The Visual Computer

Dig^{i}Dürer – a digital engraving system

Yachin Pnueli
and Alfred M. Bruckstein

Computer Science Department,
Technion, Israel Institute of Technology,
Haifa, Israel

A variety of halftone methods for reproducing gray-level images on bilevel display media have been proposed. The output of even the best of these falls far short of the quality achieved on similar resolution media in man-made engravings. We introduce **Dig^{i}Dürer** – a digital engraving/halftoning system. For a gray-level image input, the system produces a bilevel (black and white only) picture, which has the appearance of an engraving of the input. The system produces high-quality output, even when the graphic elements of the halftone are visible, and both the quality and style of the output can be improved or customized either by using further information on the image content or by interactive user intervention. The heart of **Dig^{i}Dürer** is a curve evolution algorithm generating halftones by controlling the density of line elements, which are the level contours of a potential field induced by the image via an Eikonal equation. Since the basic version of the system produces rather rough results, further capabilities were added to allow for user intervention and the use of image content information. Extensions to graphic elements other than lines are also feasible.

Key words: Digital halftone – Bilevel display – Curve evolution – Expert systems

Correspondence to: A. Bruckstein

1 Introduction

A picture is worth a thousand words. Part of the reason is the seemingly infinite variety of tones one can find in it. However, some of the most important mediums for the display of pictures are bilevel ones. The most prominent of these is the printed medium, where every point in an image is either black (with ink) or white (no ink). Other such mediums include many types of terminals that are either bilevel (such as LCDs or monochromes) or else support only a limited variety of shades. To display a gray-level picture on such a medium we must use a process called *halftoning*, which transforms the original into a bilevel, binary picture that can be displayed.

There are quite a few strategies and algorithms for digital halftoning. Ulichney (1987) lists the most important of them: clustered ordered dither, dispersed ordered dither, and error diffusion techniques (Fig. 1a–d). Other methods include those of Knuth (1987), Kollias and Anastassiou (1991), Peli (1991), Sullivan et al. (1991) and more. All these methods have several properties in common:

- They are inherently digital – that is, they can be viewed as transformations of pixels into pixels.
- When the pixel grid is fine (so that individual pixels cannot be seen) many of them achieve a quality that is between acceptable and excellent.
- When the grid is coarse (so that individual pixels are noticeable) the quality is considerably degraded.

A nondigital method that in essence shares these properties is the *halftone screen* (Harrop 1968; Ulichney 1987) used for most picture reproductions in printing presses to date. It is a photomechanical analog method that produces dots of varying sizes according to the local grayness at the position of the dot.

Before mechanized methods became available, the need to display gray-level pictures on a bilevel medium (mainly ink on paper) was met by artists or craftsmen who specialized in the production of pictures composed of single tone elements on a background of contrasting tone. Engraving, etching, woodcutting and lithography are the most common of the variety of styles and techniques used by these human masters, see Ivins 1985, 1988. (The word *halftone* is itself a translation of the Italian *mezzotint* – the name of one such technique.) Figure 2a–c are examples of the impressive achievements of man-made halftones.

Despite the large variety, there are a few things

The Visual Computer (1994) 10:277–292

1a

b

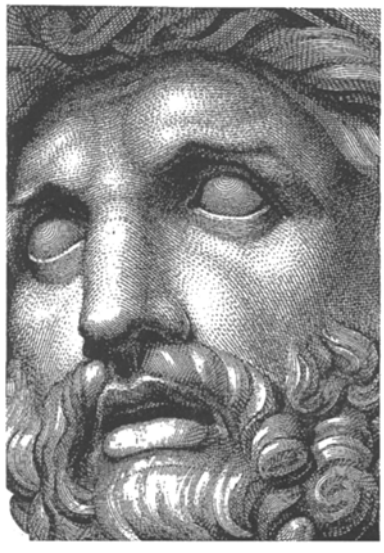
c

d

Fig. 1a–d. Common digital halftone methods: **a** clustered order dither; **b** dispersed order dither; **c** error diffusion – the Floyd-Steinberg method; **d** error diffusion method (Jarvis et al. 1976)

a

b



c

Fig. 2a–c. Examples of man-made halftones: **a** "Erasmus of Rotterdam" by A. Dürer (Strauss 1973); **b** "The Bagpiper" by A. Dürer (Strauss 1973); **c** "A Classical Head" engraved for the Society of Dilettanti, London, 1809 (Ivins 1988)

common to such halftone works of art, especially when compared to the digital methods already mentioned:

- Although one could impose a grid on such works, the picture itself suggests no regular grid of reference.
- The output is pleasing to the human eye even when the elements of the halftone are clearly visible (as often is the case).
- Many such works have an additional artistic value (which would be missing if we were to view, say, a photograph of the scene rendered).
- Like any other product handmade by a skilled person, to have an artist produce a halftone picture is both expensive and time consuming.

Both digital, or mechanized, halftones and human halftones have a place in our world. The first as a way of generating inexpensive and fast outputs and the second as a way of producing highly specialized or artistic outputs and also as a way for skilled individuals to express themselves. However, one could argue that there is also a place for a third type of halftone methods attempting to combine some good features of both.

The proposed **Dig^i*Dürer*** aims at this compromise: we wish to generate, via a software system, halftones that would resemble in style and quality hand-crafted items, yet do so at a fraction of the time and cost required by the human master. Clearly we cannot expect such a system to be as speedy or inexpensive as the regular digital halftone methods and it probably will lack in the higher artistic values associated with the human halftone art. Yet it could fill the gap between the highly specialized and expensive hand-made items and the standard digital halftones, which rely mainly on the small size of the pixel for their quality.

If we keep the system open, the human user can guide the halftone process in matters of style, or make corrections where he deems the system's algorithm has made a wrong choice. Without human intervention **Dig^i*Dürer*** would be a tool for generating higher quality halftones in the style of the human engraver. With human intervention it can become a tool for generating artistic halftones in different styles and varieties according to the user's wish and creativity.

2 The structure of **Dig^i*Dürer***

Much of the beauty of human halftones arises from the fact that the artist understands the subject of his art. To give **Dig^i*Dürer*** the quality and flexibility for which we aim, we must provide means by which such "understanding" of the picture can be encoded into and used by **Dig^i*Dürer***.

However, providing an "understanding" of a picture is extremely difficult and a whole field of research, namely "computer vision", attempts, most of the time rather unsuccessfully, to achieve this goal. Therefore, to give our system the robustness required of a working, usable system, we must not make it too sensitive to the availability and quality of this type of automatic image "understanding".

Our solution to the dilemma is twofold: first, for each type of information required by **Dig^i*Dürer*** we provide three possible sources – a built-in algorithm, an interface with the user, and a "side information" mode that allows **Dig^i*Dürer*** to use information provided externally, either off-line by a human agent or by "your favorite state-of-the-art algorithm" not implemented in **Dig^i*Dürer***.

Second, we structure **Dig^i*Dürer*** into levels according to the complexity of obtaining the information required. Each set of lower levels is designed to function as an autonomous halftone system. When "deeper understanding" is available, the appropriate higher levels become active and guide the lower levels of the halftone process. We identify four levels for **Dig^i*Dürer***:

1. The *basic level* has no understanding of the picture; it simply halftones (engraves) it "blindly" according to the rule of keeping the local density of black elements proportional to the local grayness of the picture.
2. The *syntactic level* understands the picture up to the level achieved without any knowledge of real world objects. This includes segmentation into planar regions according to gray-level similarity, edge detection, etc. (to moderate degree of accuracy, this information is considered obtainable by existing known algorithms). The syntactic level uses its knowledge to decompose the picture into regions, which are then separately halftoned by the basic level algorithm.
3. The *semantic level* understands the picture as a two-dimensional projection of a real-world scene. This includes the 3D structure of the scene, its segmentation into real world objects,

etc. The effects of this level are in guiding the halftone process to enhance the 3D structure and to correctly deal with shadows, reflections, occlusions, etc.

4. The *artistic level* is supposed to have the type of understanding that transforms a "mere picture" into a "work of art". Unlike the other levels of **Dig^{i}*Dürer***, we make no attempt at even suggesting how (if at all) such a transformation can be done algorithmically. We do, however, see it as part of **Dig^{i}*Dürer*** for the following reasons:
   1. It indicates what we lack with respect to human artists.
   2. It allows for the possibility that an experienced and artistically inclined person might be able to use **Dig^{i}*Dürer*** to produce works of art, *his* creativity and understanding replacing what cannot be supplied by the program.
   3. The belief that for some purposes this level might be replaced by a *pseudo-artistic* level; that is, an algorithm designed to mimic the style and technique of a specific human artist or artistic movement. Attempts to produce *pseudo-artistic* pictures have been made (Haeberly 1990) and they seem feasible. Keeping in mind that our goal is items that are midway between digital halftones and human art, a pseudoartistic level might be sufficient.

Remark: the aspect of *textures* should be incorporated into this scheme, although it is not completely clear to the authors how. One possibility is to have a separate *texture level* that extracts textures from the picture and guides the rendering process accordingly. Another possibility is to add it to the existing syntactic and semantic levels (since texture too, in our categorization, is either "syntactic" or "semantic").

The prototype **Dig^{i}*Dürer*** as presented in the sequel describes the current stage of development. It implements a basic level and a partial syntactic level. For the most part, we concentrate on halftones composed of line elements. The need to concentrate on a single type of graphic element was necessary to focus the development effort. We chose line elements since they dominate in human works of art. However, later we show that this was a beneficial choice in the sense that we can generalize from lines to arbitrary elements in a relatively simple manner. The presentation is as follows:

- Section 3 describes the basic level algorithm and shows its results.
- Section 4 presents improvements on this algorithm. The improved version is the *engine* used in **Dig^{i}*Dürer***.
- Section 5 explains how to extend the presented *halftone with lines* method of **Dig^{i}*Dürer*** into a halftone method with almost arbitrary shapes.
- Section 6 explains how the syntactic and semantic levels intervene in the halftone process to improve output quality. We give some examples of syntactic level outputs.

3 The raw basic level

3.1 The Eikonal equation

The basic level has no understanding of the picture. Its operation is completely local: for any given small region of the picture – A the amount of gray in the output should equal the total gray level of the input, i.e.,

$$\int_A h(x, y)\, dA = \int_A g(x, y)\, dA \tag{1}$$

where $g(x, y)$ is the original picture normalized so that 0 is white and 1 is black, and $h(x, y)$ is the halftone picture, which, at any point, is either white or black. (It is common in some cases to replace $g(x, y)$ with $f(g(x, y))$ where f is a nonlinear point operation representing the response of the human eye to the different gray levels.)

We wish to use line elements: let L be a (curved) line of the rendering, and let (s, t) be a coordinate system such that for each point on L the direction s is a tangent to L and the direction t is normal to it. Equation 1 can now be replaced by

$$\int_{s\in A}\int_{t\in A} h(s, t)\, dt\, ds = \int_{s\in A}\int_{t\in A} g(s, t)\, dt\, ds$$

Since $\frac{\partial h}{\partial s} = 0$ (the line is uniformly black and the background uniformly white). This implies that

$$\int_{t\in A} h(s, t)\, dt = \int_{t\in A} g(s, t)\, dt \tag{2}$$

In other words: the local density of halftone lines in the direction normal to the lines themselves must be proportional to the local gray level.

Looking at classic halftones, we observe the following properties:

1. The lines are nonintersecting.
2. They are approximately parallel to each other.
3. They are smooth (except at the edges).

These three properties, which seem to play an important role in making the halftone pleasing to the human eye, are characteristic of "equipotential" lines of potential fields. What we need then, is a potential function $H(x, y)$ whose equipotential lines satisfy Eq. 2. Such a function $H(x, y)$ must be a solution of the *Eikonal Equation:*

$$\|\nabla H(x, y)\| = \sqrt{\frac{\partial H^2}{\partial y} + \frac{\partial H^2}{\partial x}} = g(x, y)$$

Using this equation to produce, with line elements, an illusion of a picture was first suggested by Schroeder (1974). The motivation there was not to generate halftones, but to give an example of yet another "gestalt" effect.
The general algorithm for the *basic-level* halftone generation could be:

1. Choose a line $L(s, 0) = (x(s, 0), y(s, 0))$ as the initial condition.
2. Find an H which solves the Eikonal equation for $g(x, y)$ with boundary condition $L(s, 0)$.
3. Produce equipotential contours of H as the desired halftone (sampled at a distance ΔH apart, where ΔH is proportional to w – the width of the displayed lines).

The following remarks about this general algorithm should be made:

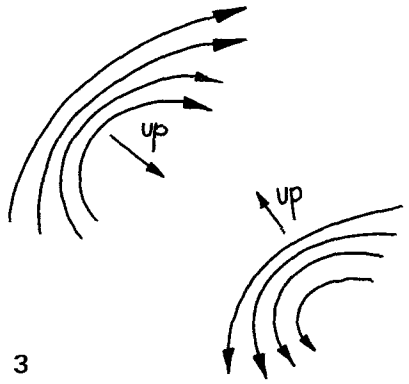
- Without further knowledge of the picture content, there is no preference for choosing one type of initial condition over another, and indeed for the basic-level algorithm any smooth line would do. Part of the control higher levels have on the basic level is in choosing appropriate initial conditions.
- The exact way in which we find H and extract its equipotential lines is immaterial to **Digi**$\mathcal{D}$*ürer*. The only considerations are therefore technical: accuracy, simplicity, and time complexity of the algorithm. We present two acceptable methods in the following subsections.

3.2 An Equipotential contour (EPC) algorithm

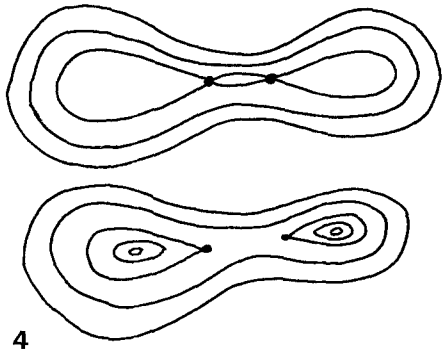
Since our final aim is not H, the potential function, but its equipotential lines, it makes sense to use a method that produces them directly. Such methods are called *front propagation methods.* This subject, being important in shape skeletonization, distance transforms, shape offsetting, and shape analysis has been extensively studied (Blum 1973; Borgefors 1986; Danielsson 1980; Kimmel and Bruckstein 1993; Sapiro et al. 1993). Since here it is only a tool for other purposes, we refer the reader to these sources for more details. An example of such a method for the solution of the shape from shading problem is given by Bruckstein (1988). Since we can view our Eikonal equation as a special case of the *image irradiance equation* solved there, we can apply this method directly for our purposes.
Given that $L(s, 0) = [x(s, 0), y(s, 0)]$ is a known equipotential line, we move a distance dt from each point on it in the direction of the normal, satisfying

$$dH = g(x, y)\, dt.$$

The points reached constitute the equipotential line $L(s, dt)$ for which the potential is greater (or less) than at $L(s, 0)$ by dH. Apart from a few numerical and topological problems, which are listed below, the iterative application of this step can give a mapping of the picture with equipotential contours $(L(s, t))$.
One problem is that the algorithm always goes in one direction (say, "up" the potential field) so that we might not cover the entire picture. If this happens, we can reinvoke the algorithm after we have reversed the direction of the parameter s on $L(s, 0)$, an operation which flips the directions of "up" and "down" (Fig. 3).
The other problem is of a topological type: even for a smooth picture $g(x, y)$ and a smooth line $L(s, 0)$, there might be cases where a step of the equipotential contour algorithm of a line $L(s, t)$ (even with a very small ΔH) produces a new line $L(s, t + \Delta t)$ that intersects itself. Since equipotential lines do not intersect themselves, we cannot proceed from $L(s, t)$ by a simple application of the iterative step. Such a situation occurs as we approach either *extrema* or *saddle points* of $H(x, y)$.
The case of extrema is relatively simple both to detect and to resolve: when we find that l, the arc length of an equiheight contour, is below a critical



3



4

Fig. 3. Changing the signs of "up" and "down"

Fig. 4. Handling saddle points

length, we assume that we have reached an extremum and stop the propagation.

The case of saddle points is more difficult: one of the indications of a saddle point, or points, is that the line $L(s, t)$ crosses itself. This requires a special algorithm to check for self-intersections of the line. After the existence of a saddle is detected, we must divide our line into three separate segments: one including all the points on one "side" of the saddle, another for all the points on the other side of it, and a third for all the points that make up the saddle (which could be a whole line and not just a single point). Once we have made this division, we can ignore the points of the saddle. For the other two groups of points, we must continue the propagation, but now separately for each of the two (now separated) equiheight contours formed (Fig. 4).

The numerical implementation of this algorithm is as follows:

1. Start with some equipotential line $L(s, 0)$.
2. If the current line is in the picture and is not degenerated, do:
   a. Normalize the line so that s is the arc length – this is necessary to ensure better numerical stability (Sethian 1985).
   b. Check the line for self-intersections: if an extremum was reached, discontinue step 2. If a saddle was reached, remove the points of the saddle itself, and separate the remaining points into two equipotential lines. Put one on the stack and make the other the current line. Renormalize if necessary.
   c. Output the line.
   d. From each sampled point on the current line calculate the direction of the normal and move a distance of ΔH in that direction. (This amounts to integrating $g(x, y)$ in the direction of the normal until the integral equals ΔH.) The points reached define the next equipotential line.
   e. Make the set of points reached the current line.
3. If the stack is not empty, read a line from it an go to step 2.
4. If we have not covered the entire picture, reverse the direction of $L(s, 0)$ and go to step 2.

3.3 Integrate to threshold

It is interesting to point out that we can arrive at this EPC algorithm by taking a totally different approach and viewing the basic level as a limiting case of the standard halftone method of error diffusion first introduced by Floyd and Steinberg (1976).

For the one-dimensional case the halftone problem can be stated as follows: Given a sampled signal $0 \leqq g(n) \leqq 1$ (n integer), produce a sequence $h(n) \in \{0, 1\}$ such that the error $E(n) = \sum_0^n g(n) - h(n)$ is minimal for every n. A solution can be produced by the *error diffusion* algorithm: given that for some value i, $h(i) = 1$, the next pixel with $h(j) = 1$ is the first pixel $j > i$ such that

$$E(i) + g(j) + \sum_{k=i+1}^{j-1} (g(k) - h(k)) = E(i) + \sum_{k=i+1}^{j} g(k) \geqq 1$$

Suppose that w is the size of the smallest black mark we can produce. In the limit, for constant

w and increasingly smaller pixels the algorithm becomes the so-called *integrate to threshold* algorithm: given that for some value x_0, $h(x_0)$ is a point of transition from 0 to 1, the next point of transition from 0 to 1, x_1, is such that

$$E(x_0)+\int_{x_0}^{x_1}(g(t)-h(t))\,dt=E(x_0)+\int_{x_0}^{x_1}g(t)\,dt-w\geqq 1$$

Suppose we wish to halftone a two-dimensional picture $g(x, y)$ with line elements of width w, and we are given some arbitrary line $L(s, 0)$ that should be part of the halftone. The next line $L(s, 1)$ should be such that

$$E(t)=\int_{s_1}^{s_2}\int_{L(s,0)}^{L(s,1)}(g(x, y)-h(x, y))\,dA$$

is minimal for any s_1 and s_2. If $g(x, y)$ is smooth and the curvature of $L(s, 0)$ is not too large, an extension of the 1D integrate-to-threshold algorithm is a possibility: treat the line $L(s, 0)$ as an infinite set of 1D elements. For each such element, s, choose a direction (which changes smoothly with s) and apply the integrate-to-threshold method. The best choice of direction is the normal to the line at s, as it ensures that we satisfy step 2, Sect. 3.2 and guarantees minimal overlap of the paths of integration. It also ensures the local similarity of $L(s, 1)$ to $L(s, 0)$. The resulting algorithm is the EPC algorithm.

3.4 An alternative – the level sets algorithm

A different method for obtaining the equipotential lines of a potential function satisfying a differential equation such as our Eikonal equation is presented in Osher and Sethian (1988). In this approach we induce a smooth function $\Phi(x, y, 0)$ on the picture with the following properties:

- For all points on $L(s, 0)$ (the intial equipotential line) $\Phi(x, y, 0)=0$.
- For all points with a higher potential $\Phi(x, y, 0)>0$.
- For all points with a lower potential $\Phi(x, y, 0)<0$.

Using this function and its derivatives we iterate on the picture to produce $\Phi(x, y, dH)$, which is a function having the same properties except that $\Phi(x, y, dH)$ is zero at points with a potential higher than that of our initial condition by dH. The algorithm that evolves is as follows:

1. Use $L(s, 0)$ to induce $\Phi(x, y, H_0)$.
2. While $\Phi(x, y, H_i)$ has zeros in the picture, do:
   a. Extract the lines where $\Phi(x, y, H_i)=0$.
   b. Output them.
   c. From $\Phi(x, y, H_i)$ calculate $\Phi(x, y, H_{i+1})$
   $\equiv\Phi(x, y, H_i+\Delta H)=\Phi(x, y, H_{i+1})$
   $=\Phi(x, y, H_i)+\Delta H\times g(x, y)\times\|\nabla\Phi(x, y, H_i)\|$.
3. If the entire picture has not been covered, reverse the signs of $\Phi(x, y, H_0)$ and go to step 2.

The advantages of this algorithm are that it is numerically much more stable and that it takes care of the topological problems automatically (Osher and Sethian 1988). Its big disadvantage is that it is much slower than the equiheight contour (for a picture of size $n\times n$, the complexity of one iteration for the EPC algorithm is $O(n\,Log(n))$ or even $O(n)$ if the number of intersections is much less than n, whereas it is $O(n^2)$ for the level-sets algorithm).

3.5 Results of the basic algorithm

Both the EPC and level-sets algorithms were run on a number of gray-level images with different initial lines. Figure 5a–d are examples of the results produced; the following observations were made:

1. Overall, the algorithm is correct – when the picture is viewed at an appropriate distance (which depends on w, the width of the line) the halftone closely resembles the original.
2. For most regions of the picture, the rendering has the pleasing qualities associated with classical human halftones (consecutive lines are roughly parallel and nonintersecting and individual lines are mostly smooth).
3. The quality of the output is not very sensitive to different initial conditions (although no two lines match in Figs. 5a–c, the impression we get from the pictures is much the same).
4. In some regions, it happens that, at certain points, breaks in the smoothness of the lines appear. Since neighboring lines are similar, correlated breaks in smoothness occur in consecutive lines with the unfortunate effect that our eyes see false contours in the picture.

The first two observations point out that, globally, the reasoning behind the development and imple-

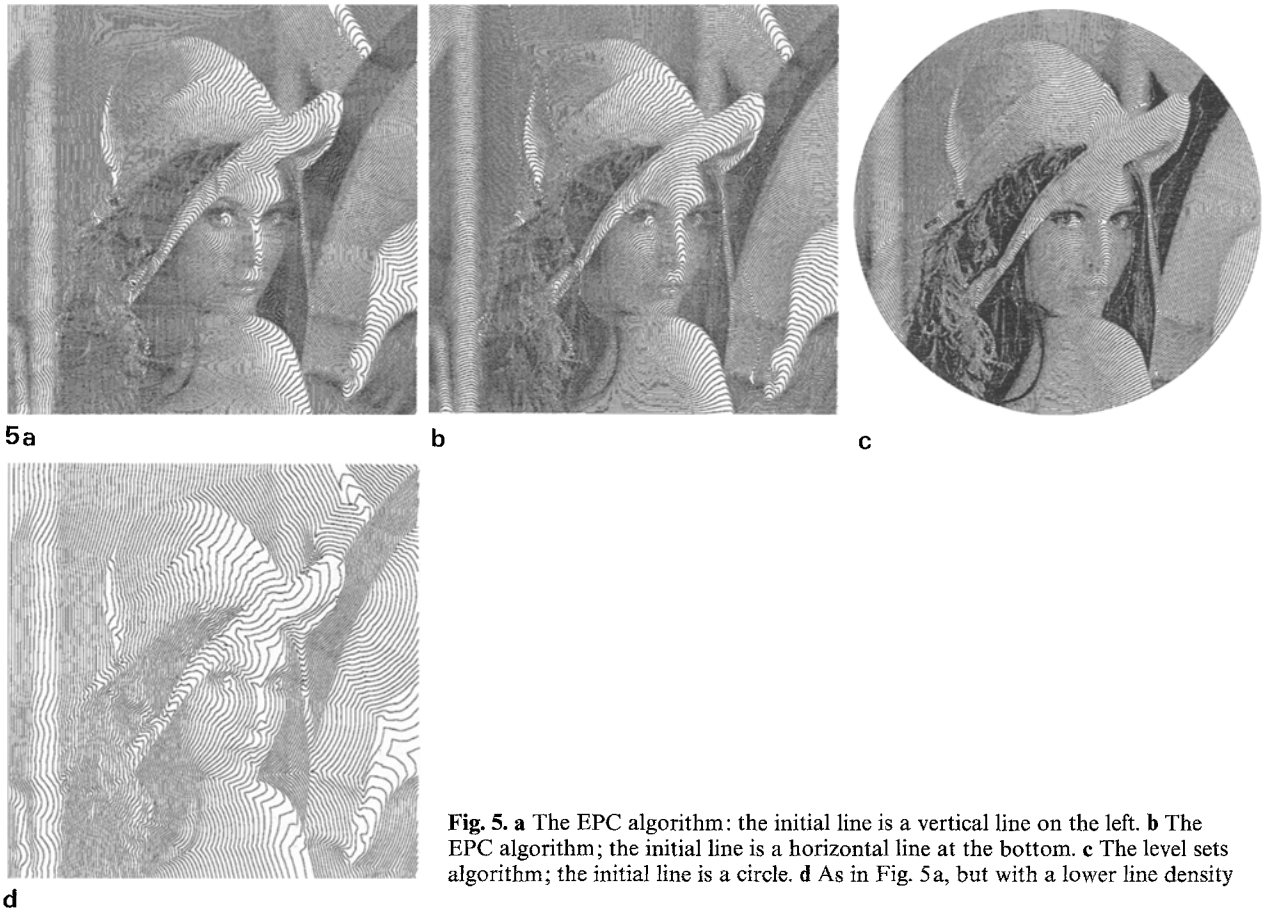


Fig. 5. a The EPC algorithm: the initial line is a vertical line on the left. **b** The EPC algorithm; the initial line is a horizontal line at the bottom. **c** The level sets algorithm; the initial line is a circle. **d** As in Fig. 5a, but with a lower line density

mentation of the method were correct. The third shows that at least at the basic level the initial line can be chosen arbitrarily. The fourth observation requires closer investigation as it is the most serious flaw in the quality of the output and also as it reveals a flaw in the design of the algorithm, since our aim was to produce lines that are, among other things, "smooth".

We term the points at which a "break in the smoothness" is formed *discontinuities* or *shock fronts*, since this effect is caused by a discontinuity in the normal vector to the line $(-y'(s), x'(s))$. The fact that such discontinuities are annoying is no surprise, given the known properties of the human visual system (Levine 1985). On one hand, the human visual system is tuned to change, so that even a small discontinuity in a long, otherwise smooth line immediately draws our attention, and on the other hand, at a higher level the visual system tends to group small *features* into a larger *object*. The first property focuses our attention on the discontinuities and the second causes us to interpret them as (false) contours. The question we have to answer is how such discontinuities are formed. Given that, we must either find a way to prevent them from being formed, or if this turns out to be impossible, we must find a way to make them less obvious.

4 The improved basic level

4.1 What causes the shocks?

At first one might suspect that the *shocks* are caused by some technicality of our implementation (a bug in the software); however, this is not so. The problem survived different implementations of the EPC algorithm, as well as the level-sets algorithm that uses a different computational ap-

proach. To find its source, we look at the mathematical analysis of the Eikonal equation.

We rewrite the Eikonal equation as an evolution equation for $L(s, t)$ – the map of the equipotential lines. We view $L(s, t)$ as a vector function of two parameters: $L(s, t)=[x(s, t), y(s, t)]$ where t is the potential parameter of the line and for $t=constant$ $L(s, t)$ is a single equipotential line. Since we climb at a constant rate, we can interpret t as a time parameter and speak of the position of the front, $L(s, t)$, at time t. Our algorithm performs the following:

$$L(s, t+dt)-L(s, t)=\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\cdot\frac{\frac{\partial L}{\partial s}}{\left\|\frac{\partial L}{\partial s}\right\|}\int_t^{t+dt} g(x, y)\,dt$$

For a small dt, a smooth $g(x, y)$ implies that

$$\int_t^{t+dt} g(x, y)\,dt \approx g(x, y)\,dt$$

and so we get that

$$\frac{\partial L}{\partial t}=\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\cdot\frac{\frac{\partial L}{\partial s}}{\left\|\frac{\partial L}{\partial s}\right\|}\,g(x, y)$$

In this form we see that our equation is hyperbolic (or a *reaction equation*).

Given that $L(s, 0)$, the initial condition, and $g(x, y)$ are smooth, it can be shown that for t close to 0, $L(s, t)$ is also smooth (this is what gives the lines their smoothness for most of the picture). However, it is well known that as t moves away from 0, shocks (i.e., lines along which $L(s, t)$ is not smooth) may appear. This happens when we reach the same point on the plane, by following the characteristics of the equation, from more than one point on $L(s, 0)$. It can be shown that the location at which a shock is formed depends on the curvature of $L(s, 0)$ and on the rapidity of change of $g(x, y)$ in the *domain of dependence* of that point. For a low curvature $L(s, 0)$ and an almost constant $g(x, y)$, such a location may be quite far away from $L(s, 0)$, whereas for a high curvature $L(s, 0)$ or rapidly changing $g(x, y)$ we encounter a shock almost infinitesimally close to $L(s, 0)$.

To ensure smooth lines we can try either to replace the Eikonal equation with another equation that is less sensitive to rapidly changing $g(x, y)$ and high curvature $L(s, 0)$ but still has good halftoning properties, or we can try to force $g(x, y)$ to change slowly.

The first approach seems the more promising as we cannot expect to choose the properties of the picture we halftone. Unfortunately, as discussed in the following subsection, this approach gives only a partial solution. Happily, it turns out that there are some operations that have the effect of limiting the rate of change of $g(x, y)$ while still keeping the output true to the original picture. Some such operations fall within the jurisdiction of the *basic level* and are discussed in a succeeding subsection; others are explained when we discuss the higher levels of **Dig^{i}***Dürer*.

4.2 The reaction diffusion alternative

It is well known that the phenomena of shocks never occurs in evolution equations of a *parabolic form* (also called *diffusion* or *heat equation* form):

$$\frac{\partial L}{\partial t}=\alpha\frac{\partial^2 L}{\partial s^2}=\alpha k\vec{n}$$

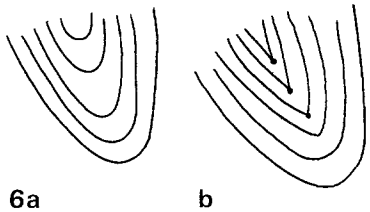
where k is the signed curvature of the plane curve $L(s, t)$ with t constant and s being the arc length parameter.

Intuitively, we may say that since propagation speed depends on the curvature, the front always "leaves the room before the walls close in" (Fig. 6).

We can, therefore, attempt to combine our Eikonal equation with a diffusion equation in the hope of getting the good properties of both: such equations are known as *reaction-diffusion equations* (Smoller 1983). Our particular reaction-diffusion equation takes the following form:

$$\frac{\partial L}{\partial t}=\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\cdot\frac{\frac{\partial L}{\partial s}}{\left\|\frac{\partial L}{\partial s}\right\|}\,g(x, y)+\alpha\frac{\partial^2 L}{\partial s^2} \qquad (3)$$

It can be viewed as a linear combination of the reaction effect produced by the Eikonal equation and a diffusion effect that serves to smooth out the shocks. The relative *strength* or *speed* of each is determined by the parameter α. As expected, the behavior of such an equation is somewhere between that of the reaction term alone and that of the diffusion term alone: as we use consecutively




7


8

Fig. 6a, b. The evolution of: **a** the diffusion term; **b** the reaction term

Fig. 7. Adding a nonselective diffusion term

Fig. 8. Adding a selective diffusion term

larger values for α, the solution remains smooth for more rapid change in $g(x, y)$ or higher curvature of $L(s, 0)$. However, we must ensure that in this transition we do not loose the desirable properties of the Eikonal equation – correct rendering of the gray tones and sharpness of output.

Correct rendering

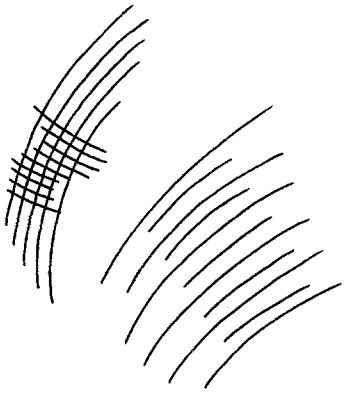
We arrived at the Eikonal equation in an attempt to satisfy Eq. 1; however, already in moving from Eq. 1 to Eq. 2 we had to make an assumption that either L is of very low curvature everywhere or that we do not stray too far from it. It can be shown that up to some value of α ($\alpha \leqq c \Delta t^2/2$, where c is the local grayness of the picture and Δt is the local step size between two consecutive lines) the reaction-diffusion Eq. 3 satisfies Eq. 1 even better than the Eikonal equation. However, we cannot go much above this value before we loose the halftoning property of the equation.

Sharpness of output

Loss of sharpness is the very nature of a diffusion process and the effect becomes more noticeable as the "strength" of the diffusion is increased relative to the reaction term. However, the effect is also related to the rate of change of $g(x, y)$. For regions of the picture that are constant or slowly changing, diffusion has little or no undesired effect. For rapidly changing $g(x, y)$ it is highly objectionable. To keep objectionable desharpening at a minimum we apply the diffusion adaptively. α, the strength of the diffusion, is inversely related to the local rate of change of $g(x, y)$ and ranges from 0 (no diffusion) at the edges (places of a high rate of change) to a maximum when the picture is constant locally.

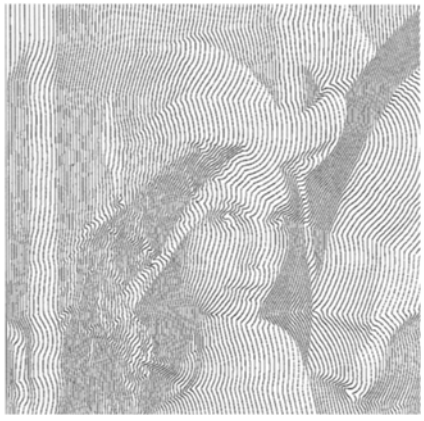
Happily those regions of the picture for which we cannot apply diffusion (or apply it very weakly) are precisely those regions where the discontinuities in the output lines are the least objectionable. A rapidly changing $g(x, y)$ manifests itself as an edge to the human visual system, since the edge is a "break in smoothness" in the picture. The fact that our lines might display a break in smoothness at an edge location is not objectionable and might even help enhance the edge. (In other words a "false contour" that happens to coincide with a "real contour" is much less objectionable than one which does not.)

Figures 7 and 8 display the effects of a diffusion term: we can see that constant diffusion (Fig. 7) results in a severe loss of sharpness, whereas the effect is only slightly noticeable for adaptive diffusion (Fig. 8). Comparing these with Fig. 5a, we see that most of the shocks have disappeared. However the most severe shocks – the ones which are most



9

10a



b

Fig. 9. Using multiple sets of lines

Fig. 10a, b. Outputs with dashed lines

noticeable – are precisely those that, although softened, did not disappear. To smooth them out would require a value of α that causes a severe loss in sharpness and rendering quality.

In summary, we can improve the quality of the output by using a reaction-diffusion version of the Eikonal equation. However, there is a limit to the strength of the diffusion (the value of α) we can apply. Hence the method is effective only up to a certain rate of change in $g(x, y)$ beyond which it cannot overcome the shocks inherent in the reaction term.

4.3 Breaking the lines

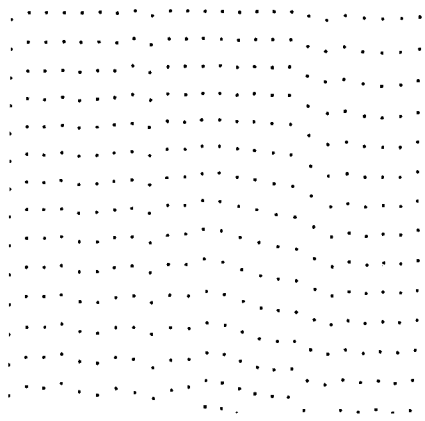
Going back to handmade halftones, we see that the human artist also had to deal with the problem of $g(x, y)$ changing too fast to make a good rendition with line elements. His solution, many times, was to change the type of lines used. For bright regions it is not uncommon to see the artist switch from continuous lines to dashed lines, where the proportion between the length of the black and white segments decreases as the rendered tone becomes lighter. For dark regions the artist may either switch to wider lines or else use two overlaying patterns of lines at the same or different orientations (see Fig. 9 and of course Figs. 2a–c).

Similar solutions may be adopted to our case. Suppose we set a minimum $d_{\min}$ and a maximum $d_{\max}$ on the distance between a point on $L(s, t)$ and its corresponding point on the consecutive line $L(s, t+\Delta t)$. In effect this limits the range of $g(x, y)$ from $0 \leqq g(x, y) \leqq 1$ to $w/d_{\max} \leqq g(x, y) \leqq w/d_{\min}$ (and hence limits the maximum rate of change over a constant length Δt. Since we want a correct rendering, when we develop a line in regions that are darker or brighter, we must remember the amount of black or white we "owe" each point. Later, when we output the lines, we pay this "due" back by making the lines wider or dashed depending on whether we "owe" black or white. Thus, it is possible that, by appropriately selecting the values of $d_{\min}$ and $d_{\max}$, we can get an output of smooth lines, without sacrificing the other desirable properties of our original algorithm. (In the limit, when setting $d_{\min} = d_{\max}$ the output will be parallel lines with the halftone effect achieved only by locally varying the width or dashedness of the lines.) An example of the application of this method can be seen in Fig. 10a, b (where we limited only $d_{\min}$ and paid back the "due" by making the lines dashed). It can be seen that the shocks have all but disappeared.

5 From lines to general elements

An interesting feature of our method is that it forms the basis for a general halftoning algorithm with any type of elements – even halftoning with different types of elements in the same picture.

Consider the output of the basic level algorithm



11

12

Fig. 11. The phase problem

Fig. 12. Generalizing to dot elements

when run for extremely thin lines. Except at some special points (such as saddle points and extrema of the potential function) this output can be thought of as an approximation of a function $L(s, t)$: $R^2 \to R^2$. A walk along a $t = constant$ line is a walk along one line of our output. A walk in the t direction, on the other hand, is a walk in *blackness space*, that is, $t_4 - t_3 = t_2 - t_1$ if and only if

$$\int_{t_3}^{t_4} g(x, y)\, dt = \int_{t_1}^{t_2} g(x, y)\, dt$$

Suppose we recalibrate the s direction so that it also has this property, that is, $s_4 - s_3 = s_2 - s_1$ if and only if

$$\int_{s_3}^{s_4} g(x, y)\, ds = \int_{s_1}^{s_2} g(x, y)\, ds$$

(Note that for the purpose of numerical stability s was an arc length parameter – but once we have computed the lines we can recalibrate s any way we choose.) We can now view s and t as a new coordinate system that has the useful property that regions of equal area in the s, t plane cover equal amounts of blackness in the picture. This has the following implications:

1. In the same way that lines of constant t can be used as halftones of the picture, so could lines of constant s be used to form an orthogonal halftone (each line of the $t = const$ halftone intersects the lines of $s = const$ halftone at right angles). Moreover, displaying both types of lines together would result in a crisscross pattern that is quite common in man-made halftones.
2. Suppose we wish to halftone a picture, not with lines, but rather with area elements of a certain shape and size A. Since the amount of blackness in one such shape is simply the area of the shape, a halftone results when we place these elements on a proper grid in the s, t space: we know that the resulting locations of the elements in the x, y satisfies Eq. 1 and hence is a halftone of the picture with these elements.
3. This observation remains true if instead of one type of such elements we tile the s, t plane with a set of possible elements. We get a halftone composed of elements from this set.

To turn the ideas of this section into a robust algorithm we must still deal with two types of problems. One is the problem of the saddles and extrema and the other is the problem of a phase correlation. When the elements are placed on the s, t grid in the same phase, the correlation between the elements may result in unpleasant Moiré patterns although the output is a halftone (Fig. 11). However, even without addressing these problems, the results can be quite satisfactory as the test example in Fig. 12 shows. This example was produced by running the EPC algorithm to produce lines $L(s, t)$ and then running a one-dimensional version of it on each line to recalibrate s. The dot elements were placed on the resulting (s, t) grid.

6 The higher levels of **Dig$^{i}_{D}$ürer**

6.1 The role of the higher levels

The *improved basic level* as already presented can be viewed as the engine of **Dig$^{i}_{D}$ürer**. By properly setting up its different parameters we can produce all (or at least most) of the elements or patterns we would like to see in a halftone. Moreover, when viewing a small patch of the picture the output is usually, indeed, what we would like to see.
The *higher levels* of **Dig$^{i}_{D}$ürer** should be viewed as the driver (actually, automatic driver) that sits behind the wheel and controls the engine so that **Dig$^{i}_{D}$ürer** goes where we want it to. Like the driver, it exerts its influence by properly setting up the parameters under which the engine runs. These parameters include the following:

- $g'(x, y)$ – The picture that the basic level sees and halftones (as opposed to $g(x, y)$, the raw input to **Dig$^{i}_{D}$ürer**).
- $L(s, 0)$ – The initial condition (first line) of the halftone.
- The rendering parameters including:
  - w, the width of the lines produced.
  - α, the strength of the diffusion term.
  - $d_{\min}$ and $d_{\max}$, the maximum and minimum distance between two consecutive lines.
  - The graphical elements to be used in the halftone (if we include the ideas of the previous section and allow for general elements).

Note that the rendering parameters can be set either statically (once for each run of the basic level), adaptively (different values at different locations in the picture), or interactively (recomputing them after each step or iteration of the basic level).
When designing the higher levels of **Dig$^{i}_{D}$ürer** we must answer the following two intertwined questions: What information do we need to properly set the controls of the basic level? Given that information, how do we use it to give each parameter its optimal value? To answer these questions we compare our outputs to the works of artists and to the ideas we have as to how we would like our outputs to appear. Then we try to find those aspects with which we are least satisfied and improve them.
This leads to an iterative development process that eventually produces a set of heuristics comprising a level of **Dig$^{i}_{D}$ürer**. Since some of these heuristics are useful as separate algorithms, we speak of them as *partial levels*. One important partial level which deals with the decomposition of the picture into objects is presented in the following subsection. Ideas about other aspects of the higher levels are discussed in the concluding subsection.

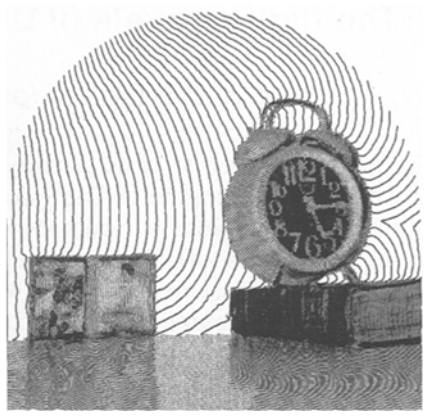
6.2 Separation into objects – a partial syntactic level

One of the most obvious differences between human halftones and the results of our basic level is that the human artist almost always renders each *object* in the picture separately, whereas our basic level renders a picture as a whole. Among other things, by doing so, the artist generally avoids the problem of rapidly changing gray levels that produce the shocks encountered at our basic level.
For most pictures and most objects the local gray level does not change within an object as rapidly as it does across objects (a change which is interpreted as an "edge"). This is especially true for syntactic segmentation where the criteria for segmentation are local similarity in gray levels and the paradigm that an object cannot cross an edge. Indeed, examining outputs of the basic-level algorithm, we observe that all the severe discontinuities (those that do not disappear in the reaction diffusion version) and quite a few of the less severe ones, have their origin on an edge between two distinct objects or an object and its background.
(An instructive way of stating this is that for an input picture containing edges – unsmooth or rapidly changing gray levels – it might be that there is no smooth solution to the Eikonal equation or its reaction-diffusion counterpart. The way to generate a good halftone is then by placing the discontinuities where they are least objectionable or even beneficial, i.e., on the edges between objects in the picture. Hence we modify our method by artificially forcing discontinuities at the edge locations.)
Once a segmentation of the input picture is available, we can perform a higher-level algorithm (which is syntactic or semantic – depending on the type of information at hand) as follows:

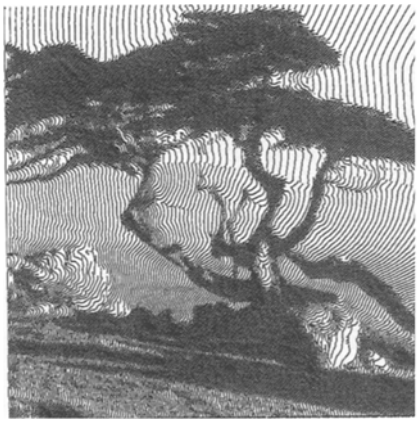
1. For each object of the input picture (including the background as a separate object):
   a. Create a new picture $g'(x, y)$ that consists of the current object and an interpolation of it into its convex hull.

13

14



15



16

17

Fig. 13. An example of syntactic level output of **Dig**$^{i}_{D}$*ürer*

Fig. 14. Simulation of a wood cut (syntactic level output of **Dig**$^{i}_{D}$*ürer*)

Fig. 15. An example of syntactic level output of **Dig**$^{i}_{D}$*ürer*

Fig. 16. An example of syntactic level output with more than one set of lines

Fig. 17. Syntactic level output with dot elements

b. Choose an initial condition, $L(s, 0)$, for $g'(x, y)$.
c. Choose the parameters of the basic level for $g'(x, y)$.
d. Let the basic level run on $g'(x, y)$.

2. Produce the complete output by clipping and putting together the renderings of the different objects.

Note that the interpolation of each object (which may even be composed of a number of disconnected regions in cases of occlusion) into its convex hull is necessary, since our basic level expects the picture it is rendering to be convex.

To make the algorithm a complete syntactic or semantic level we must specify the information we need and the mechanism(s) we use to choose the parameters in steps 1b and 1c. When such information or such a mechanism is not available, we are forced to use default parameters and we are left with a *partial level* algorithm. This partial level algorithm is conceptually much simpler than a full level algorithm, since the separation of a picture into objects is a task most humans do easily, whereas setting the rendering parameters requires some artistic or graphical expertise (this is also true when the rendering is done by hand and not with a computer).

To check the feasibility of the partial level algorithm we let it run on pictures for which a decomposition into objects was performed semiautomatically. (To separate the issue of good edge detectors and segmenters from the issue of using such information for halftones, the acceptable range of gray levels for each object in the segmentation was chosen by a human agent.) The results can be seen in Figs. 13–17. Indeed, among other improvements in quality, the lack of shocks can be clearly noticed.

6.3 Towards full syntactic and semantic levels

At this stage of our research, we have neither a working syntactic level nor a working semantic level. However, we do have a few ideas, which are being tried, as to how we should go about implementing these levels. We present some of them here, first for the syntactic and then for the semantic levels.

At the syntactic level, a main goal can be to ensure the global similarity of lines within the same syntactical object (which is often associated with a face of a real world object). This can be done by the methods outlined in Sect. 4.3. However, the $d_{\min}$ and $d_{\max}$ should now change interactively to ensure that the "front" does "skip over" local disturbances while it still modifies itself according to the general variations of gray over the entire object.

Once the lines are globally similar, the major remaining issue at the syntactic level is the orientation of the overall group of lines, which can now be controlled by properly selecting the initial line of the rendering. At first glance Figs. 5a–c seem to indicate that this parameter has little effect on the overall quality of the output. However, even there we find cases (for example, the portion of the hat above and slightly to the left of the nose) where the effect is noticeable. When a group of lines is globally similar this effect should become even more dominant.

Possible ways of selecting an initial line at the syntactic level include choosing:

- An edge of the object.
- A line perpendicular to an edge of the object.
- A line along a direction of maximal correlation between pixels in the object.

All three ideas are inspired by observing human halftones. The first two seem to emphasize the object against its background and sometimes enhance its perceived three-dimensional structure. The third is expected to enhance texture.

Observing human halftones, we also notice that not only are the lines similar to one another over large areas, but that typically they also follow the projection of the three-dimensional structure of the object rendered. When we have semantic-level information at our disposal, one of our main aims should be to make our lines follow this 3D structure. Again the method is similar to the previous ones, except that now we do not wish the "front" to move at roughly constant speed, but rather at the predicted speed of the *equiheight* or *steepest descent* "front". Note that unlike the syntactic level, here the problem of making lines in a *crisscross* pattern might actually become simpler since we can use the directions of the normal and tangent of the projection of the three-dimensional structure as our two intersecting directions.

When a choice of the halftone elements is available, it appears that it should be used mainly to enhance texture (especially if a semantic level algorithm already takes care of the three-dimensional structure). This can be done either by selecting halftone elements that match the elements of the texture (dots when the texture is dots, etc.) or else choosing elements that give the feeling of that structure – smooth elements for a smooth surface etc. See examples in Gill (1984).

Acknowledgements. The first author thanks Ron Kimmel from the Electrical Engineering Department at the Technion for introducing him to many aspects of curve evolution, among them the level sets algorithm presented. All gray-level pictures used as input to our algorithm(s) where taken from Weber (1983). The authors wish to thank Dover Publications and John Murray Publishers for permission to use the reproductions of man-made halftones in Fig. 2.

References

Blum H (1973) Biological shape and visual science. J Theor Biol 38:205–287

Borgefors G (1986) Distance transformations in digital images. Computer Vision, Graphics and Image Processing 34:344–371

Bruckstein AM (1988) On shape from shading. Computer Vision, Graphics and Image Processing CVGIP 44:139–154

Danielsson PE (1980) Euclidean distance mapping. Computer Graphics and Image Processing CGIP 14:227–248

Floyd RW, Steinberg L (1976) An adaptive algorithm for spatial grey scale. Proc SID 17:75–77

Gill RW (1984) Rendering with pen and ink. Thames and Hudson, London

Haeberly P (1990) Paint by numbers: abstract image representations. Proc Siggraph 24:207–214

Harrop D (1968) Modern book production, Clive Bingley, London

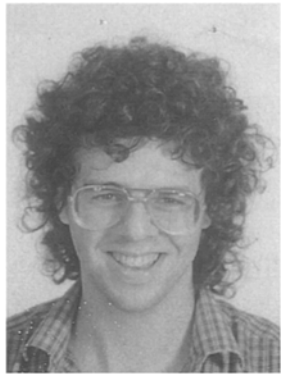
Ivins WM (1985) Prints and visual communication. MIT Press, Cambridge, Mass

Ivins WM (1988) How prints look: an illustrated guide. John Murray, London

Jarvis JF, Judice CN, Ninke WH (1976) A survey of techniques for the display of continuous-tone pictures on bi-level displays. Computer Graphics and Image Processing CGIP 5:13–40

Kimmel R, Bruckstein AM (1993) Shape offsets via level-sets. Comput Aided Design 25(3):154–162

Knuth DE (1987) Digital halftones by dot diffusion. ACM Trans Graph 6:245–273
Kollias S, Anastassiou D (1991) A unified neutral network approach to digital image halftoning. IEEE Trans Signal Processing 39:980–984
Levine MD (1985) Vision in man and machine. McGraw-Hill, New York
Osher SJ, Sethian JA (1988) Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jaccobi formulations.] Comput Phys 79:12–49
Peli E (1991) Multiresolution, error-convergence halftone algorithm. J Opt Soc Am 8:625–636
Sapiro G, Kimmel R, Shaked D, Kimia BB, Bruckstein AM (1993) Implementing continuous-scale morphology via curve evolution. Pattern Recogn (in press)
Schroeder M (1974) The Eikonal equation. The Math Intelligencer C-23: 1264–1269
Sethian JA (1985) Curvature and the evolution of fronts. Commun Math Phys 101:487–499
Smoller J (1983) Shock waves and reaction-diffusion equations. Springer, Berlin Heidelberg New York
Sullivan J, Ray L, Miller R (1991) Design of minimum visual modulation halftone patterns. IEEE Tran Sys Man and Cyber 21:33–38
Strauss WL (1973) The complete engravings etchings & drypoints of Albrecht Dürer. Dover Publications, New York
Ulichney RA (1987) Digital halftoning. MIT Press, Cambridge, Mass
Weber A (1983) Image data base. USCIPI Report no 1070. Image Proc Inst University of Southern California



YACHIN PNUELI is currently on a Post Doctoral position in Berlin, after having received his PhD in Computer Science at the Technion = Israeli Institute of Technology. There he received his BCc in 1985 and MSc in 1987. He has also worked for *i*-Logix Inc. His current research interests include digital halftone methods, computer vision and abstract model theory.

ALFRED M. BRUCKSTEIN was born in Transylvania, Romania, on January 24, 1954. He received the degrees of B.Sc. and M.Sc. in electrical engineering, from the Technion, Israel Institute of Technology, in 1977 and 1980, respectively, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, California, in 1984. From October 1984, he has been with the Technion, Haifa, He is a frequent visitor at AT & T's Bell Laboratories, Murray Hill, New Jersey. His present research interests are in computer vision, pattern recognition, image processing, and computer graphics. He has also done work in estimation theory, signal processing, algorithmic aspects of inverse scattering, point processes and mathematical models in neurophysiology. Prof. Bruckstein is a member of SIAM, MAA and AMS.