SURFACES TO LINES: RENDERING RICH LINE DRAWINGS

Stefan Schlechtweg, Bert Schönwälder, Lars Schumann, Thomas Strothotte
Department of Simulation and Graphics,
Otto-von-Guericke University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany
e-mail: {*stefans*|*bert*|*lars*|*tstr*}@*isg.cs.uni-magdeburg.de*

ABSTRACT

Rendering algorithms have tended to restrict themselves to represent the effect of light sources on scenes as could be observed by the human eye. For certain applications, like teaching surgery and anatomy, somewhat more schematic renditions are called for. Such graphics tend to be line-oriented and encode other information than just the effect of light. In the lack of appropriate computer-based tools, such images are practically always drawn by hand by a scientific illustrator.

In this paper, we study techniques for rendering what we call *rich line drawings*. We develop tools for selectively mapping attributes of surfaces of an object onto lines which depict it. This enables us to render images which encode only those properties which are needed for the application at hand.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism.

Keywords: Rendering Pipeline, Non-Photorealistic Rendering, Line Drawings, Shading.

1 INTRODUCTION

Different rendering techniques may encode a wealth of information about an object being depicted in an image. However, in an effort to mimic a (photographic) camera, such images force the encoding of a great deal of information pertaining to lighting. We are interested in systems which reduce the complexity of images in the sense of encoding less information in renditions and thus producing more schematic images. Our main reasons are:

1. A geometric model may not contain all the information needed for a rendition to look natural. The rendition may then look as if it is only half finished.

2. The user may himself or herself want to select which attributes are encoded in the rendition and which ones are not.

Hence, we are searching for ways to encode some (but not all) of the information encoded in a 3D geometry model within a 2D image in a way which, despite the reduction in the amount of information in the image, does not leave the impression that something is missing or wrong with the rendition. The hypothesis is that this will allow the viewer to focus on the information he or she wants without being bothered by extra, unnecessary and unwanted information. This, in turn, is a response to the general problem of information overload in computer-generated presentations.

As a solution to this problem, we introduce in this paper the notion of *rich line drawings*. These are line drawings where a selection of the information which is normally encoded on the surface of an object (such as lighting conditions, surface smoothness and material properties) is mapped onto attributes of the lines to be drawn. In their complexity, such line drawings go well beyond those which would normally be drawn by hand, because a variety of properties can be encoded simultaneously. On the other hand, they enable a convenient way in which the amount of information normally encoded in a shaded image can be reduced:

1. The information encoded on the lines can be reduced to those properties which are needed without leaving the impression that the information not shown is in fact missing.

2. The information pertaining to the surface is approximated when mapping it onto the contour lines, which has the advantage that if the details are not important for the user's application at hand, he or she is not bothered by extra information.

Within this paper, we introduce a rendering pipeline for rich line drawings (Section 2) consisting of two major parts. First, a projection phase which performs the projection of the geometry from three into two dimensions and which encodes the surface attributes in a so-called *enriched 2D model* (Section 3). Second, an interpretation phase where the enriched 2D model is interpreted in terms of mapping the encoded surface attributes onto style attributes of the lines finally drawn (Section 4). Some examples and extensions are shown in Sections 5 and 6.

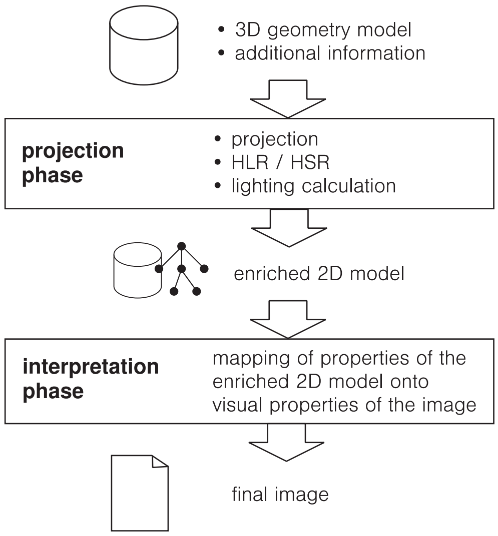


Figure 1: The revised rendering pipeline.

2 THE GENERAL CONCEPT

The term *rendering*, as defined by WATT [Wat93], describes the "overall process of going from a database representation of a three-dimensional object to a shaded two-dimensional projection on a view surface."[1] Involved herein is not only the projection, which transforms the geometry from three into two dimensions, but also the process of depicting three-dimensional phenomena with the means being available in two dimensions.

An explicite decomposition of the rendering process into a projection phase and an interpretation phase of an enriched 2D model offers a flexible treatment of image creation in terms of:

- *pictorial style*—Different interpretations lead to different pictorial styles and thus to different images from one given model.
- *rendering time*—The division in projection and interpretation leads to a decoupling of the stages in the rendering pipeline. The interpretation can therefore be carried out on the enriched 2D model without the need of a new projection, thus saving rendering time.

We therefore propose a rendering pipeline for rich line drawings as follows (see Figure 1).

The three-dimensional geometry model (in our case in a polygonal representation) is obtained from a database and put into the first stage of the rendering pipeline. Here the actual projection takes place as well as the hidden line and hidden surface removal. Besides that, data which would normally affect the scan conversion and lead to an effect on the pixels is encoded in 2D data structures and prepared for use in the second stage. The product of this first stage is the enriched 2D model containing the projected geometry with the following information attached:

- the visibility of the faces and edges,
- lighting information (normal vectors and/or computed intensity values),
- texture properties,
- special properties of surfaces, edges and vertices regarding their smoothness.

In addition, user defined attributes, as for example a classification of objects, may also be encoded in this model. These attributes will be used extensively in the interpretation phase, however, they may also affect the projection phase.

The enriched 2D model is then passed on to the second phase of the pipeline. Here the interpretation of the data takes place in terms of the visual presentation of the given values. It is the user's choice of *which* attributes are to be displayed and *how* this interpretation is performed. Thus a range of images from simple wireframe drawings to sophisticated pencil-like illustrations can be produced based on the enriched 2D model without reprojecting the model.

It should be mentioned that in our case the enriched 2D model exists physically such that it can be represented by an internal data structure as well as in an external file. This means that the interpretation stage might be used completely independently of the projection phase.

This division into the creation of an intermediate model and the interpretation of this model can actually be found in many rendering systems, how-

[1] While the term "shaded" in the sense we are using it throughout this paper means "realistically rendered", WATT uses "shaded" in a more general way describing any rendering style.

ever, the modularization there is in most cases invisible for the user. Some examples using an explicit decoupling have emerged, but without studying it as a process in its own right.

In the system presented by Salesin et al. [WS94], *stroke textures* are used, which are applied in a shading process to depict material and tone. The model serving as base for the application of those textures consists of a planar map and a reference image for tonal values and can be regarded as an enriched 2D model in our sense.

The Piranesi system by Schofield ([RS95] and [LS95]) uses an enriched 2D model, the so called *epixel* (enriched pixel) matrix. Here, based on a pixel-oriented rendering process, information is stored for each pixel which go far beyond standard rendering systems. An enriched pixel contains in addition to the color value for example the z-depth of the corresponding point in the model, normal vectors or material identifiers. This information is then used for interactive drawing tools working on the epixel matrix and producing the final image. Different drawing tools use the stored information in a different way and it is thus possible to create a wide variety of images.

Studying the process of creating and interpreting an enriched 2D model enables us to build a rendering pipeline for rich line drawings which is presented in the following sections.

3 THE PROJECTION PHASE

The main task to perform within the projection phase is the transformation of the given three-dimensional geometry data into 2D. The final result of all processes performed here is the enriched 2D model containing all necessary data for the interpretation phase. For this we need more than just a standard projection of the geometry. Thus the projection phase also includes:

- visible line and surface determination,
- lighting calculations,
- the assignment of further attributes to the output which are computed based on the results of the sub-processes carried out, and
- further processing of the data to match the requirements of the interpretation phase.

It is of great importance to create the enriched 2D model as general as possible to allow a wide variety of interpretations. Thus preserving the computed data is necessary.

In the following we will explore the contents of the enriched 2D model in more detail before we explain the most important steps for its creation.

3.1 The Enriched 2D Model

As the main part, the enriched 2D model contains the projected geometry. The projection itself is performed according to a given camera model which has been specified by the user within the modeling stage. Different camera models can be used, leading to different projections. During the projection process, we preserve all coordinates which are computed, i.e. local , world and screen coordinates. Information for lighting and shading, such as vertex normals, texture coordinates or material properties are also included.

For hidden surface and hidden line removal we currently use an extended version of the analytical algorithm developed by Sechrest and Greenberg [SG81]. This scan-line based algorithm combines the determination of visible lines and visible faces in a very elegant way. It produces a list of visible line segments as well as a list of visible faces (polygons). The only drawback of the algorithm is that it can not handle intersecting models. Here a preprocessing of the models is necessary. Besides the visible part of the model, the invisible lines and faces can also be determined. This information can then be used for instance to draw hidden lines in a special manner, as is sometimes done in CAD drawings. In summary, the process of visible line and surface determination is carried out as splitting the model in visible and invisible parts and marking them appropriately.

Lighting calculations are performed to compute intensity values for specific points of the model. To compute light intensities, an illumination model is evaluated and the resulting values are assigned to the faces of the model.

The connectivity of the model, i.e. information of how faces are assembled to meshes and how meshes finally form objects is useful for displaying the internal structure of an object. This information is given for the initial geometry and has to be computed at least for the visible part of the output.

Putting it all together, the enriched 2D model for our line render pipeline contains the following surface-oriented information:

- the projected geometry with all intermediate coordinates attached,
- information about the visibility of parts of the geometry,
- normal vectors or intensity values for model surfaces, edges and vertices, and
- connectivity information.

3.2 Additions to the Enriched 2D Model for Line Drawings

In order to be successful in the creation of rich line drawings, some additional information has to be computed based on the surface-related data described above.

A list of 2D line segments as produced by the HLR algorithm is not sufficient as a starting point for further computations. Thus a special postprocessing will create *segment chains* from the visible segments with the help of given (or computed) smoothing information. This smoothing information is normally used for interpolative shading algorithms (and thus belonging to the faces of the model) by determining over which faces of a given polygon mesh interpolation has to be performed.

We use this smoothing information in the same way for line drawings. We can say that all faces which are smooth to one another can be displayed in the same style and thus represent an area of common features in the image. As we will see, by displaying the borders of these areas we get an image which highlights the structural properties of the objects to be displayed.

The generation of the segment chains from the projected model edges can be summarized as follows:

1. Determining the smoothness of the model vertices by inspecting all edges connected to this vertex. A vertex is said to be smooth if only one or two of the adjacent edges are sharp (here invisible edges have to be considered, too). This process has to be carried out only once for the model.
2. Selecting the edges which should be involved. This means, for example, eliminating triangulation edges which do not contribute to the final drawing.
3. Creating segment chains by tracing the visible line segments of the model. Here, two sharp segments sharing a vertex are combined. This combination process ends at sharp vertices (or if all segments are used). Care has to be taken at sharp vertices where more than two segments meet. Here smooth segments end a chain, sharp segments are combined.

We now use these chains rather than the original line segments as a basis for our enriched 2D model. These chains are defined as a sequence of vertices connected by the original line segments. We do not lose any data since the projected model edges are used here. All further information computed during hidden line/surface removal as well as lighting information are attached to those segment chains in an appropriate manner.

4 THE INTERPRETATION PHASE

4.1 Expressing 3D Phenomena with 2D Means

Within the second phase of the rendering pipeline, the enriched 2D model is interpreted in terms of mapping attributes contained in this model onto visual properties of a 2D image. For this, means of expressing three-dimensional phenomena in two dimensions are needed. For pixel based rendering, color can be used to display any spatial properties like z-depth as well as lighting and shading, and material properties. Also non-spatial, more abstract information can be encoded by the use of different colors. For line drawings, other means of expression have to be found.

By exploring line art illustrations, especially in the area of medicine and anatomy, we found the following key concepts of encoding 3D information in line drawings:

- Providing at least the silhouette of the objects to display helps to understand the structure of the observed scene. Different levels of detail can encode more information.
- Presenting the model with curved rather than with sharp polygonal borders. This resembles interpolative shading techniques used for pixel-based rendering.
- Encoding lighting by the use of lines of varying width and brightness.
- Suggesting surface properties by hatching or similar techniques.

Those concepts are so far partially realized within computer generated line drawings. For example, G. Elber mapped the z-depth onto width and brightness of rendered lines to achieve depth cueing [Elb95]. More abstract properties, like object importance, were depicted using diffently drawn lines by D. Dooley and M. Cohen [DC90].

Given the enriched 2D model as described above, we are now in a position to build different interpretation modules. As a tool for expressing three-dimensional phenomena within line drawings we developed so called *linestyles* which are described in the following.

4.2 Linestyles

Observing the way line illustrations are created by human illustrators, one can find that a line can be seen as consisting of two parts. First, there is a description of the overall course of the line in terms of geometry as well as line thickness (given by the pressure with which the drawing tool is used) and line brightness (given by the amount of ink left behind by the drawing tool). The second part is the description of the deviations from this general path which are determined by:

- different handling of the drawing tool (for instance regarding the pen angle),
- peculiarities of the person using the drawing tool (for instance irregular hand movement),
- different drawing surfaces (leading to different behavior of the ink).

Based on this observation our model for drawing lines also consists of two parts. The *path* determines the overall shape of the line as parametric curve. Attributes may be attached describing for example the pressure and saturation of a drawing tool. These attribute values finally result in a certain appearance of the line. The *style* determines all deviations of the final line from this general path. Here geometric perturbations as well as changes in the attribute values have an effect. The style, also, is given as an attributed, in this case non-cyclic parametric curve.

The final line is computed as the superposition of a path and a style providing the possibility to express alternative behavior of the line by choosing different styles on one path.

While the function of the path is relatively clear to understand, the style needs further explanation. As already said, the style describes the deviations of the line to be drawn from a given path. It is thus responsible for customizing the appearance of the drawing. The definition of the style curve is based on a straight line segment, the so called *style line*. A style not changing the path at all would geometrically match exactly this style line. Any geometric divergence of the style curve from the style line will be mapped onto the path within the drawing process. Changes in pressure and saturation defined with the style are processed in the same way and influence the appearance of the line.

The path and the style are geometrically described as parametric curves including a set of control vertices, additional weights (if needed) and an interpolation method determining the type of curve used. Non-geometric attributes are defined over the parameter range of the curve either by giving direct values at the control points or by specifying parameter value pairs which are then interpolated. It is also possible to describe these attributes by a general function over the parameter range.

The superposition of path and style resulting in the final line can be computed in different ways. We have examined several possibilities, as for instance computing the arithmetic or geometric mean of path and style positions as well as simply adding the given curves. More flexibility is offered by the so called *difference vector* method.

To map the style geometry onto the path we use the line segment from the style's starting point s_s to the end point s_e (the style line introduced above, see Figure 2). We then compute for a given parameter $t \in [0..1]$ the point $l_0 = s_s + t(s_e - s_s)$. Furthermore, we calculate the point s_0 on the style curve for the same parameter value t and the *difference vector* $v_0 = s_0 - l_0$.

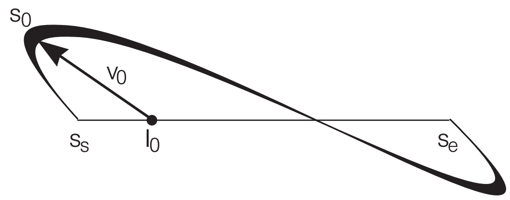


Figure 2: A style definition

To apply the difference vector to the path, it is first rotated to match the style line with the path's tangent at p_0, which is the position on the given path for the parameter t. To get the final point p, this rotated difference vector is added to the point p_0.

The mapping of the geometric properties as described above leads to a *skeleton* of the curve which is to be drawn. In contrast to *skeletal strokes* introduced by Hsu, Lee and Wiseman in [HLW93], where the *flesh* of this skeleton is computed by transforming an arbitrary picture, we create the flesh from the given pressure and saturation values yielding a hull shape for the line to be drawn. This is done by interpolating them over the length of the created skeleton curve. Also here, attribute values given with the style influence those given with the path. This is accomplished by computing the weighted arithmetic or geometric mean of the corresponding values. The user may specify the weight for this computation and thus determining how strong the style's influence is on the path. Finally, the hull shape is filled according to the computed width and brightness values.

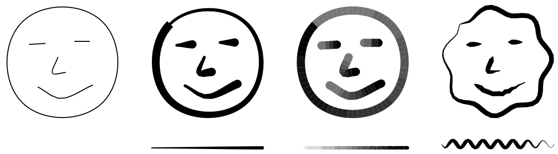


Figure 3: Examples of different styles applied to the same path (left).

Some examples of how different styles applied to one path change the appearance of the drawing can be found in Figure 3. Here the leftmost image shows the path which is used. From left to right, the styles applied change the line width (pressure), the line brightness (saturation), and finally the geometry as well as the line width.

4.3 Interpretation Using Linestyles

The linestyles are perfectly suited to create an interpretation module for the enriched 2D model as described above. We can derive the path geometries directly from the segment chains created from the visible lines. For each chain, we create a path by using the segment vertices as control points. Interpolating them, however, will not exactly match the projected model edges. Instead, the curves will interpolate the model (or approximate it, depending on the chosen interpolation method), leading to more smooth, curved images. After having set up the geometry, there are several methods how the path attributes may be derived from the enriched 2D model, for example:

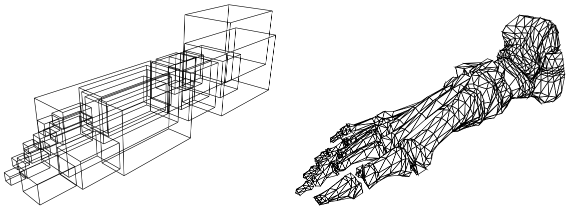
- Using the computed intensity values as pressure attribute yielding lines which change their width according to the lighting situation.
- Using user supplied importance values to highlight any objects in the scene by giving their paths different attributes.
- Simulating depth cueing by mapping the z-depth onto the path's saturation attribute.

Further flexibility is attained if the style is included within the interpretation. Since the style can be chosen independently of the path, it offers possibilities to encode non-geometric attributes in the drawing. As an example, user defined object categories (for medical drawings this could be blood vessels, muscles, nerves, organs, etc.) can be encoded by applying a different style for each category. The combination of path and style is a powerful tool of encoding both, geometric and non-geometric information in one image.

5 EXAMPLES

In the following, a few examples of the line drawings generated with the methods outlined above show the quality of the images thus created. All images are created based on a geometry model of the bones of a human foot. After the projection phase of the rendering pipeline, the enriched 2D model was written to a file. This file was used as the input to different interpretation modules.

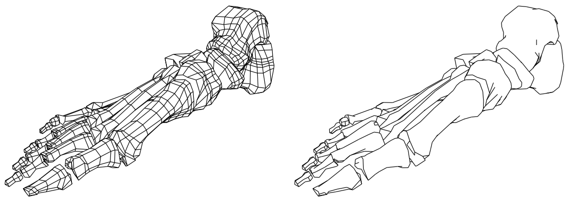
Figures 4 and 5 show how aspects of the contents of the enriched 2D model can directly be mapped on the line drawing. Here only specific parts of the model are visualized by drawing them with regular lines.



(a) The bounding boxes of the model parts giving an impression of the size relations.

(b) A wireframe drawing showing only visible parts of the model

Figure 4: Two simple renditions without using linestyles.



(a) The boundaries of all visible faces (no triangulation edges are shown)

(b) The visible line segments forming the contour of the objects.

Figure 5: The visible faces and line segments serving as a basis for further operations.

Figure 6 shows how the computed light intensities can be used to enhance a line drawing. If a light source is assumed in the upper left corner, all lines depicting darker areas are drawn wider than those showing bright areas. This technique is frequently used within the area of scientific illustration.

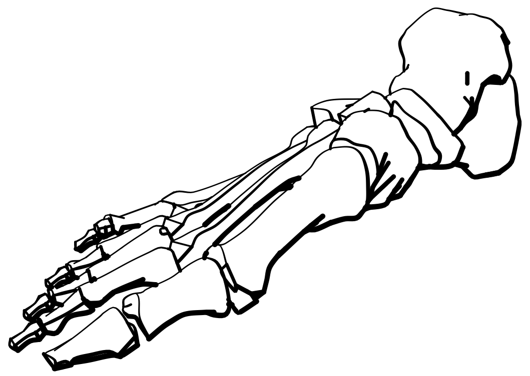


Figure 6: Interpretation of light intensities as line widths (slightly overdone).

A different model containing more objects was used to create Figure 7. Here each material property is mapped onto a different linestyle.

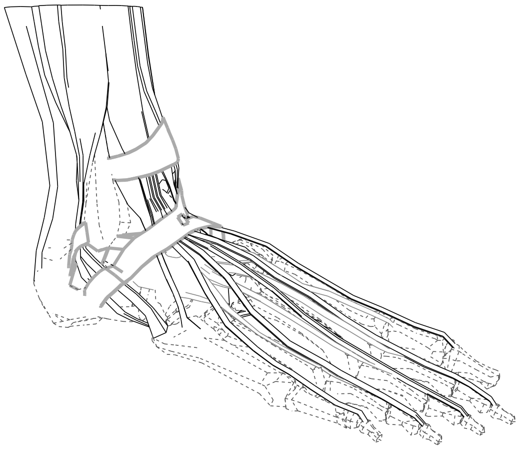


Figure 7: Material properties visualized with different linestyles. Bones are drawn with a dashed, ligaments with a wide brighter style. (Courtesy of Andreas Raab)

These examples already show the variety of images which can be created with the techniques introduced above. The next section shows some advanced renditions where hatching techniques are used to visualize even more surface attributes.

6 FURTHER INTERPRETATIONS

The enriched 2D model as presented in Section 3.1 offers many degrees of freedom which can be used within the interpretation phase. Since we have access to all visible faces in the drawing, we can apply hatching techniques to suggest surface and material properties. Using this in connection with differently drawn lines yields a tool for the creation of highly illustrative images.

Two techniques are especially of interest: *stippling* and *overlay drawings*. Stippling is very common in scientific illustrations. Here form is suggested by graduations of tone. Different tonal values are depicted by groups of dots where the distance between the dots and the size of the dots is depending on the specific part of the surface. A simple stippling can be implemented based on flat shading. For each visible polygon an intensity value is computed. This intensity value determines the distance between the dots. The coordinates of the dots for each face are generated by a function providing randomly distributed but almost equidistant values. An example of this technique can be found in Figure 8.

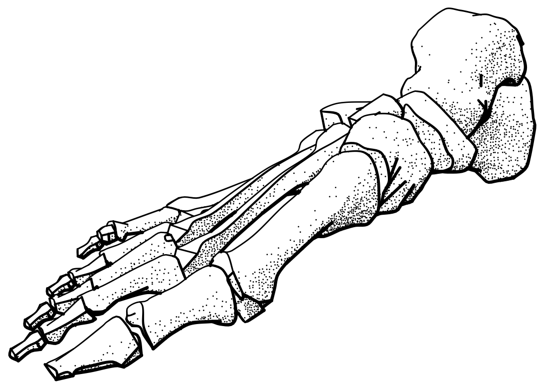


Figure 8: Stippling as a way of interpretation of face related attributes.

An overlay drawing is a combination of a shaded image with a line drawing. Line illustrations concentrate more on the structure of the depicted scene by displaying the contours of objects and sub-objects whereas shaded images are used to depict surface and material properties. If we combine a shaded image and a line-drawing created with the tools described so far, we not only combine the advantages of both techniques but also get even more flexibility and freedom in depicting different kinds of information. Also here, an example can be found in Figure 9.

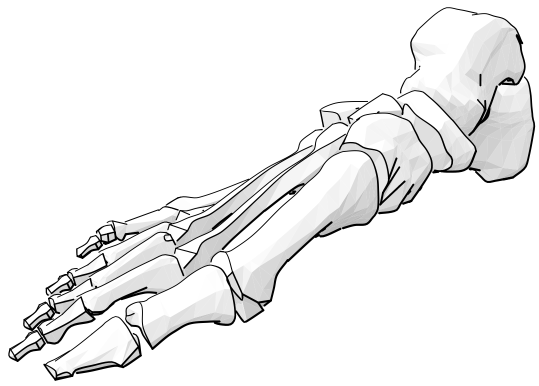


Figure 9: A combination of a flat shaded image and a line drawing.

The final example (Figure 10) shows an extension of the stippling technique. Instead of dots, small lines are used which may depict material properties. In this manner, similar hatching methods can be implemented.

In general, interpretation techniques can be combined. However, it is the user's responsibility of how different properties of the model are interpreted to not create images which use too many different techniques and thus disrupt the user's orientation.

The user's choice in how to interpret the enriched

Figure 10: Another hatching technique similar to stippling.

2D model or parts of it can either be given in a configuration file (which is appropriate for batch rendering) or with an interactive tool. Here for each object or each material contained in the scene the interpretation module which is to be used is chosen. Besides of that choice, additional parameters, as for instance hatching parameters or linestyle properties have to be supplied.

7 CONCLUSIONS

The technique of dividing a rendering pipeline into several independent stages, as we presented in this paper, enables a more flexible treatment of the image generation process. Already the division into two parts, a projection and an interpretation phase, helps to set the stage for creating a wide variety of images from one given model. We argue that a modularization of the rendering pipeline will provide much more flexibility than the use of a number of different rendering systems. With a completely modular renderer, the user may choose the way an image is created by linking different modules together. It should even be possible to mix different pictorial styles by applying an interpretation module only to part of the objects.

As for the line drawings, the linestyles developed offer further options to choose the pictorial style with which an image is created. Their application yields possibilities to visualize surface-related attributes with lines. The mapping of attributes contained in the enriched 2D model onto properties of paths and styles can be customized by the user and is thus a tool for expressing ideas which go far beyond showing the structure of a scene (as would be possible with simple wireframe drawings). The linestyles work completely vector-oriented, i.e. the output is a set of filled polygons which are then sent to the output device. Thus we do not need to do the rasterization, this is relegated to the output device, for example a PostScript printer. Furthermore, by using PostScript, the images can be easily included in documents with any word processing system.

There are a number of open problems and areas for future research. Which information can be extracted from standard geometric models and computed by standard rendering algorithms has to be studied in more detail. Also, we have not examined yet which techniques are possible to further enhance the enriched 2D model based on user interaction. Here we expect further improvement of the image quality.

ACKNOWLEDGEMENTS

The work presented within this paper has been carried out as part of the project "Adaptive Graphical Zoom", reference `STR 398/2-1`, funded by the German Research Society DFG. The authors wish to thank all their colleagues for their helpful discussion. Special thanks to Tobias Isenberg who helped us with the implementation.

REFERENCES

[DC90] D. Dooley, M. Cohen: "Automatic Illustration of 3D Geometric Models: Lines". In: *ACM Computer Graphics*, 24:77–82, 1990.

[Elb95] G. Elber: "Line Illustrations ∈ Computer Graphics". In: *The Visual Computer*, 11(6):290–296, 1995.

[HLW93] S. C. Hsu, I. H. H. Lee, H. E. Wiseman: "Skeletal Strokes". In: *UIST'93 Proceedings*, pp. 197–206.

[LS95] J. Lansdown, S. Schofield: "Expressive Rendering: A Review of Nonphotorealistic Techniques". In: *IEEE Computer Graphics and Applications*, 15(3):29–37, 1995.

[RS95] P. Richens, S. Schofield: "Interactive Computer Rendering". In: *Architectural Research Quarterly*, 1(1), 1995.

[SG81] S. Sechrest, D. P. Greenberg: "A Visible Polygon Reconstruction Algorithm". In: *Proceedings of SIGGRAPH'81*, pp. 17–27.

[Wat93] A. Watt: *3D Computer Graphics*. Addison Wesley Publishing Company, 1993.

[WS94] G. Winkenbach, D. H. Salesin: "Computer-Generated Pen-and-Ink Illustration". In: *Proceedings of SIGGRAPH'94*, pp. 91–100.