Focus+Context Taken Literally

Robert Kosara and Silvia Miksch
Vienna University of Technology

Helwig Hauser
VRVis Research Center, Austria

We present a focus+context method that blurs objects based on their relevance (rather than distance) to direct the user's attention.

A common feature of information visualization applications—and also other areas—is to direct the user's attention to certain objects. This alerts users to a problem or shows the matching objects in response to a query. Often users also want to quickly understand the information pointed out in the context of the other information and not just see the results. This is one type of the focus+context (F+C) technique, which provides both detailed information of the currently most relevant objects, as well as giving users an idea of the context (see the sidebar "Focus+Context").

Depth of field

One method for guiding the user's attention is by blurring the less relevant parts of the display while sharply displaying the relevant information. This method has been used in photography for a long time, where the depth of field (DOF) determines which depth range to depict sharply (for an example, see Figure 1). Thus it's possible to show an object's or person's context by keeping it in the image but showing it out of focus. We briefly describe the theory behind DOF in the sidebar "Optical Basics of DOF."

Using that same idea to blur objects based not on their distance from the camera but on their current relevance in the application makes it possible to direct the viewer's attention. We call this method semantic depth of field (SDOF).

Why semantic depth of field?

Other visual cues exist that can be used for the tasks described in this article, such as color, hue, and so on. But we believe SDOF to be a valuable addition to the visualization toolbox. (For more information on SDOF, visit http://www.ifs.tuwien.ac.at/research/sdof/ and http://www.vrvis.at/vis/research/sdof/.) We can use it when all other cues are already used, to either reinforce another cue or provide additional information. Blur is also intuitive and therefore lets even untrained users quickly understand what's pointed out. Blur also has the advantage of working independently of color, so it can be useful for black and white images as well as for color-blind users.

Visualization is so effective and useful because it uses one of the channels to our brain that has a really high bandwidth: our eyes. But even this channel can be used more or less efficiently. One special property of our visual system is preattentive processing.[1,2] Preattentive

1 **An example of depth of field in photography.**

0272-1716/02/$17.00 © 2002 IEEE

Focus+Context

Information visualization often deals with data of which users have no mental image. A visualization imposes a graphical structure—a mapping from data to screen space—on the data that users have to learn. It's therefore necessary to change this mapping as little as possible; but often there isn't enough space on the screen to display all information with enough detail. Focus+context (F+C) methods make it possible to show more detailed or targeted information—and at the same time, give users a sense of where in the data the zoomed-in, more detailed, or pointed out information is.

We divide the currently existing F+C methods into three groups:

- *Spatial methods*. This is the most prominent group of F+C methods. The image created with an existing visualization is distorted to allow more space for the currently more important objects, and less for the context. Examples for this method are fish-eye views,[1] hyperbolic trees,[2,3] the document lens,[4] stretchable rubber sheets,[5] and other distortion-oriented methods[6] (as they are generally called). One drawback of these methods is that they don't allow control of the degree of interest that's completely independent of the layout of the objects.
- *Dimensional methods*. Users can move a focus over a visualization to display different data about the same objects. These methods don't display more objects, but they allow more or different data dimensions of the already displayed ones. Examples of this type of F+C method are magic lenses,[7] tool glasses,[8] and so forth.
- *Cue methods*. In an existing visualization, objects that meet certain criteria are stressed by assigning visual cues to them so that they are more prominent to the viewer without hiding the context. An example of such a method is to use color saturation and brightness.[9] Another method relevant in this context is a system that lets up to 26 layers of geographical information be displayed at the same time.[10] Users can move the focus between these layers by changing their blur level and transparency.

References

1. G.W. Furnas, "Generalized Fisheye Views," *Proc. ACM Conf. Human Factors in Computer Systems*, ACM Press, New York, 1986, pp. 16-23.
2. M. Kreuseler, N. López, and H. Schumann, "A Scalable Framework for Information Visualization," *Proc. IEEE Symp. Information Vizualization* (InfoVis 2000), IEEE CS Press, Los Alamitos, Calif., 2000, pp. 27-38.
3. T. Munzner, "Drawing Large Graphs with H3Viewer and Site Manager," *Proc. Graph Drawing 98*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, vol. 1547, Aug. 1998, pp. 384-393.
4. G.G. Robertson and J.D. Mackinlay, "The Document Lens," *Proc. ACM Symp. User Interface Software and Technology*, ACM Press, New York, 1993, pp. 101-108.
5. M. Sarkar et al., "Stretching the Rubber Sheet: A Metaphor for Visualizing Large Layouts on Small Screens," *Proc. ACM Symp. User Interface Software and Technology*, ACM Press, New York, 1993, pp. 81-91.
6. Y.K. Leung and M.D. Apperley, "A Review and Taxonomy of Distortion-Oriented Presentation Techniques," *ACM Trans. Computer–Human Interaction*, vol. 1, no. 2, June 1994, pp. 126-160.
7. M.C. Stone, K. Fishkin, and E.A. Bier, "The Movable Filter as a User Interface Tool," *Proc. ACM Computer-Human Interaction Conf. Human Factors in Computing Systems*, ACM Press, New York, vol. 1, 1994, pp. 306-312.
8. E.A. Bier et al., "Toolglass and Magic Lenses: The See-through Interface," *Computer Graphics* (Proc. Siggraph 93), ACM Press, New York, vol. 27, 1993, pp. 73-80.
9. I. Lokuge and S. Ishizaki, "Geospace: An Interactive Visualization System for Exploring Complex Information Spaces," *Proc. Computer–Human Interaction* (CHI 95), ACM Press, New York, 1995.
10. G. Colby and L. Scholl, "Transparency and Blur as Selective Cues for Complex Visual Information," *Proc. SPIE, Image Handling and Reproduction Systems Integration*, SPIE Press, Bellingham, Wash., vol. 1460, 1991, pp. 114-125.

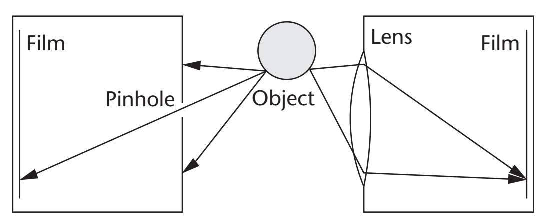
Optical Basics of DOF

In contrast to a pinhole camera (the camera model that's mostly used in computer graphics, especially in visualization), a lens causes more than one light ray to depict every point of an object (Figure A).

The lens equation[1] defines the distance v of the sharp image from the lens, depending on the distance u of the object and the lens' focal length f:

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

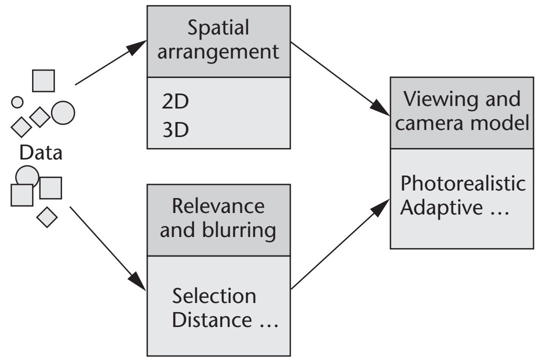
Because the film has a fixed distance from the lens, only points in a plane parallel to the film plane at a certain distance are perfectly sharp. Other points aren't depicted as points, but as small discs, the so-called circles of confusion (CoC). Their size depends on the distance of the point from the focus plane, as well as the focal length and the size of the lens. An object is perceived sharp when the CoC of its projection is smaller than the resolution of the eye (which depends on its absolute size and the viewing distance).



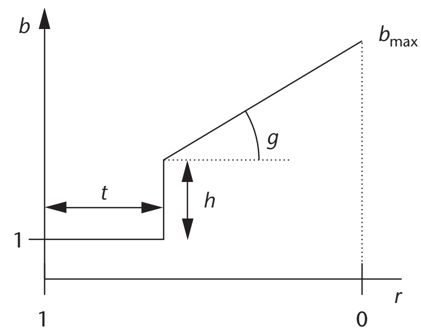
A The pinhole camera versus the lens camera.

Reference

1. H.-C. Lee, "Review of Image-Blur Models in a Photographic System Using Principles of Optics," *Optical Eng.*, vol. 29, no. 5, May 1990, pp. 405-421.



2 The building blocks of SDOF. We arrange objects in the image and assign their relevance to determine their blurring. We present a model for rendering SDOF images with existing computer graphics techniques in Kosara et al.[4]



3 The blur function. The relevance value *r* range from 0 to 1, where 0 means the object is completely irrelevant and 1 means the object is maximally relevant. The relevance values are translated by the blur function into a blur value *b*. The blur function is determined by the threshold *t*, the step height *h*, and the maximum blur diameter b_{max}. The gradient *g* is calculated by the application.

processes take place within 200 ms after exposure to a visual stimulus, and don't require a sequential search. They therefore convey information to the human brain efficiently.

The "User study" section summarizes some of the results of a user study we conducted to show that SDOF is, in fact, preattentively perceived.

Semantic depth of field

The central idea of SDOF is to blur objects based on their relevance. Human perception divides our field of view into foreground and background objects (or preferred and nonpreferred stimuli[3]). This is a semantic division that doesn't depend on the physical positions of objects—for example, closer objects can be considered background objects while more distant ones can become foreground objects. Using SDOF helps the eye with this division: blurred objects immediately become the semantic background while sharp ones stay in the foreground.

We can use SDOF for both 2D and 3D images, but this article focuses on 2D applications—we discuss a 3D application in another work.[4]

Relevance and blur

Apart from the spatial layout of the visualization, relevance and blurring are the main parts of rendering an SDOF image (see Figure 2).

Each object or data point is assigned a relevance value *r* by a relevance function. The application provides this function and calculates the results. The actual function, and how many and which parameters it needs, depends on the application. Some might not allow any interaction at all while others could be very configurable.

There are three types of relevance functions:

- binary functions only classify objects into two classes (interesting and not interesting),
- discrete functions create a small number of classes, and
- continuous functions use the whole range of values.

Our relevance function resembles what others call the degree of interest (DOI) function.[5] But relevance is completely independent of layout (quite in contrast to fisheye views,[5] for example) and isn't directly used for blurring (but is passed through the blur function first).

The value range of *r* goes from 0 to 1, inclusively; where 1 stands for an object of maximum relevance to the current query, and 0 for a completely irrelevant object. For example, a file system browser could assign a relevance of 1 to all files that were last changed no more than three days ago and 0 to all objects that are older. It could let users change that threshold to different minimal ages or switch to a continuous *r* scale that assigns a value of 0 to the oldest file, a value of 1 to the newest one, and interpolates the *r* value between them for all other files. Users can then use the threshold of the blur function to change the visualization's appearance.

After assigning relevance values to data elements, the blur function is used to translate relevance values *r* into blur diameters *b*. The blur function can in principle take any shape, but we found the one depicted in Figure 3 to be sufficient for most cases (see the "Parameterization" section for a discussion).

Properties

Blurring an image or object has two effects. It removes the high spatial frequencies and reduces the contrast. Both effects are direct consequences of the fact that neighboring pixels are summed up—both effects could be reached independently, but the visual impression would be quite different. This also leads to small details getting lost, such as icons possibly becoming unreadable. But this only applies to the context objects, and thus shouldn't be a relevant problem.

SDOF is intuitive. Like a lens in a camera, the lens in the human eye can't show all objects in focus at the same time. But we're seldomly aware of the fact that parts of our field of view are blurred—our visual system ignores these areas, the same way it ignores blurred parts of photographs. We also showed that it is, in fact, a preattentive feature (please see the "User study" section).

SDOF is also independent of color. People can use SDOF

- as an additional cue when color is already used in an application,
- when color is not available (for example, on black and white printers), and
- even if they're color-blind.

SDOF distorts irrelevant objects rather than relevant ones. We believe this to be more useful because users can still see the features of relevant objects without having to adapt to a different kind of display. Only the features of irrelevant objects change and can become obscured. This is also an important feature because blurring makes using icons or other objects with fine details difficult—these features disappear when blurred. But because this is done to the less relevant objects, it's acceptable (and if not, users have to refocus or find a different relevance function).

We can use SDOF to point out objects in any layout. Distortion techniques typically only have one focus and don't deal well with several foci distributed over the display (for example, objects that aren't relevant are also emphasized).

Applicability

Although SDOF is a general method, it's better suited for some applications than others. SDOF suits applications where objects should be pointed out that are of sufficient size so that they don't have to be magnified to be shown to the user. SDOF doesn't work well with pixel-based visualizations.[6]

When there's no knowledge about the output device and no way for users to interact with the application, the use of SDOF can be problematic. This is due to the dependence of the appearance of blur on the viewing angle (see the "Parameterization" section).

But we can apply SDOF in many cases where other visual cues have already been used and additional ones are needed. We can use it when the properties of the output device don't let color, saturation, and other cues be used, or when these cues would interfere with the visualization. We can also use SDOF as an intuitive cue to point users to information or controls in user interfaces.

Parameterization

Because SDOF depends on the output device and the conditions under which it's viewed, we must provide users with a means to adjust the parameters of the display or at least use good default values. Users can adjust the values h and $b_{\max}$ in the blur function (Figure 3), which gives the program the limits of the usable blur. This can be done at the startup of applications, by showing users a blurred and an unblurred image of the same object. Users can adjust the blur level of the blurred object to the smallest level that's still distinguishable from the unblurred object. Users select the maximum desired blur through the same interface. The values obtained can be stored per user so that recalibration is required only when the viewing conditions change drastically.

Once the application also knows the threshold t, it can calculate g. Now the whole blur function is known and can be used to render images. Users can change the threshold t as often as necessary to show different amounts of objects in focus while examining data.

Interaction

A key part of SDOF is interaction. Blurring objects is quite useless if users can't change the focus or see what happens after they've changed parameters. Changing the focus by blurring different objects is also effective in terms of guiding users' attention to the most relevant information. The apparent movement resembles methods used in cinematography, where the focus changes from one actor to another to focus on the person currently speaking, or to make the viewers aware of a person in the background. Therefore, we must animate the change in focus so that users can follow it. The following list enumerates a few interactions that are typical for SDOF applications:

- *Selecting the SDOF dimension*. Users decide which data dimension is mapped to blurring. This may not be configurable in all applications.
- *Selecting the relevance function*. If an application provides more than one relevance function per data dimension, users must be able to select one and possibly set some parameters. Often, the relevance function is an implicit feature of the DOI specification.
- *Changing the threshold*. As soon as the display shows SDOF, users should be able to change the blur function threshold. Users can do this directly on the level of the blur function (changing the t value) or by selecting values in the data domain, which the application then translates to r values for the threshold by the application.
- *Using autofocus*. As soon as users have seen the relevant information, they might want to go back to a sharp display. This is done with the autofocus feature, which brings all objects into sharp focus again—after a certain timeout or after being triggered by users.

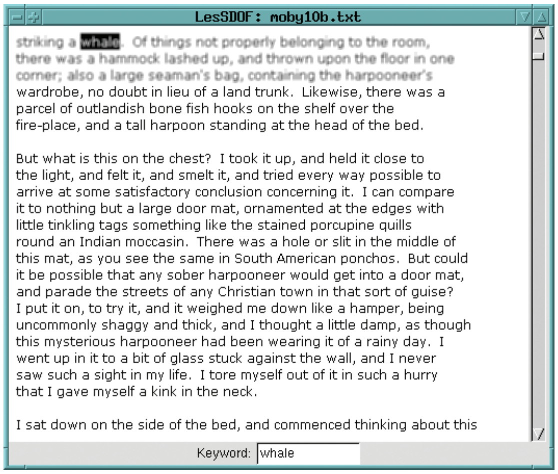
Applications

In this section, we describe four applications to demonstrate the usefulness of SDOF. The first application, called LesSDOF, supports text display and keyword search. The second application, called sfsv, facilitates the viewing of a file system. The next application is a useful tool to get an overview of complex multidimensional data and test hypotheses, which we call sscatter. The last application, sMapViewer, displays different layers of information in a visualization.

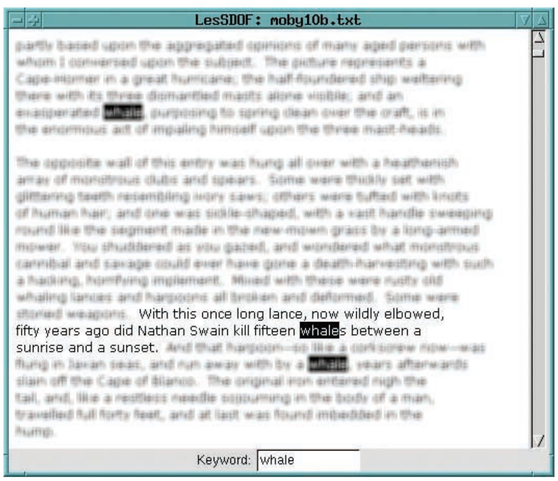
LesSDOF: Text display and keyword search

Displaying text and being able to search for keywords is a very common application. Most applications only show the found keyword (such as using color), but leave it to users to understand the context. Letting users see the whole sentence would help make quicker use of the search result.

The application. LesSDOF displays a text file and lets users scroll through it, much like the Unix program called less. When scrolling a whole page, a few lines are displayed on both pages as context. These lines are slightly blurred so that users understand that this is context information (Figure 4). When searching for a keyword, the found words appear with their foreground and background colors exchanged, and therefore they clearly stand out. The sentence in which they appear is displayed sharply, while the rest of the page is blurred. It's possible to jump between hits and move the focused sentence or show all context sentences in focus (Figure 5).



4 Scrolling in LesSDOF. Three lines on the top are context from the last page, and therefore blurred, but still readable.



5 Finding a keyword in LesSDOF. Three hits appear 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.

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

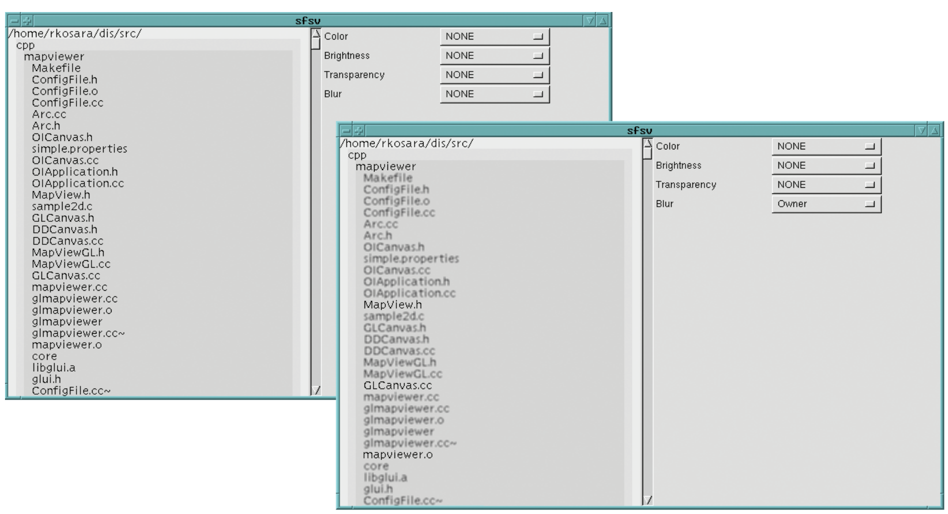
Interaction. In LesSDOF, users can't directly influence either the relevance or the blur function. When paging through a text, the overlapping lines are displayed using the minimum perceivable blur; when showing the results of a search, the irrelevant parts are displayed using the maximum acceptable blur.

sfsv: File system viewer

File system viewers like the Windows explorer are among the most used applications on today's personal computers. Some aspects 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 isn't a solution to this problem, because it destroys a user's mental map.

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). It's possible to do different queries on this data and show the results using different visual cues. One of these cues is blur. So if users select their own files as the focus (Figure 6, bottom right), the files of other users are blurred.

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

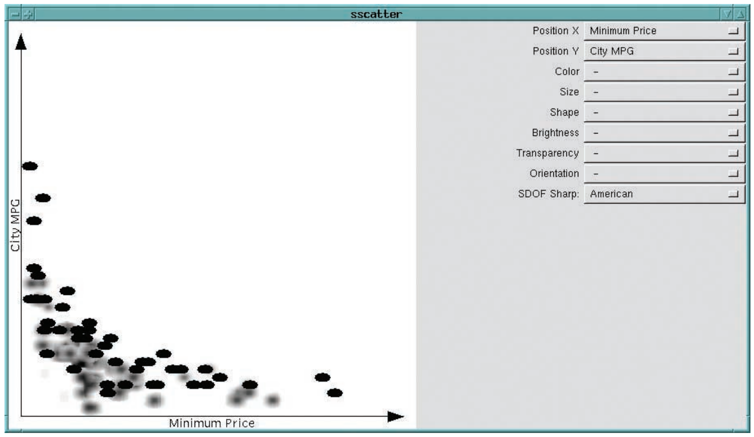


6 A file system viewer with all files in focus (top left) and one focusing on the files of one user (bottom right).

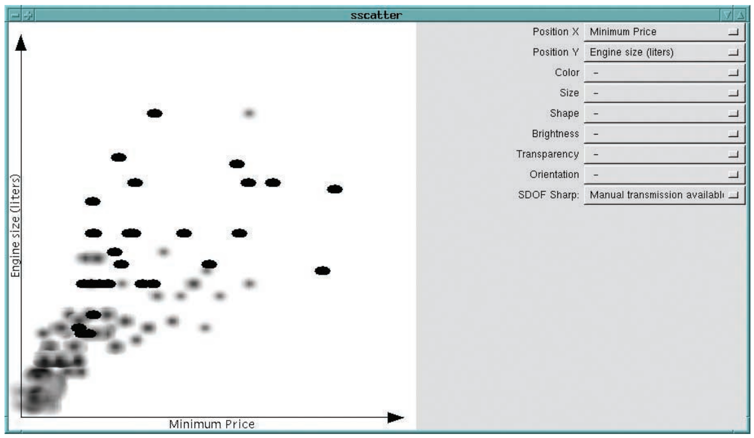
sscatter: Scatter plots

Scatter plots are useful tools to get overviews for data and 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 are therefore needed.

The application. The sscatter program can read data files in different formats whose structure (column delimiters, sizes, names, how many lines per data point, and so on) can be specified in a configuration file. It displays the data in a scatter plot, where users can select which data dimensions are mapped to which visual features. When used on data of car models from 1993, for example, you can see that more expensive cars have lower fuel efficiency and that American and other cars are available over the whole price range (Figure 7). It's also possible to find out that the availability of manual transmission is generally a feature of more expensive cars (Figure 8).



7 A scatter plot 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.



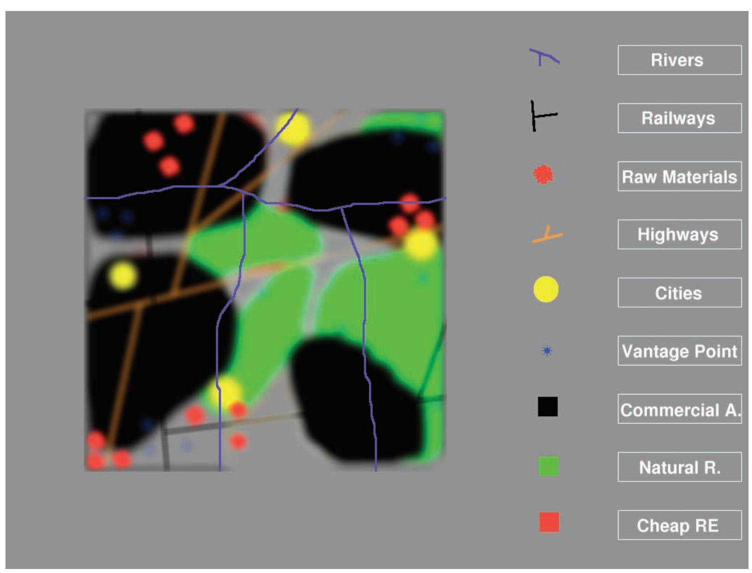
8 A scatter plot of car data showing that more expensive cars have larger engines and that the availability of manual transmission is generally a feature of more expensive cars.

SDOF aspects. Because users are free to choose data dimensions for SDOF display, binary (such as availability of manual transmission), discrete (such as number of cylinders) or continuous (such as price, engine size, and so on) relevance measures can apply. What exactly is needed depends on what users want their new car to be or do.

sMapViewer

When displaying a large number of information layers in a geographical visualization, users must decide whether they want too much or have less—and possibly too little—information visible at the same time.

The application. sMapViewer (Figure 9) lets users stack layers of geographical information on top of each other. The topmost layer is displayed sharply, while all other layers are increasingly blurred. This creates a sense of depth that makes it easy to see relevant objects clearly in their context.



9 sMapViewer showing rivers, railways, raw materials, highways, cities, vantage points, commercial areas, natural reserves, and cheap real estate. The foreground layer of blue lines depicting rivers is sharp, in contrast to the remaining (blurred) background layers.

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.

Interaction. Users can select the layer to be put on top of the stack, thus indirectly changing the relevance of all objects.

Other applications

Other application examples include an SDOF-enhanced 2D and 3D chess tutoring system that can

Table 1. Performance figures for the applications shown in this article.

Program	Size (Pixels)	Percentage Blurred	Frame Rate	Figure
LesSDOF	500 × 400	12	167	4
LesSDOF	500 × 400	92	143	5
sfsv	380 × 480	25.5	31	6
sscatter	600 × 600	37.2	23	7
sscatter	600 × 600	46.5	19	8

point out pieces and constellations on the board, like which chessman covers which others or which pieces threaten a particular one.[4]

It's also possible to alert users to new information in a window by displaying it blurred when no new information is there and displaying it sharply when new data are coming in. This would be especially useful when more than one object can be the source of an audible alarm, for example.

Implementation

One of the reasons blur is seldomly used in computer graphics is that it's slow when done in software on the processor architectures currently used in desktop computers. This is because information must be summed up over an area for every pixel, which quickly adds a load to the application and makes it unusable.

Modern graphics hardware makes it possible to render blurred images quickly and thus makes it usable in interactive applications. The key to the implementation described here is texture mapping. Texture mapping is the central operation on low-cost graphics hardware, and computer games use it extensively. A number of tasks in visualization have been accelerated tremendously by (ab)using texture mapping.

Blurring simply sums up the information around a pixel for every pixel in the image. We can do this by drawing an image several times at slightly different positions and having the graphics hardware sum up the color information at every step. We draw the image into the frame buffer once and then copy it into a texture (this is a fast operation). We then use the texture to draw the image several times, by mapping it onto a rectangle. After displacing the image several times in the x direction, we copy the result of this operation into another texture. This texture is then used when displacing the image in the y direction. We thus produce an image filtered through a box filter.

Because of the limited precision of low-cost graphics cards (typically eight bits per color component), we can't simply add up the image for any blur diameter we want. We therefore have to calculate auxiliary sums that we then sum up in a second step. Because of the filter kernel we use, this actually makes the operation faster. All auxiliary sums (except at the edges) are equal and only displaced. We therefore only have to calculate this sum once and can then displace it (with larger distances in between) several times, thus doing fewer additions overall.

Table 1 gives an overview of frame rates for some of the applications presented in this article. We collected these numbers on a standard PC with a 450-MHz CPU and an NVidia GeForce2 MX graphics card.

It's quite clear from Table 1 that this implementation of SDOF is faster for fewer large objects than many small ones, even if the large objects cover a much bigger part of the display. Most of the time is obviously lost in the setup and communication between the CPU and the graphics card, and not the actual texture operations on the card. Even though the current implementation uses display lists in many places, there's room for improvement, so that the amount of communication needed is decreased even further.

User study

We performed a user study with 16 subjects (male, aged 18 to 25 years, university students, very good vision, basic computer skills) to test the preattentivity of SDOF and to see how useful it was in applications. This section gives an overview of the results of this study, which are covered in more detail elsewhere.[7]

The study shows that SDOF is preattentive. Subjects were able to detect and locate objects after being shown images containing them (and up to 63 distractors, such as blurred objects) for only 200 ms (with accuracies of more than 90 percent, depending on blur level and number of distractors). They were also able to estimate the number of sharp objects after the same time. This estimation was significantly better than chance.

The combination of visual features was also interesting. For these tests, participants could look at the images as long as they needed to find the answer but were asked to answer as quickly as possible. This test showed that there was no significant difference in search time between blur and color, which exceeded our expectations. The conjunction of blur and color ("find the red sharp object") wasn't significantly slower than a simple search for color or sharpness, either. This is also surprising, because conjunctive search is usually slower than simple search. Sharpness and color were also significantly faster than any of these two features combined with orientation, which we expected.

A further test block clearly showed that we can't use SDOF as a fully fledged visualization dimension. It was tiring for participants to try to tell the difference in blur between blurred objects, and they weren't able to tell that difference in any meaningful way. SDOF is still useful for discriminating a small number of classes (three or four—this is subject to further tests).

Conclusions and future work

SDOF lets users literally focus on the currently relevant information. Thus, it's possible to display the results of queries in their context and make them easier and faster to comprehend.

Blur uses a visual feature that's inherent in the human eye and therefore is perceptually effective. We showed that perception of SDOF was preattentive and not significantly slower than color.

The next step is to find out how well SDOF works together with other F+C techniques, especially with distortion-oriented ones. We also want to see how SDOF can be applied to areas such as volume and flow visualization and user interfaces.

We also need to do more user testing with regard to temporal aspects of SDOF—such as how fast to apply autofocus, animating the change of focus, and so forth. Furthermore, we want to continue evaluating the practical use of SDOF in real-world applications. ■

Acknowledgments

We thank the reviewers, and especially the guest editor, for their valuable remarks and suggestions that helped us improve this article.

This work is part of the Asgaard Project, which is supported by the Fonds zur Förderung der wissenschaftlichen Forschung (Austrian Science Fund), grant P12797-INF. Parts of this work has been carried out in the scope of 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.

References

1. A. Treisman, "Preattentive Processing in Vision," *Computer Vision, Graphics, and Image Processing*, vol. 31, 1985, pp. 156-177.
2. C. Ware, *Information Visualization: Perception for Design*, Morgan Kaufmann, San Francisco, 2000.
3. E.B. Goldstein, *Sensation and Perception*, Brooks/Cole Publishing, Florence, Ky., 5th ed., 1998.
4. R. Kosara, S. Miksch, and H. Hauser, "Semantic Depth of Field," *Proc. IEEE Symp. Information Visualization* (InfoVis 2001), IEEE CS Press, Los Alamitos, Calif., 2001, pp. 97-104.
5. G.W. Furnas, "Generalized Fisheye Views," *Proc. ACM Conf. Human Factors in Computer Systems*, ACM Press, New York, 1986, pp. 16-23.
6. D.A. Keim, "Designing Pixel-Oriented Visualization Techniques: Theory and Applications," *IEEE Trans. Visualization and Computer Graphics*, vol. 6, no. 1, Jan.–Mar. 2000, pp. 59-78.
7. V. Giller et al., *Experimental Evaluation of Semantic Depth of Field, a Preattentive Method for Focus+Context Visualization*, tech. report TR-VRVis-2001-021, VRVis Research Center, Austria, 2001, http://www.vrvis.at/.

***Robert Kosara** currently works as a researcher with the Asgaard Project headed by Silvia Miksch, where he is concerned with visualization matters. His main research interest is information visualization, with a special focus on the interdisciplinary possibilities of this field. He received his Dipl-Ing (MSc) in computer science from Vienna University of Technology in 1999. He has just finished his PhD work on SDOF.*

***Silvia Miksch** is an associate professor at the Institute of Software Technology and Interactive Systems, Vienna University of Technology, Austria. Her main research interests are information visualization, planning and scheduling, data and process modeling, temporal representation and reasoning, and task-oriented design and evaluation. Since 1998, she has headed the Asgaard research group. She belongs to the editorial board of* AI Communications *(AICOM; IOS Press) and* Artificial Intelligence in Medicine *(Elsevier).*

***Helwig Hauser** is a key researcher at the VRVis Research Center in Vienna, Austria, leading the basic research there in the field of visualization (http://www.VRVis.at/vis/). His main interests include volume, flow, and information visualization, as well as visualization and computer graphics in general. He graduated in 1995 from Vienna University of Technology (TU Wien), Austria, with a Dipl-Ing (MSc) after studying computer science with a special focus on computer graphics. In 1998, he received the degree of DrTechn (PhD) from TU Wien after completing a scientific visualization research project. He is a member of the IEEE.*

Readers may contact Robert Kosara at the Institute of Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstr. 9-11, E188, A-1040 Vienna, Austria, email rkosara@asgaard.tuwien.ac.at.