Tuning Rendered Line-Drawings

Bernhard Preim and Thomas Strothotte

Department of Simulation and Graphics, Otto-von-Guericke University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany

{bernhard | tstr}@isg.cs.uni-magdeburg.de

Abstract

Normally rendered images are the polished output of rendering software and are, by definition, "perfect". However, as we think about rendering abstractions of the underlying models, rather than photorealistic images, it becomes evident that user involvement is necessary to tune rendered images to his or her satisfaction.

In this paper we outline methods to tune rendered images. We concentrate on line-drawings, as these are most readily edited to make the modelled objects appear more abstract. Rather than having users edit individual lines, our emphasis lies on studying, what users really want to achieve, and how we can support them more directly to attain these effects. This emphasis is used to guide the design of the user-interface to the system for tuning line-drawings.

We present a tool which allows high-level intent-based interaction to generate line-drawings from rendered 3D-models. Included in the system are editing facilities to fine-tune and thereby individualizing line-drawings. High-level interaction draws on statistical methods to distribute lines over the rendered image. We show how the interaction can be improved if information about the model is available to the image editor.

Keywords: Non-photorealistic rendering, line-drawings, architectural sketches, human-computer interaction

1 Introduction

Computer-graphics research has been obsessed with the dream of emulating the effect of a camera pointed at a scene. Ever more sophisticated display processing hardware has, over the years, turned this dream into reality to the point when so-called photorealistic images can be generated from 3D-models. However, there is much more to graphics than photorealism, as Scott McCloud (1993) recently demonstrated when explaining the concept of abstraction in comics (see Figure 1).

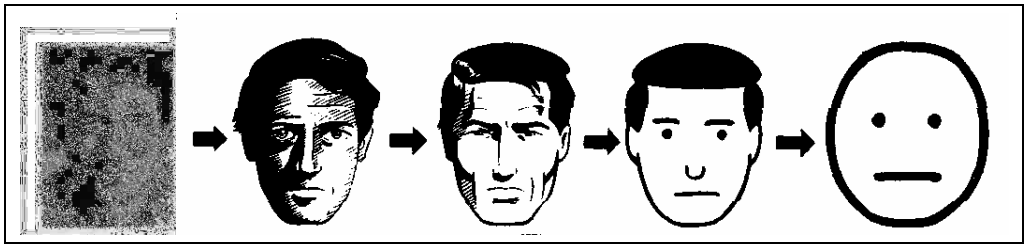

Figure 1: Stepwise abstraction in comics: There is more to graphics than just photos (left). Computer graphics is lacking algorithms to render more abstract images

The picture on the left of Figure 1 is redrawn by hand at different levels of abstraction; indeed there even appears to exist a *continuum* of images, from realistic ones to highly abstract ones. Actually, since there are many different rather abstract images, there no doubt are many corresponding *continuums*.

Our long-range goal is to learn more about the generation of successively more abstract renditions of geometric models. In the present instance we are restricting ourselves to line-drawings, for which there are important applications in CAD. In this area, abstraction plays an important role for several reasons. First, if every line corresponding to an edge in the model were drawn, the resultant image would often become rather cluttered. Second, ”extra” polygons are often encoded in a geometric model as artifacts of the modelling process; drawing all such polygons would mislead the viewer. Finally, abstraction is the way of amplifying the actual message contained in an image.

In this paper we study a particular approach to abstraction in rendering line-drawings. First, we introduce a post-rendering process which reduces the line-density in an image by foreshortening or even removing lines based on heuristics. The important point is that the heuristics do not have the 3D-model at their disposal, but rather work exclusively on the rendered output. Second, we relax this restriction and consider what additional benefit can be achieved when the 3D-model can be exploited as an information source for interaction. We provide tools whereby a user can fine-tune a rendered line-drawing to suit the purposes of the application at hand. The fine-tuning process turns out to use an interesting combination of graphical and textual information.

The paper is organized as follows. We discuss previous papers on non-photorealistic rendering in Chapter 2. Global techniques to tune line-drawings are described in Chapter 3. Chapter 4 describes the sketch-renderer on which our work is based. In Chapters 5 and 6, we first apply the global techniques of tuning to rendered images and then turn to interactive facilities to enable local object-oriented changes. These Chapters also describe the system we implemented and give a variety of examples. Finally, we discuss the usefulness of heuristics and possible extensions of the system in Chapter 7.

2 Rendering Non-Photorealistic Images

Surveying previous work on rendering images which go over and above photorealism shows that work has been done on pixel-based systems to design drawings with filters and other image-processing methods as well as special purpose systems.

Saito and Takahashi (1990) created line-drawings from 3D-models with an emphasis on communicating spatial relations. By contrast, Häberli (1990) worked with scanned images which he processed with special filters to make them look like some artistic style (e.g. impressionistic). Both systems use image-processing algorithms, in addition to which Saito and Takahashi also use information which are stored in so-called G-Buffer from the rendering-process. These G-Buffers contain geometric information (e.g. surface normals) about each individual pixel and are employed to detect and enhance edges.

Another interesting area of rendering pertains to the generation of copper-plates. Leister (1994)) presents a system which is based on a ray-tracer and uses image-processing software to detect edges. These edges form the basis for producing interesting line-drawings.

Tools to design sketches are presented in Salisbury *et al.* (1994) as well as in Winkenbach and Salesin (1994). The focus in Salisbury *et al.* (1994) is on creating flexible hatching techniques and an interface to allow incremental improvement of an illustration. These applications are restricted to 2D-models. In contrast to this work Winkenbach and Salesin (1994) use a 3D-model and generate illustrations without offering the user the possibility to interact with the picture.

Imai and Iri (1988) as well as Hershberger and Snoyeink (1994) present algorithmic solutions to the problem of approximating and simplifying geometric objects with focus on transforming complex polygons into more abstract ones with a given number of vertices or a given error-tolerance. Such techniques can be useful for an abstraction process for line-drawings, although they have not yet been used in this context.

Seligman and Feiner (1991) regard illustrating as a process which is goal-driven and develop style and design rules to create illustrations to fulfill communicative intents which the user has directly specified. In their paper the focus is on generating illustrations with the appropriate perspective and placement of arrows. Although the kind of illustrations they create is totally different from our system the approach to the illustration process is related.

3 Line-Drawings as a Medium of Expression

In this Chapter we describe some effects which can be achieved using line-drawings, irrespective of whether the drawings come from a computer or a human artist. With these observations in mind we later develop techniques to support the user of an interactive system directly to attain some of these effects.

3.1 *Attracting Attention*

We concentrate on drawing techniques for focusing and examined the images in Figure 2, both of the same scene. To understand how we derived techniques to influence the viewer's attention, the interested reader should ask himself the questions: Where do you look first and what do you look at most intensively? Is there a relationship between the techniques applied in the pictures and your impression?

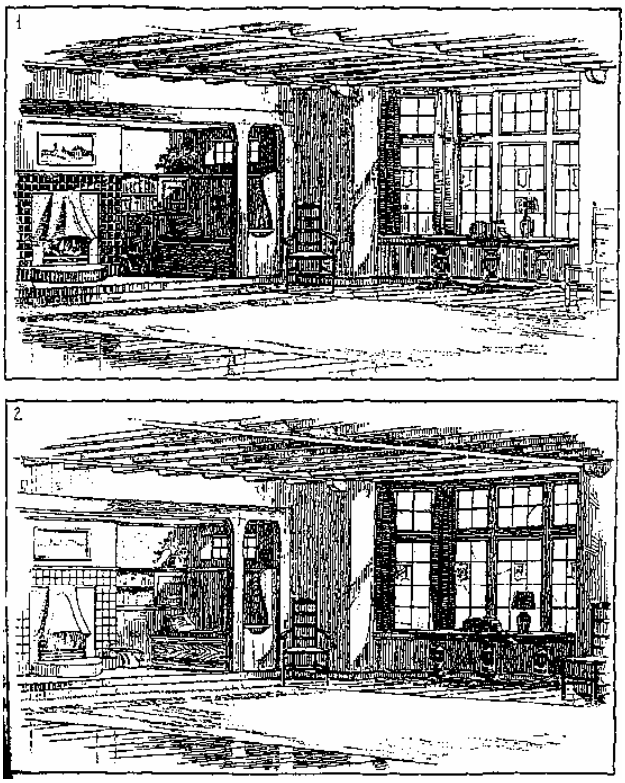

Figure 2: Examples of hand-drawn images - by an artist

The nature of pictures is responsible for the diversity of interpretations. In the upper picture, the hatchings on the left side are denser than in the right picture, whereas the relation is vice versa in the lower picture. Furthermore, the left side of the upper picture is modelled with more detail (see, for instance, the fireplace), while the opposite is true for the lower picture. The dense hatching and the amount of detail influences how viewers perceive the picture. Generally viewers of the upper image which focus on the left part of the room, while viewers of the bottom image will focus on the right part of the room.

In addition to the above-mentioned means we found that thick ### or multiply-drawn ### contour lines attract the user, reinforcing a dramatic effect. Using these heuristics to generate pictures can lead to the desired effect, even though a user cannot be forced to focus on a particular part. Experiments of Schumann (1994) showed that observers tend to regard the different amount of detail and the strength of contours as indicators for a lighting configuration.

3.2 *Amplifying Effects*

Abstraction is a process by which an image is simplified so as to amplify a certain effect. There are at least two reasons to do this. First, the amount of detail should be adapted to the available screen space; a small image generally should not contain as much detail as a large one of the same object. Second, the amount of detail should be dependent on the focus within a picture. To allow explicit abstraction, we classify the lines in a picture and sort them to decide which ones could be left out on a certain level of abstraction. The main question is how to generate a picture with a limited number of lines which shows the most important features while still looking "natural". There is a relatively close relationship between focus and abstraction: Looking again at Figure 2 shows that more detail in one part of the picture is connected with the simplification of other parts.

4 Sketch-Rendering

We now turn to practical aspects of rendering line-drawings. The system we developed to support focusing and abstraction is based on the sketch-renderer which was presented in Strothotte *et al.* (1994). This renderer processes 3D-CAD models encoded in the ".rib"- format of the Pixar Renderman®. Over and above this unstructured format

these models are enriched with object definitions which group several polygons belonging together. These object definitions can be nested so that the *modeller* (the person who creates the model) can construct an object hierarchically. The renderer produces not only a projection of the 3D-model to the screen, but in addition provides a list of visible lines (vectors) which can contain pointers to the objects to which they belong. This additional information binds together the rendered image and the model for use in algorithms. This relation allows powerful means to interact with the graphics.

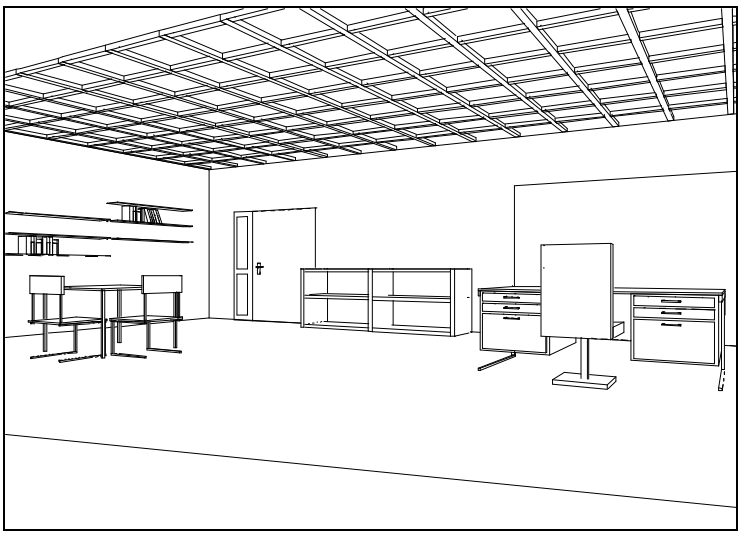

Figure 3: An office room with all lines available produced by the sketch-renderer

The images the sketch-renderer produces immediately after loading the model (see Figure 3) are the basis for the tuning process. In the system of Strothotte *et al.* (1994), this tuning involves the selection of individual objects and the assignment of line-styles to them.

In the system presented here, we extend the editing facilities to combine object-oriented editing (editing based on object-information) with picture-oriented editing (editing exclusively on the picture). Furthermore, we provide additional interactive features. While in the original sketch-renderer the set of visible lines is invariant (only the apperance can be changed), the focus here is on statistical methods to leave certain lines out to support visual effects.

5 Global Tuning Techniques

Based on the heuristics from hand-made drawings, we developed algorithms to change a picture's apperance with a a small number of parameters. These algorithms bridge the gap between the intention of the designer and the basic operations, and can be combined with fine-tuning facilities to be described later in Chapter 6. The prototypical system was implemented on a PC and runs under MS Windows. The following experiments were done on the office room (recall Figure 3).

5.1 Influencing the User's Attention

In accordance with the observations from Chapter 3, we use the amount of detail and the strength of lines to influence which parts of a picture are accentuated more thouroughly. The user can specify directly the focus via mouse-click or alternatively by selecting a direction.

We experimented with different strategies how to handle the lines in the focusing process. It turned out that the question whether to draw a line or to leave it out is not appropriate, as it leads to too severe discontinuities. It is necessary to differentiate more strongly and to display foreshortened lines. This leads to the following classification of lines: *left out*, *foreshortened*, *drawn in original length*, or *reinforced* (by drawing it thicker). To judge, which lines to draw and which to leave out, we need a measure as to which lines are more important than others and a way to calculate this measure based on objective criteria. This measure we refer to as the *relevance* of a line and define it as a function of its length, the distance to the focus and the density of lines in the environment. The calculation of the relevance and its consequences on the appearance of a line are shown in Figure 4. With this formula, long lines and those near to the focal point are highly relevant. If there are many lines in the vicinity of a given line (more than average), the relevance is reduced; by contrast it is increased if the line is in a sparse part of the picture.

L Length of Line

E Distance to focus

D_{loc} local line density

D_{glo} global linedensity

v variable, depends on global parameters

$$Rel = \frac{L * D_{loc}}{E * D_{glo}} * c$$

Rel Relevance of a line p Random number from [0..1]

Rel << p Line **not** drawn at all

Rel < p

Rel > p

Rel > 2p

Rel >> p

Figure 4: Relation between relevance and appearance of a line

Figure 5 shows an example of two images of the office scene, one with the focus to the left and one with the focus to the right. To design these images, the user only specified the focal point by an appropriate click of the mouse. For the influence of the focus on the contour and the amount of detail, the default values were used.

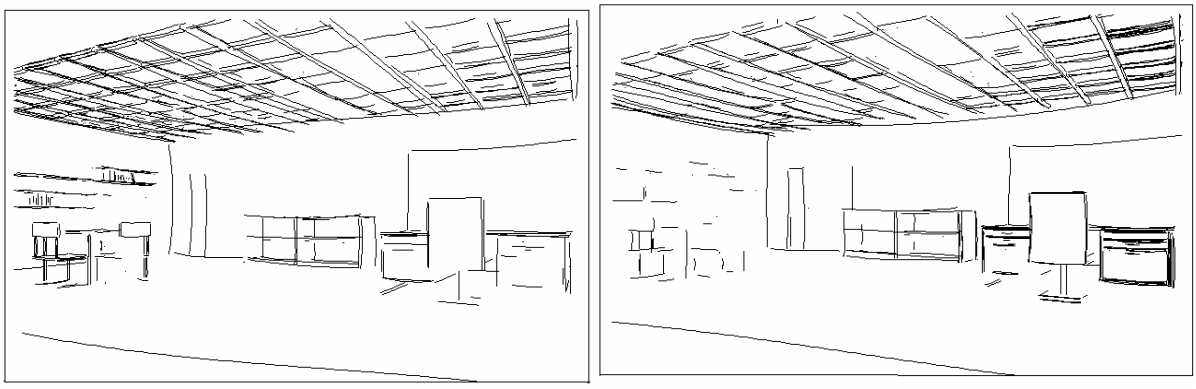

Figure 5: The office room with more detail allocated on the left (left image) and right (right image)

5.2 *Techniques for Simplification*

When applying the described focusing techniques, some lines are left out and some others foreshortened. The user can specify to what extent the amount of detail changes with respect to the focus, but he cannot specify the overall amount of detail which is necessary for abstraction. To adjust this amount of detail, we implemented a dialog which allows a user to move sliders to indicate the amount of detail to be allocated. These sliders represent the number of lines and the overall length of all lines. They can be moved from a minimum (0%) where nothing is displayed to a maximum (100%) where all lines which are produced by the renderer are also presented in their full length. The movement of the sliders results in a more or less abstract picture. This way the user has control over the amount of drawing resources used to display a picture.

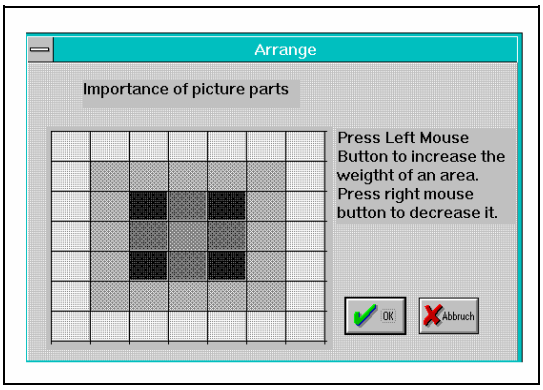

Figure 6: Dark positions indicate important parts in the picture (Empirical function based on the rule of the *Golden Section)*

To restrict the lines in a scene in a meaningful way, all lines are sorted according to their *relevance*. In addition to the formula in Figure 4, we consider the position within the picture as a factor which influences the relevance of lines. According to the rule of the *Golden Section*, we empirically developed a function which assigns a real number between 0 and 1 to all points in the picture. Points near the borderline are assigned minimal values to, whereas points near the four dark points get maximal values. The visualization of this function (see Figure 6) is necessary to understand how the position influences the display process and a prerequisite for direct manipulation of this function. The process of displaying a simplified picture with reduced resources is shown in Figure 7.

Limited Resources to Show A Simplified Picture

```
Ask how much to reduce the length
Ask how much to reduce the line number
Sort all lines with respect to screen position, local density and line length
Initialization for the drawing process
while(not Finish)
  Draw the corresponding line
  Update how many resources are allocated (length, pixels, ..)
  Check whether the limits are exceeded
   if(AllocatedLineLength > LengthLimit)
      Finish
   if(AllocatedLines > LineLimit)
      Finish
```

Figure 7: Displaying a simplified picture with reduced resources

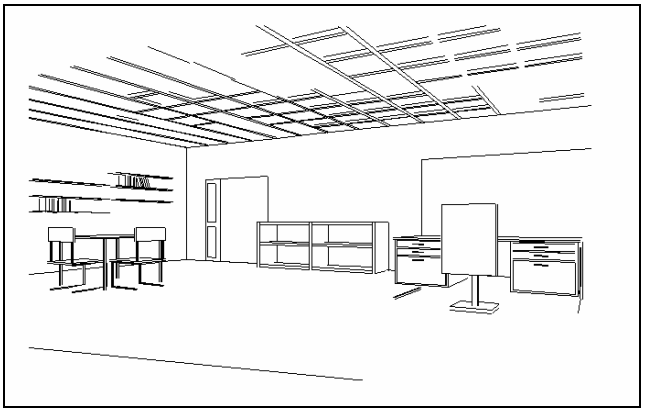

Figure 8: The office room with the total line number reduced to 30%

The result of such an abstraction process as described in Figure 7 can be seen in the picture in Figure 8. This picture was generated moving the slider to set the line number to 30%. The influence of the position within the picture can be seen in the left part of the bookshelf and in the beginning of the ceiling, where more detail is presented than in the right part of the ceiling. Whilst of the ceiling, which produces more visible lines than any other object, 90% were left out or foreshortened by the algorithm of Figure 7, the lines of the right bookshelf (which is the object with the minimum of visible lines) are almost not affected by the reduction. Indeed, 70% of the lines of the right bookshelf were drawn in the original length.

Combining Abstraction and Focussing

According to the observations from Chapter 3, abstraction and focusing can be combined to specify a degree of abstraction with emphasis on a certain part. To do this, the user can take the grey-level-picture (recall Figure 6), specify a focus (which leads to a changed grey-level-scale) and fine-tune the grey-level image. This presents a comfortable way to control the relevance of picture parts, however, the obvious disadvantage is that the manipulation is not directly on the picture. Figure 9 shows two images of a simple street scene with two cars meeting. The images were simplified by setting the overall line length to 50% compared to the total line length of the originally rendered output. After reduction, different picture parts were emphasized (see the grey-level picture which visualizes different focal directions).

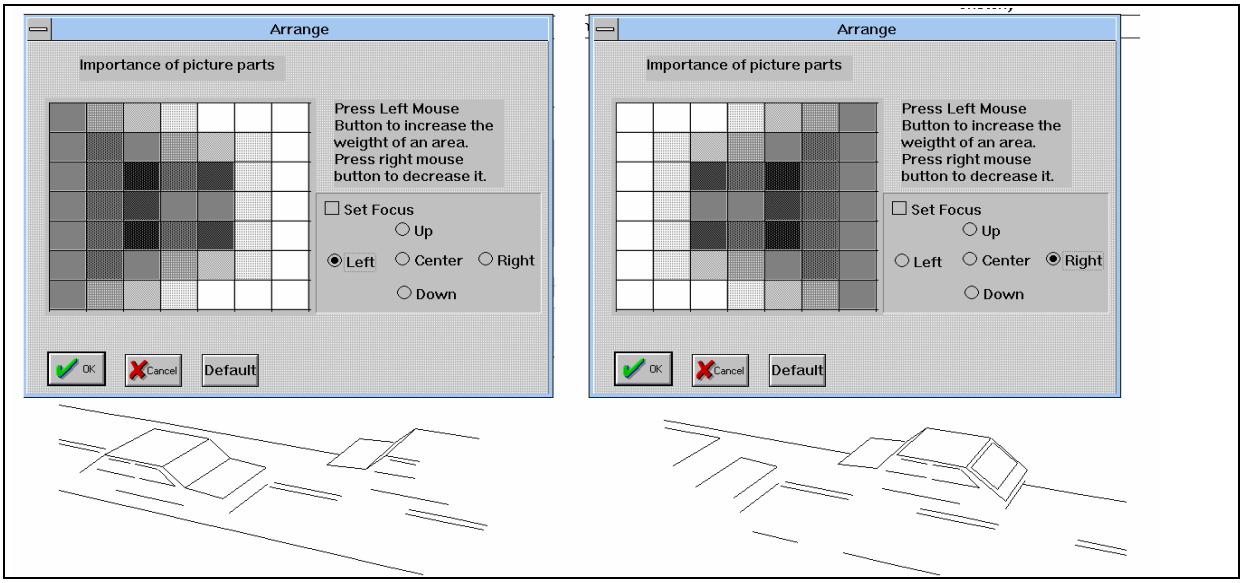

Figure 9: Simplified pictures with a focus, the overall line length was reduced to 50%

When generating simplified drawings, the question arises how many lines are necessary to display objects so that they remain recognizable. Taking into account the object structure of the model, we collect statistics about how many lines an object has after simplification, how long they are and how many of them are connected. These statistics can be used as a basis to determine whether an object is unrecognizable by a viewer. Naturally the question whether an object is recognizable cannot be decided exactly but the statistics lead to a rough estimation. The process of allocating resources can be modified to take the recognizability into account. Different strategies are possible, e.g. to try to prevent the drawing of unrecognizable objects or to try to allocate enough resources for all objects.

6 Interacting with Line-Drawings

The previous chapter described global techniques which are applied to the picture as a whole. While interesting pictures can be generated this way, interaction facilities are necessary to allow individualization. These interaction techniques can be used for a fine-tuning process.

Several issues arise in this context. The first is how to define what should be changed. How can a user describe the small changes, so that the computer can carry them out? The second concerns the operations to perform and the relation between several edit operations in overlapping areas. Because smooth transitions seem to prevail in hand-drawn pictures, we aim at a continuous transition between the image area changed and its environment.

6.1 *Object-Oriented Editing*

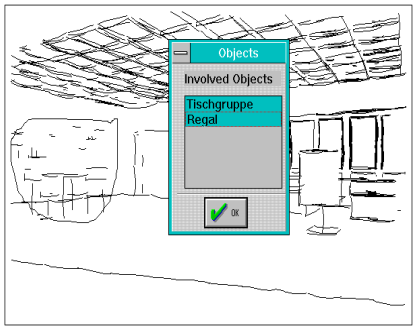

Figure 10: A region marked with the pen and the objects involved

Whilst some changes can be carried out easily on the picture (by marking a region with a pen), other editing-operations could be performed better on the level of the objects involved. Because mutual occlusions between the projections of different objects can occur, it is not always possible to mark a region which contains all the lines of exactly one object. To exploit the object structure contained in the model, we present a list of all objects involved in a dialog box after marking a region. This list can be used as an additional filter.

The next editing operation refers to all lines which are in the marked region and which belong to objects that were tagged. Figure 10 shows an example of a drawing with a sketchy linestyle. On the left side a region is marked for editing and a list of involved objects (in this case the table and the bookshelf) are presented. Per default all involved objects are tagged (no filter).

To extend this object-oriented approach the object-structure (especially the hierarchy of objects) could be made transparent in the form of a structure-browser. Schleich and Dürst (1994) describe structure browsers and employ them to allow direct manipulation on the level of objects. Such a feature would be desirable in our application but has not been implemented.

6.2 *Editing Operations*

The edit operations are selected such that everything that can be changed in the picture as a whole can also be changed for a part of a picture. Editing operations can be classified basically in two categories: absolute and relative changes. Absolute changes are independent of the current appearance of the picture. An example for an absolute change is defining the thickness of contour lines with a numerical value.

Relative changes are incremental taking into account how a picture currently looks. Therefore, they are important with respect to our design principle that the system should make a suggestion which is adapted and modified by the user. In accordance with the above-mentioned considerations we concentrated on the following operations:

- Modifying the amount of detail *(no details, less detail,more detail, all details)*
- Modifying the strength of lines (*stronger, much stronger, less, much less*)
- Controlling the allocation of resources for drawing (*reduce lines*)
- Changing the line-style (*select a new line-style*)

The modification of the amount of detail is carried out by changing the relevance (cp. Figure 5) of all lines involved. That is, the relevance is multiplied by a factor above 1 if *more detail* is specified for a region the line belongs to and below 1 if *less detail* is specified. See Figure 11 for an example of a picture with an edit operation and its result.

Combining Edit Operations

The process of fine-tuning normally consists of a series of edit operations. If a certain part of a picture is involved in several edit operations, the question is how these operations influence each other. Operations of different categories are independent of each other (basically those which are in different lines, e.g. the modification of the amount of detail and the modification of the contour) and therefore don't influence each other at all.

By contrast, operations of the same category influence each other in one of the following ways:

- Relative operations reinforce each other (e.g. the sequence *more detail, more detail*)
- Relative operations neutralize each other (e.g. *more detail, less detail).*
- Absolute operations (e.g. *all details*) overwrite all previous operations of the same category not considering whether they are relative or absolute (*no details, more detail*).

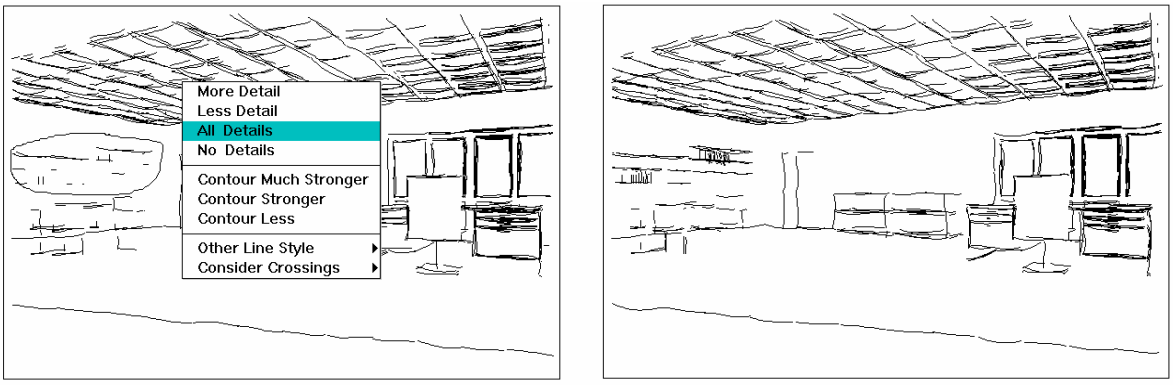

Figure 11: An edit operation (left) and its result (right). The user has requested that *all details* in the specified region be displayed.

6.3 *Smooth Transition*

One important feature of line-drawings is that changes occur continuously over the space of the image. Based on this observation, we aim at smooth transitions between changed parts of a picture which have been changed and its environment.

While it is easy to provide a smooth transition for modifications of the amount of detail and for the strength of contours, we have no smooth transition between different line-styles. Smooth transitions can be realized taking into account the percentage of a line which is inside a marked area. A selected editing operation is applied to all lines which are fully inside the corresponding marked area. The operation is weakened slightly for those lines which belong only partly to the marked area.

6.4 *Display Process*

Figure 12 summarizes the process of displaying a picture taking into consideration local and global tuning-operations. The contour factor (the thickness with which a line is drawn) and the relevance are both initialized depending on the focus and the global parameters (which specify how strong should the contour and the amount of detail vary, which factors are to be considered).

```
for each visible line do
  contfac = calculate contourfactor
  rel     = calculate relevance
  for each edit operation do
    modify contfac
    modify rel
  end
  drawline(contfac,rel)
end
```

Figure 12: Display process (including fine tuning)

These values are modified if the line belongs ### at least partly ### to regions which are edited. If several editing operations were performed on the line, they are combined as described above. Finally, the ”drawline” procedure decides how to draw the line. According to Figure 4, the line is either reinforced, drawn (with normal contour) in the original length, foreshortened or left out.

7 Conclusions and Future Work

We presented a system which supports the user to achieve some effects which are typical for line-drawings. To do this we developed algorithms to make suggestions for a modified image, concentrating on such effects which could be realized better in line-drawings than in pixel-oriented graphics. The interface developed allows a user to create interesting pictures easily which would be hard to draw by hand. Further, we provide flexible fine-tuning mechanisms and enable smooth transitions between parts of an image.

A number of areas are opened up for future work:

Enhancing the Tool to Improve the Design Process in Modelling

Looking at the process of tuning as a whole, there appears to be a relatively strict separation between global parameters influencing the picture as a whole and local fine-tuning operations. The first step is to adjust global parameters, after this the interaction to fine-tune the image can take place. Changing again a global parameter would result in a new picture for which all operations of the fine-tuning are neglected. If global settings are made (e.g. referring to a different line-style or hatching style) and local changes (e.g. of linewidth) are performed, it seems to be possible to make a global change to the amount of detail which does not conflict with the previous changes.

It remains an open question to what extent this obvious drawback can be overcome, whether the tuning process can be designed as a kind of loop consisting of (arbritarily) mixed global and local changes. One point necessary to come closer to a design cycle could be to ”undo” not only fine-tuning operations but also global settings. Further work should be concentrated on this aspect to allow as many cycles as possible in the design process.

The way our fine-tuning process works is not very intuitive for a designer who is accustomed to work directly with the picture. Employing movable widgets ### like the Magic Lenses of Bier *et al.* (1993) which aim at exploring photorealistic images and texts ### could make the fine-tuning closer to the way designers work.

Editing by Drawing

The methods of tuning images described in this paper are text-oriented: The user has to input commands, to specify parameters or to select items from a menu. Instead designers often prefer to draw some lines or hatchings themselves and to tell the system to draw the rest this way. We carried out first experiments in this direction (Preim (1994)), for instance to recognize the line-style the user himself created using a pen. This line-style recognition is the prerequisite for enabling the system to continue the hatching process a user has initiated drawing a few lines. The techniques developed in this context are not very sophisticated; future work should concentrate on extracting more characteristics from user input and the modification of data structures for line-styles and hatching styles to store user-defined styles.

Curved Surfaces

The examples used in this paper are from the area of architecture, meaning that we could get away with mainly flat surfaces. To be able to handle other applications in which natural forms and hatched regions are important, we have to be able to deal with curved surfaces. In this context we will combine the techniques for picture reduction with polygonal approximation. To hatch curved surfaces requires other techniques than the ones employed here; Saito and Takahashi (1990) provide a good start to deal with those hatchings; we are currently building on their work to achieve more flexible fine-tuning mechanisms.

Acknowledgements

We want to thank our colleagues Steffi Fritz, Andreas Raab and Jutta Schumann as well as our senior students Alf Ritter and Stefan Schlechtweg for fruitful discussions. Furthermore, we wish to thank Andreas Raab who implemented the sketch-renderer and modelled the scenes used as examples. Thanks also to Ute Lau, who proof-read the manuscript.

Literature

E. Bier, M. Stone, K. Pier, W. Buxton, T. De Rose (1993).
”Toolglass and Magic Lenses: The See-Through Interface”, *Computer Graphics* 27(3), pp. 73-80

B. Dodson (1985).
Meisterschule Zeichnen - 48 Lektionen zum Selbststudium, Augustus-Verlag, Augsburg

H. Imai, M. Iri (1988).
”Polygonal Approximation of a Curve ### Formulation and Algorithms”, G. T. Toussaint (Ed.) *Computational Morphology ### Machine Intelligence and Pattern Recognition 6*, North Holland, Amsterdam, New York, Oxford, Tokyo, pp. 70-86

P. Häberli (1990).
”Paint By Numbers: Abstract Image Representations”, *Computer Graphics* 24 (4) , pp. 207-214

J. Hershberger, J. Snoeyink (1994).
”An *O* (*n* log *n*) Implementation of the Douglas-Peucker Algorithm for Line Simplification”, *Proc. 10th Annual Symposium on Computational Geometry*, Stony Brook, New York, pp. 383-384

S. McCloud (1993).
Understanding Comics ### The Invisible Art, Kitchen Sink Press Inc, Northampton

J. M. Parramon (1991a).
Wie male ich Licht und Schatten, Edition Michael Fischer, Stuttgart

J. M. Parramon (1991b).
Richtig Zeichnen und Skizzieren, Edition Michael Fischer, Stuttgart

B. Preim (1994).
”Kommunikative Ziele in computer-generierten Skizzen”, Master Thesis, Faculty of Computer Science, Otto-von-Guericke University of Magdeburg

T. Saito, T. Takahashi (1990).
”Comprehensible Rendering of 3-D Shapes”, *Computer Graphics* 24(4), pp. 197-206

M. Salisbury, S. Anderson, R. Barzel, D. H. Salesin (1994).
”Interactive Pen-and-Ink Illustration”, *Computer Graphics* 28(4), pp. 207-214

R. Schleich, M. Dürst (1994).
”Beyond WYSIWIG ### Display of Hidden Information in Graphic Editors”, *Computer Graphics Forum* 13(3), pp. 185-194

D. D. Seligman, S. K. Feiner (1991).
”Automated Generation of Intent-Based 3D Illustrations”, *Computer Graphics* 25(4), pp. 123-132

Th. Strothotte, B. Preim, J. Schumann, A. Raab, D. R. Forsey (1994).
”How to Render Frames and Influence People”, *Computer Graphics Forum* 13(3), pp. 455-466

J. Schumann (1994).
Private Communication (Paper in preparation)

G. Winkenbach, D. H. Salesin (1994).
”Computer-Generated Pen-and-Ink Illustration”, *Computer Graphics* 28(4), pp. 91-100