 = 18.3$, $\mathrm{df} = 2$, $p < 0.001$).

7.4.6 Block 6: Sscatter

Unfortunately, there were some errors in the software that made the results of this block almost unusable. In the images, the encoding for orientation was wrong, and so the tasks (finding the sharp / rotated target with a certain set of features) were very different in their difficulty.

With all due caution, the results are shown in Figure 7.16a. They show no significant difference (and, in fact, hardly any difference at all) between blur and orientation.

For the second part – finding the center of gravity – these problems were not quite as severe, and therefore an analysis was possible. It showed that when using blur, the participants' guess for the center of gravity was a significantly smaller distance away from the correct answer than with color ($F(1, 10) = 50.78$, $p < 0.001$; Figure 7.16b). There was no significant influence of time to answer on the correctness of the result.

7.4.7 Block 7: sMapViewer

The last quantitative block yielded even fewer interesting results. The response times (Figure 7.17a) were not significantly different. In *semi-transparent* mode, there were fewer clicks, because participants could see all the layers at once. This is also reflected in the number of interactions (Figure 7.17b), and there is a significant correlation between the number of clicks and the response time ($p < 0.001$). This means that participants mostly clicked on layers, and did not spend different amounts of time thinking about the answer between the different methods.

The lower number of interactions in *semi-transparent* mode lead to some forejudgments (Figure 7.17c). This is not a statistically significant finding, however.

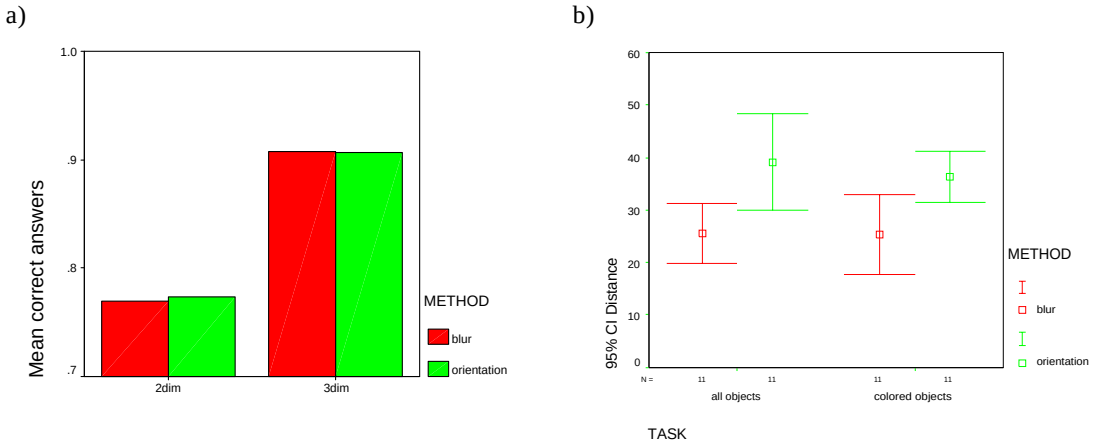


Figure 7.16: Results of block 6 (Sscatter). a) shows the mean rate of correct answers for blur and orientation encoding for two and three relevant criteria; b) shows the distance from the real center of gravity dependent on the method (blur or orientation) and on whether all objects or only those of one color were considered.

Blur was judged better than the other two modes by participants (Figure 7.17d), but this also is not significant.

7.4.8 Block Q: Qualitative Questions

Most participants liked the idea of blur to stress certain parts of the display, but only as an additional tool that they could deactivate.

The difference between context display using blur or color in block 5 was considered a good idea, but some participants said they preferred the use of color because the relevant context can also lie outside of the immediate context sentence.

The high marks for SDOF in block 6 (LesSDOF) were justified by the fact that ellipses with different orientation were hard to see when partly obscured.

Participants generally found the opaque mode of block 7 (sMapViewer) the most useful. Some said they would have wanted a combination of that mode with blur.

The opacity of layers in both opaque and SDOF mode was criticized by several of participants, because it required a lot of interaction to get the right layers into the right order. One person also said he would have wanted a “back” button to undo actions.

As for additional applications, participants suggested design tools, where less important parts could be “hidden”. Another idea was the use in computer games. A route planner was also given as an example of application, where the proposed route would be displayed sharply, while the surroundings were blurred.

Several participants said that they had seen a strong difference in size between blurred and sharp objects. Some participants reported seeing an after-image of the sharp objects, which made answering the question easier. Participants also mentioned a white border around sharp objects.

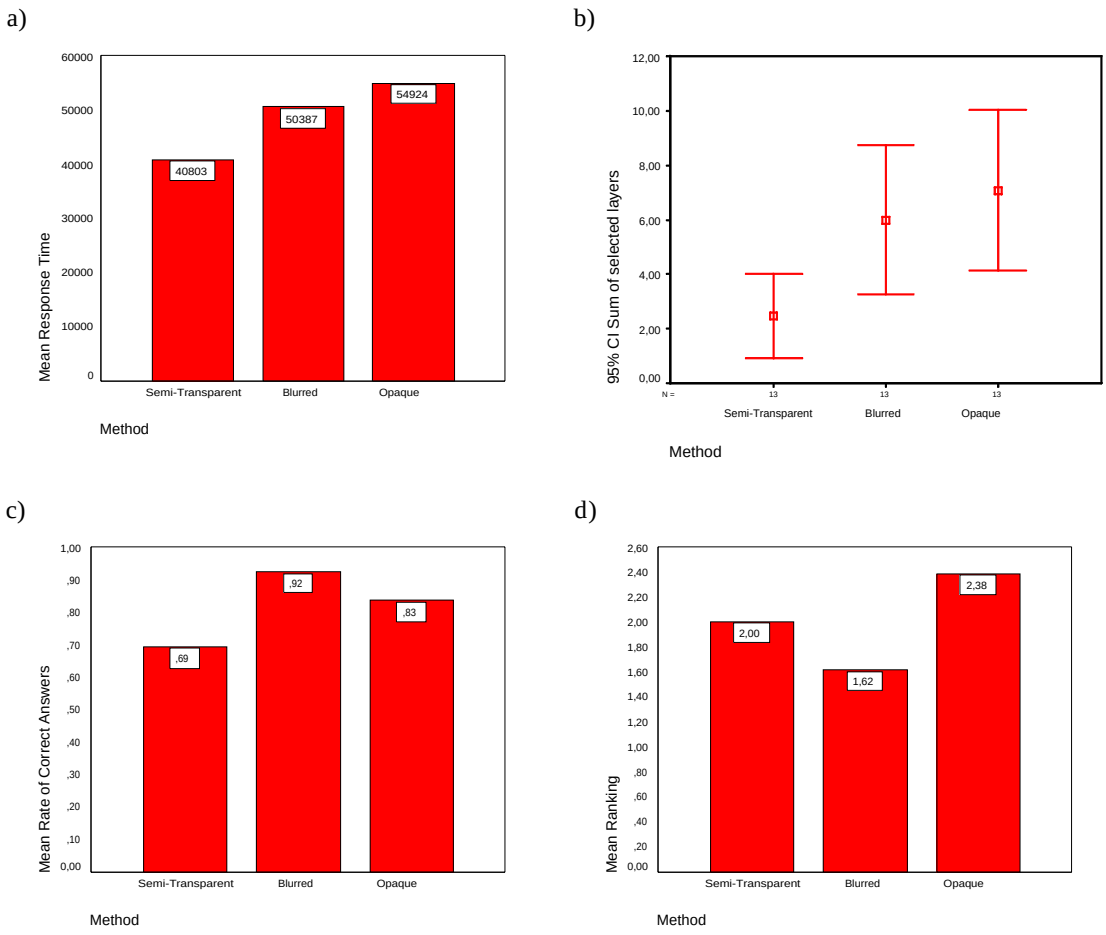


Figure 7.17: Results of block 7 (sMapViewer). a) shows the mean time from the start of the task to giving the answer; b) shows the number of selected layers before giving an answer; c) gives the accuracy for each method; d) shows the ranking of the methods by users (smaller numbers mean higher ranking).

7.5 Discussion and Conclusions

Blocks 1 and 2 clearly show what we were expecting: sharpness vs. blur is a preattentive feature. We did not compare blur to other features in these two blocks, and it is hard to compare this result to other studies which were done with different images, objects, numbers of targets / distractors, sample, etc. But the accuracy is very high (at least for larger blur levels), which gives us confidence in the method.

The results of block 3 are also very positive. The fact that SDOF was not significantly slower than color was a surprisingly good result, and shows the great strength of this method. It is astonishing that such a strong method has only been developed and thoroughly tested at this point in time – after information visualization and certainly perceptive psychology have existed for a long time.

Participants found block 4 particularly straining and annoying, because they had to concentrate on blurred objects and judge them, which is very unnatural. This is, of course, quite contrary to the way SDOF and blur would be used in practice, so this bad judgment is not a real problem for SDOF. But it shows that SDOF must be used with care, and it must be possible to turn it off quickly when the user wants to see context objects clearly.

Participants did not like the idea of a blurred text display so that they could not read it anymore. This means that in an application, text should never be blurred to such an extent as to be unreadable.

The white border around sharp objects is easy to explain [13]: The human eye is very sensitive to strong contrasts, and so saw the black-on-white contrast even stronger than it was.

The results from block 4 clearly show that SDOF cannot be used as a separate and fully-fledged visualization dimension. This is due to the fact that discrimination between blur levels is very poor, and participants were not able to judge the relations between blur levels in any useful way – they hardly saw any difference at all. This does not mean, however, that not a small number (three or four) blur levels could be distinguished in an application. This is subject to more research, though.

The application test blocks yielded very poor results, which is not only due to the programming errors. There were also design errors, especially in block 7 (sMapViewer). It appears much easier to design a “pure” psychological test like the first few blocks than a real application test. The number of clicks was a particularly bad choice for the performance measure in block 7.

Just to be sure, a further preattentiveness-test should be conducted that displays white noise after the image. This rules out the possibility of real after-images, and thus makes the test result stronger.

Further tests would also be needed to check how other populations react to the same stimuli.

Chapter 8

Summary and Future Plans

Focus+Context is an important area in information visualization and also related fields. A number of methods exist that provide more space on the screen or more data dimensions. But there are many applications where it is useful to point out information and to do that in the context of the other data so that the user knows what to make of that data.

These methods can be put into three categories, *distortion-oriented* or *spatial* methods, *dimensional* ones, and *cue* methods. Spatial methods provide more space on the screen by assigning larger portions to more important objects, and less to the rest. Thus, relevant objects appear larger, and their features can be read more clearly.

Dimensional methods display different data dimensions within a focus area ("lens"). This makes it possible to display less or different information for the remaining objects, and thus not clutter the display too much.

Both these methods are *user-driven*, i.e., they require the user to point at objects and then show more or different information. They also imply that the user's idea of relevance is tightly coupled with the physical layout of the display. But this is not necessarily the case, especially when the user wants to query the data for information that is not immediately visible.

Cue methods point out information in the context of other objects, but without really adding a display dimension. When the route from A to B should be pointed out on a map, for example, this could be done by drawing that route with a higher color saturation than the rest of the display. These methods are *data-driven*, they highlight objects or parts of the display based solely on their properties, but not on their layout. This makes very fine-grained control possible of what should be stressed.

Semantic Depth of Field (SDOF) is a new cue method. It was inspired by the depth-of-field effect that has been known for many years in photography. We extended the means of photography by assigning blur not based on distance, but on relevance of objects or data points. Relevance is computed by the application based on a query by the user. A blur level is then calculated from the relevance value, and is used to blur objects on the display. The user can select which relevance function to use, and how to translate from relevance to blur. This gives the user a lot of flexibility in exploring and eventually understanding data.

A model is presented that makes it possible to use existing methods from photo-realistic rendering to create images with blur not based on depth, but on relevance. More efficient techniques are also presented, which can blur objects very quickly using state-of-the-art consumer graphics hardware. These methods allow for interactive applications to use SDOF.

One of these applications is a text viewer which allows the user to search for a word. If the word is found, it is displayed in reverse video, and the sentence it appears in is displayed sharply, while the remaining page is blurred. Thus, it is possible to quickly understand the immeadiate context the word appears in, and at the same time see what other occurences there are, without being distracted.

Another application is a chess board, which makes it possible to display chess games and to point out constellations of chessmen. This makes it possible to see which figure threaten a particular one, or which chessmen cover that figure.

Similar to its use in photography, SDOF can be used to move the focus through layers, such as the information layers in a map. The sMapViewer displays maps and allows the user to focus on one or more layers at the same time, or to stack them on top of each other, so that only the top-most layer is in focus, and the ones below are more and more blurred – thus creating a strong impression of depth and also of relevance.

When analyzing data, scatter-plots are a very important tool. One of our applications is a scatter-plot where blur can be used to distinguish between classes of objects, to get a better understanding of the data.

Using SDOF is possible and effective, as our study has shown. More than that, it has shown that SDOF is a preattentive feature, and can thus be perceived in a very short time without serial search. This is a very interesting feature for a visual cue, because it maximizes the bandwidth to the human brain. SDOF is not significantly slower than color in its perception, which is a very surprising and impressive result. Different to other features, its combination with orientation does not make search times significantly longer, as is the case with all other features.

Summing up, we have put a new tool into the toolbox of visualization application designers that rivals color – something that has not happened for quite some time.

Even though this thesis has shown that SDOF is a useful and effective method, it has shown the need for quite a lot of future work in this area.

One interesting question certainly is which parts of blur (reduced contrast and lower spatial frequencies) are responsible for its effectiveness. It is also possible that only their combination works sufficiently well.

The time dimension also needs to be explored further. We have to find out how long a transition between blur levels has to take, and how its absence affects perceptual performance. A “natural” interval for auto focus (at least for the default) also needs to be found.

SDOF is not limited to information visualization. We want to explore the use of SDOF in Volume and Flow Visualization for integrating F+C there. But also in information visualization, it will be interesting to see how much it can work together and aid other F+C methods, like distortion oriented ones or different kinds of trees.

But SDOF is not even limited to visualization: We also want to explore its possibilites in Virtual Reality. It could be useful for navigation or tutoring in VR.

Still another application area are user interfaces. Guiding the user could be useful not only in help or tutoring systems, but also when a lot of information must be available at the same time, but not too distracting.

The interplay of SDOF an depth cues for 3D visualization also needs to be explored further to find out how well SDOF works for this kind of application.

Chapter 9

Conclusions

The most fascinating part of this work certainly was the user study. It not only meant an elaborate and meaningful evaluation of the underlying idea and a few applications, but also brought us into contact with people from HCI and perceptual psychology, which was a very interesting and rewarding experience.

And the results of this study are also notable: We were surprised that the difference between color and blur was not significant, and that it worked so well (not significantly worse) in conjunction with color. This shows that neglecting blur as a visual cue so far has meant a loss to the science.

Similar to color, SDOF is perceived very strongly as judging or weighting, and therefore is very intuitive if pointing out relevance or importance. What is also similar to color is the fact that SDOF cannot be used for displaying a full data dimension, because users cannot accurately estimate the represented value from blur.

One difference between color and blur is that people are used to changing the color of objects by painting them, etc. But few people are used to changing the blur of objects directly by the means a lens provides, and nobody has yet had direct control over the blur of single objects. This is probably part of the reason why people are having difficulties judging and distinguishing blur levels. An intuitive way of changing SDOF parameters is also very important and at the same time very hard to find.

What is also surprising is that this visual cue – which has been known for about 150 years (since photography became used in practice) – has not been recognized by the (information) visualization community, even though it is visible in many images and practically every movie. The conclusion from this fact should be that more interdisciplinary work is needed not only with researchers from perceptual psychology, but also with artists, who deal with perception in their daily work.

Bibliography

[1] Ansel Adams. *The Camera*. Little Brown & Company, 1991.

[2] Keith Andrews, Josef Wolte, and Michael Pichler. Information pyramids: A new approach to visualising large hierarchies. In *IEEE Symposium on Information Visualization, Late Breaking Hot Topics*, 1997.

[3] Matt Belge, Ishanta Lokuge, and David Rivers. Back to the future: A graphical layering system inspired by transparent paper. In *INTERCHI '93 Conference Companion*, pages 129–130, 1993.

[4] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: The see-through interface. In *Proceedings SIGGRAPH '93*, pages 73–80, 1993.

[5] Alan F. Blackwell, Anthony R. Jansen, and Kim Marriott. Restricted focus viewer: A tool for tracking visual attention. In *Proceedings of the First International Conference on the Theory and Application of Diagrams (Diagrams 2000)*, Lecture Notes in Artificial Intelligence (LNAI), pages 162–177, Edinburgh, Scotland, UK, September 1–3 2000. Springer.

[6] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Extending distortion viewing from 2D to 3D. *IEEE Computer Graphics and Applications*, 17(4):42–51, July / August 1997.

[7] Grace Colby and Laura Scholl. Transparency and blur as selective cues for complex visual information. In *SPIE Vol. 1460, Image Handling and Reproduction Systems Integration*, pages 114–125, 1991.

[8] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. *Computer Graphics (Proceedings SIGGRAPH '84)*, 18(3):137–145, July 1984.

[9] Giuseppe di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Algorithms for drawing graphs: An annotated bibliography. *Computational Geometry: Theory and Applications*, 4(5):235–282, 1994.

[10] Paul Fearing. Importance ordering for real-time depth of field. In *Proceedings of the Third International Computer Science Conference: Image Analysis Applications and Computer Graphics (ICSC '95)*, volume 1024 of *Lecture Notes in Computer Science*, pages 372–379, Hong Kong, December 11–13 1995. Springer.

[11] George W. Furnas. Generalized fisheye views. In *Proceedings of the ACM Conference on Human Factors in Computer Systems*, SIGCHI Bulletin, pages 16–23, New York, USA., 1986. Association for Computer Machinery.

[12] George W. Furnas and Benjamin B. Bederson. Space-scale diagrams: Understanding multiscale interfaces. In *Proceedings of ACM CHI '95 Conference on Human Factors in Computing Systems*, volume 1 of *Papers: Navigating and Scaling in 2D Space*, pages 234–241, 1995.

[13] E. Bruce Goldstein. *Sensation and Perception*. Brooks / Cole Publishing Company, 5th edition, June 1998.

[14] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. *Computer Graphics (Proceedings SIGGRAPH '96)*, 30(Annual Conference Series):43–54, 1996.

[15] Paul Haeberli and Kurt Akeley. The accumulation buffer: Hardware support for high-quality rendering. *Computer Graphics (Proceedings SIGGRAPH '90)*, 24(4):309–318, August 1990.

[16] Helwig Hauser, Lukas Mroz, Gian-Italo Bischi, and M. Eduard Gröller. Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):242–252, 2001.

[17] Christopher G. Healey, Kellogg S. Booth, and James T. Enns. Harnessing preattentive processes for multivariate data visualization. In *Proceedings Graphics Interface '93*, pages 107–117, May 1993.

[18] Christopher G. Healey, Kellogg S. Booth, and James T. Enns. Visualizing real-time multivariate data using preattentive processing. *ACM Transactions on Modeling and Computer Simulation*, 5(3):190–221, July 1995.

[19] Christopher G. Healey and James T. Enns. Large datasets at a glance: Combining textures and colors in scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):145–167, April 1999.

[20] Wolfgang Heidrich, Philipp Slusalek, and Hans-Peter Seidel. An image-based model for realistic lens systems in interactive computer graphics. In Wayne A. Davis, Marilyn Mantei, and R. Victor Klassen, editors, *Graphics Interface '97*, pages 68–75. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1997.

[21] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, January-March 2000.

[22] Jian Huang, Klaus Mueller, Naeem Shareef, and Roger Crawfis. Fastsplats: Optimized splatting on rectilinear grids. In *Proceedings Visualization 2000*, Salt Lake City, UT, USA, October 8–13 2000. IEEE.

[23] Oliver Jennrich. Ein Blick auf die Schärfentiefe, 1999. Available from http://www.astro.gla.ac.uk/users/oliver/articles/Schaerfentiefe.pdf (in German).

[24] T. Alan Keahey. The generalized detail-in-context problem. In *Proceedings IEEE Symposium on Information Visualization 1998*, pages 44–51. IEEE, 1998.

[25] Craig Kolb, Don Mitchell, and Pat Hanrahan. A realistic camera model for computer graphics. *Computer Graphics (Proceedings SIGGRAPH '95)*, 29(Annual Conference Series):317–324, November 1995.

[26] Robert Kosara and Silvia Miksch. Metaphors of movement: a visualization and user interface for time-oriented, skeletal plans. *Artificial Intelligence in Medicine*, 22(2):111–131, May 2001.

[27] Robert Kosara, Silvia Miksch, and Helwig Hauser. Semantic depth of field. In *IEEE Symposium on Information Visualization 2001 (InfoVis 2001)*, San Diego, CA, USA, October22–23 2001.

[28] Robert Kosara, Silvia Miksch, and Helwig Hauser. Focus and context taken literally. *Computer Graphics & Applications, Special Issue on Information Visualization*, 2002.

[29] Matthias Kreuseler, Norma López, and Heidrun Schumann. A scalable framework for information visualization. In *IEEE Symposium on Information Visualization 2000*, Salt Lake City, UT, USA, October 8–13, 2000. IEEE.

[30] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings CHI '95*. ACM, 1995.

[31] Hsien-Che Lee. Review of image-blur models in a photographic system using principles of optics. *Optical Engineering*, 29(5):405–421, May 1990.

[32] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, June 1994.

[33] Marc Levoy and Pat Hanrahan. Light field rendering. *Computer Graphics (Proceedings SIGGRAPH '96)*, 30(Annual Conference Series):31–42, 1996.

[34] Henry Lieberman. Powers of ten thousand: Navigating in large information spaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology '94*, Visualization I, pages 15–16, 1994. TechNote.

[35] Henry Lieberman. A multi-scale, multi-layer, translucent virtual space. In *IEEE International Conference on Information Visualization '97*, London, September 1997. IEEE.

[36] Helwig Löffelmann. Extended cameras for ray tracing. Master's thesis, Institute of Computer Graphics, Vienna University of Technology, 1995.

[37] Helwig Löffelmann and Eduard Gröller. Ray tracing with extended cameras. *Journal of Visualization and Computer Animation*, 7(4):211–228, 1996.

[38] Ishantha Lokuge and Suguru Ishizaki. Geospace: An interactive visualization system for exploring complex information spaces. In *CHI '95 Proceedings*, 1995.

[39] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: Detail and context smoothly integrated. In *Proceedings of ACM CHI '91 Conference on Human Factors in Computing Systems*, Information Visualization, pages 173–179, 1991.

[40] Krešimir Matković, Laszlo Neumann, and Werner Purgathofer. A survey of tone mapping techniques. In *Proceedings of the Thirteenth Spring Conference on Computer Graphics*, pages 163–170, Budimerce, Slovakia, 1997. Comenius University.

[41] Tom McReynolds and David Blythe. Advanced graphics programming techniques using OpenGL. SIGGRAPH 2000 Course 32, Course Notes, 2000.

[42] Harold M. Merklinger. A technical view of bokeh. *Photo Techniques*, May/June 1997. Available from http://fox.nstn.ca/~hmmerk/ATVB.pdf (25-Sep-2001).

[43] Harold M. Merklinger. Scheimpflug's patent. *Photo Techniques*, November/December 1996. Available from http://fox.nstn.ca/~hmmerk/SHSPAT.pdf (25-Sep-2001).

[44] Kazuo Misue, Peter Eades, Wei Lai, and Kozo Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages and Computing*, 6(2):183–210, June 1995.

[45] Tamara Munzner. Drawing large graphs with H3Viewer and Site Manager. In *Proceedings of Graph Drawing '98*, number 1547 in Lecture Notes in Computer Science, pages 384–393. Springer Verlag, August 1998.

[46] Thomas Porter and Tom Duff. Compositing digital images. *Computer Graphics*, 18(3):253–259, July 1984.

[47] Michael Potmesil and Indranil Chakravarty. A lens and aperture camera model for synthetic image generation. *Computer Graphics (Proceedings SIGGRAPH '81)*, 15(3):297–305, August 1981.

[48] Paul Rademacher and Gary Bishop. Multiple-center-of-projection images. In *Proceedings SIGGRAPH '98*, pages 199–206, 1998.

[49] George G. Robertson and Jock D. Mackinlay. The document lens. In *Proceedings of the ACM Symposium on User Interface Software and Technology '93*, Visualizing Information, pages 101–108, 1993.

[50] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees: Animated 3D visualizations of hierarchical information. In *Proceedings of ACM CHI '91 Conference on Human Factors in Computing Systems*, Information Visualization, pages 189–194, 1991.

[51] Jaroslaw R. Rossignac. Considerations on the interactive rendering of four-dimensional volumes. In *Volume Visualization Workshop '89*, pages 67–76, University of North Carolina, Chapel Hill, NC, May 18–19 1989.

[52] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In *Proceedings of ACM CHI '92 Conference on Human Factors in Computing Systems*, Visualizing Objects, Graphs, and Video, pages 83–91, 1992.

[53] Manojit Sarkar and Marc H. Brown. Graphical fisheye views. *Communications of the ACM*, 37(12):73–83, December 1994.

[54] Manojit Sarkar, Scott S. Snibbe, Oren J. Tversky, and Steven P. Reiss. Stretching the rubber sheet: A metaphor for visualizing large layouts on small screens. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Visualizing Information, pages 81–91, 1993.

[55] Cary Scofield. $2\frac{1}{2}$-d depth-of-field simulation for computer animation. In D. Kirk, editor, *Graphics Gems III*, pages 36–38. Academic Press, Inc., 1992.

[56] Mikio Shinya. Post-filtering for depth of field simulation with ray distribution buffer. In *Proceedings of Graphics Interface '94*, pages 59–66, Banff, Alberta, Canada, May 1994. Canadian Information Processing Society.

[57] Ben Shneiderman. Tree visualization with tree-maps: A 2-D space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, January 1992.

[58] Maureen C. Stone, Ken Fishkin, and Eric A. Bier. The movable filter as a user interface tool. In *Proceedings of ACM CHI '94 Conference on Human Factors in Computing Systems*, volume 1 of *Information Visualization*, pages 306–312, 1994.

[59] Anne Treisman. Preattentive processing in vision. *Computer Vision, Graphics, and Image Processing*, 31:156–177, 1985.

[60] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, 2000.

[61] Lee Westover. Interactive volume rendering. In *Volume Visualization Workshop '89*, pages 9–16, University of North Carolina, Chapel Hill, NC, 18–19 May 1989.

[62] Lee Westover. Footprint evaluation for volume rendering. *Computer Graphics (Proceedings SIGGRAPH '90)*, 24(4):367–376, August 1990.

[63] Steven E. Wixson. Four-dimensional processing tools for cardiovascular data. *IEEE Computer Graphics and Applications*, 3(5):53–59, August 1983.

[64] Steven E. Wixson. The display of 3d MRI data with non-linear focal depth cues. In *Computers in Cardiology*, pages 379–380. IEEE, September 1990.

Acronyms, Abbreviations, Variables

Acronyms and Abbreviations

BL	Blur Level
CC	Camera Coordinates
CoC	Circle of Confusion
DD	Data Domain
DOF	Depth of Field
F+C	Focus+Context
IS	Image Space
RI	Relevance Interval
SDOF	Semantic Depth of Field
VS	Visualization Space

Variables

b	blur level (CoC diameter)
b_{max}	maximum blur level
h	step height (standard blur function, Figure 4.3 on page 26)
r	relevance
t	blur threshold (standard blur function)

Acknowledgements

I would like to thank the following people:

Silvia Miksch for letting me go ahead with this project even though it was not exactly a core part of our research ... and also for her support throughout my master's and Ph.D. work – and for supplying me with chocolate when I was down.

Helwig Hauser for making long lists of things to do and trying to make me meet deadlines. And also for letting me steal some useful LaTeX stuff from his thesis ...

Johann Schrammel and Peter Fröhlich of CURE for their enthusiasm and willingness to work with the early, buggy versions of my user study software. And also for making it possible to design, implement, and evaluate a user study in such a tight time frame.

Meister Eduard Gröller for being so enthusiastic about this work, and for being the only person in the world to drink banana liquor.

Markus Hadwiger for his help with OpenGL problems, and for helping me to develop FastSDOF.

VRVis and the Insitute for Software Technology and Interactive Systems for financing the user study for the evaluation of SDOF, which was performed by CURE.

Niki Sahling for looking at 2000 pictures and sorting out the bad ones.

Monika Lanzenberger for proof-reading and for some interesting points.

Dr. Heinrich Tauscher for supplying me with the images in figure 3.10 on page 22 – and for a *very* interesting discussion.

de.rec.fotografie (the German-speaking photography newsgroup) for teaching me a lot about photography, especially technical stuff.

This work is part of the Asgaard Project, which is supported by *Fonds zur Förderung der wissenschaftlichen Forschung* (Austrian Science Fund), grant P12797-INF. Parts of this work have been carried out in the scope of the basic research on visualization at the VRVis Research Center (`http://www.VRVis.at/vis/`) in Vienna, Austria, which is funded by an Austrian governmental research program called Kplus.

Curriculum Vitae

Address

Robert Kosara
Radetzkystraße 21/7
A-1030 Vienna, Austria
Email: robert@kosara.net

Born

February 3rd, 1975, in Graz, Austria

Education

October 1999 to December 2001
Ph.D. studies in Computer Science at Vienna University of Technology
Finished in October 2001
Ph.D. Thesis: *Semantic Depth of field – Using Blur for Focus+Context Visualization*
Supervisors: Silvia Miksch, Helwig Hauser

March 1997 to February 1999
Four semesters of Medicine (unfinished)

October 1994 to July 1999
M.S. studies in Computer Science at Vienna University of Technology
Graduated July 1999 to "Dipl.-Ing." (M.Sc.) with distinction
Master's Thesis: *Metaphors of Movement – A User Interface for Manipulating Time-Oriented, Skeletal Plans*
Supervisor: Silvia Miksch

Job Experience

Since October 1999
Researcher with the *Asgaard Project*, headed by Silvia Miksch

Professional Memberships

ACM (Association for Computing Machinery)
IEEE (Institute of Electrical and Electronics Engineers)
ÖGAI (Austrian Society for Artificial Intelligence)

Publications

Please see http://www.kosara.net/publications.html