INVERSE HALFTONING OF SCANNED COLOUR IMAGES

Master's thesis by

JÖRGEN RYDENIUS

Image Processing Laboratory
Department of Electrical Engineering
Linköping University and Institute of Technology

Reg. no. LiTH-ISY-EX-1713

Examiner: Björn Kruse

Linköping, January 1997

ABSTRACT

This thesis is concerned with inverse halftoning of scanned colour halftone images. Different aspects of the problem are treated thoroughly and the difficulties are pointed out. Halftones, and especially colour halftones, are analysed in the Fourier domain.

A method to perform inverse halftoning of colour halftone images is proposed and implemented in Matlab. The resulting image quality is good and the method may very well be fully automized in the future.

The thesis gives an introduction to the concept of inverse halftoning and could serve as a base for future development of educational material on the topic.

TABLE OF CONTENTS

Chapter

1. INTRODUCTION

Photographs, and many computer generated images, are of continuous tone; the colour tone or gray level varies in a continuous manner. Most of the current printing devices are not capable of directly reproducing this continuous tone. They can only produce binary outputs. Therefore the process of halftoning—also known as screening—is used. The image is thereby converted into one or several binary images, usually consisting of small dots, which then can be printed. The process of halftoning is information lossy and highly non-linear. Because of this, there is no simple way of inverting the process. This work concerns how to make good reconstructions of scanned colour halftone images, back to continuous tone images.

No official reports have yet been published on inverse halftoning of printed *colour* images. Besides the problem of monochrome inverse halftoning, this may add the task of *reseparating* the printing process colours. Also, when working with images that has been printed, a phenomenon called *dot gain* is introduced. In this field the Image Processing Laboratory (IPL) at Linköping University has done research earlier, and these concepts may now be reused.

The target input image is a printed colour halftone image from a newspaper or similar. The work has been done with the issue to handle *real* printed halftone images, which are very noisy, and *not* only halftone test images that never have left the computer.

1.1 BACKGROUND

The past few years much research has been made in order to perform inverse halftoning—or descreening—of monochrome images. Different ways has been proposed (see chapter 3, “Methods for Inverse Halftoning”), but perhaps the most successful work has been done by Søren Forchhammer et al. at Denmark Technical University. Their knowledge has been shared with Eskofot DGS, who has developed a commercial system for inverse halftoning, called EskoDDS.

1.2 METHOD OF WORK

The first part of the project was devoted to literature studies on inverse halftoning. I decided that the model in figure 1.1 would be a good approach to solve the problem. It later proved that the reseparation process was hardest to handle, and I later modified the model to perform inverse halftoning directly on the scanned RGB image. Different methods for inverse halftoning were constantly applied to printed test images, and evaluated.

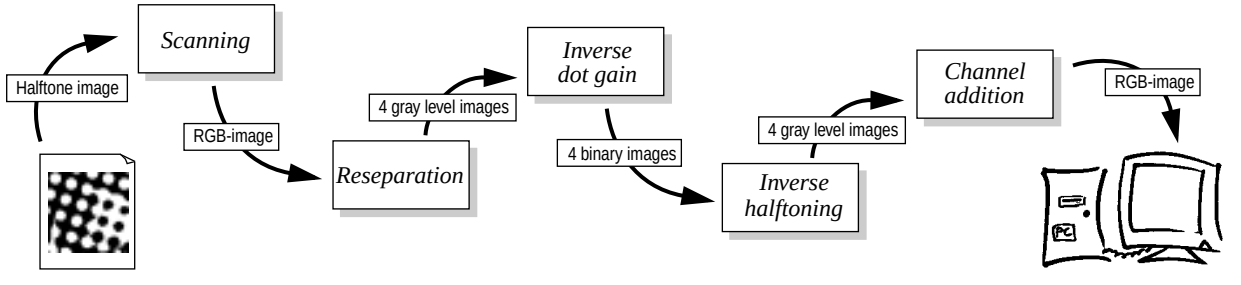


Figure 1.1 *Proposed model for inverse halftoning of scanned colour images.*

The main issue during this project has been to get the subjectively best possible output image — rather than, for example, the best possible signal to noise ratio, compared to the original image. It is important to remember that the target is to construct a continuous tone image, which is percepted by the eye as similar as possible to the halftone image. Not to actually achieve the original continuous tone image. The original image is typically not available, and we thus do not know when our reconstruction looks similar to it. It is really the halftone image and *not* the original image we should compare our reconstruction to.

1.3 OUTLINE

To read this thesis from start to finnish will probably be quite exhausting. But, depending on the reader's prior knowledge about inverse halftoning, and the reader's purpose with his reading, some chapters may be skipped.

Chapter 2 explains the basics of traditional halftoning. The Swedish reader may find it useful to look at the dictionary in *appendix C* sometimes, while reading this chapter.

Chapter 3 is supposed to give the reader some knowledge about the previous research made on inverse halftoning. One particular method is more thoroughly described in *chapter 4*. This chapter can be quite hard for the inexperienced reader, and is not essential for the final results of this report. For further research in this field, however, the method described here is very interesting. The same statement applies to *chapter 6*, which treats the concept of dot gain.

Some of my own work is reflected in *chapter 5*, which is an investigation over the behaviour of halftones in the Fourier domain. This chapter is essential for the understanding of the proposed method, described in *chapter 8*.

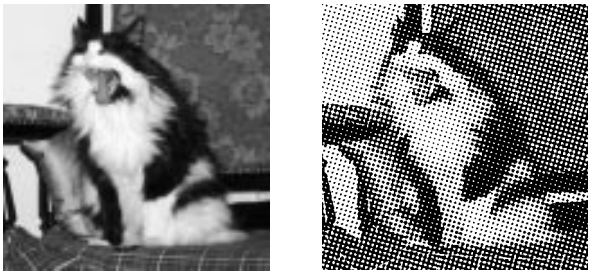
Chapter 7 holds a brief introduction to colour science, and is after that devoted to my attempts to perform reseparation of the printing colours.

The work is summarized in *chapter 9*, which also contains some suggestions for further work.

Reference images and some Matlab code are included in *appendices A* and *B*, respectively. *Appendix D* contains the used literature sources.

1.4 ABOUT THE IMAGE REPRODUCTION

To visualize the continuous tone images, a dye diffusion printer or something similar would be necessary. In this document the continuous tone images will instead be reproduced with a fine screen. The actual halftone images will be halftoned with a more coarse screen to distinguish them (see figure 1.2).



a *b*

Figure 1.2 *Examples of images. One continuous tone image (a, reproduced with a fine screen) and one binary halftone image (b, halftoned with 40 lpi).*

Because this report is printed in only one colour, the process colours have to be reproduced as different gray levels. The actual colour of the object will also be marked with a letter, as in figure 1.3.

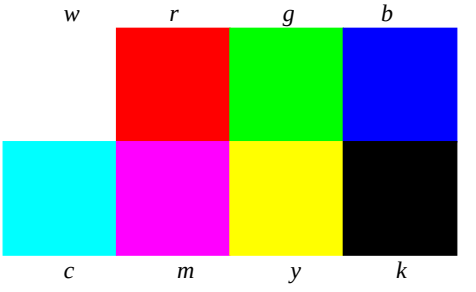


Figure 1.3 *The base colours of the common printing processes. White, red, green, blue, cyan, magenta, yellow, and black (which is represented by the letter "k").*

1.5 ACKNOWLEDGMENTS

I would like to thank Anders Gustafsson, author of [AG96], for lots of valuable comments on this thesis, and for many interesting discussions. I would also like to thank the rest of the staff in the *Graphic Arts and Image Processing Group* (which is a subdivision of the Image Processing Laboratory), for support and a friendly atmosphere.

Chapter

2. BASIC CONCEPTS

Graphical production is a large commercial field, with lots of notions and advanced technology. To understand the difficulties in, and the need for, inverse halftoning, it is important to be familiar with some of these notions. This chapter provides the basic concepts of printing, producing, and scanning halftone images.

2.1 THE PRINTING PROCESS

To transfer a document or an image from a computer to paper, a printer or a press is needed. Since printers are used only for small amount editions, they will not be considered here. There are three dominant press types: *offset* (lithography), *letterpress* (flexography), and *gravure*. The three technologies require different types of *printing plates*.

The classical letterpress uses mirrored printing plates, where the areas that are supposed to be covered with ink are raised from the surface of the plate. Gravure also use mirrored printing plates, but here the areas to be printed are depressed from the surrounding plate surface, leaving thin channels to be filled with ink. The offset press uses non-mirrored plates, where the areas that are supposed to be covered with ink, are neither raised, nor depressed from plate surface. Instead the printing plate has two chemically different surface types—one water repellent, and one not. The term "offset" is derived from the fact that the printing plate transfers the ink to an intermediate rubber cylinder, which then transfers the ink on to the paper. This is also why the offset printing plate is made non-mirrored—it is mirrored on the intermediate cylinder. The rubber cylinder gives several advantages. The running life of the plate is increased, for example. For illustrations of how the different press types works, please refer to other literature, for example [Wed95].

Common for all these devices is the fact that each printing colour can only be printed binary. Either a spot is covered with ink, or it is not. There are no in-betweens.

The offset and gravure printing plates are usually made from (positive or negative) *printing films* created by an *imagesetter*. The setter can be described as an advanced high resolution printer.

2.2 WHY HALFTONES?

As exemplified in the previous section, most printing devices are not capable of reproducing continuous tone images. Rather the device has a set of process colours, usually one or four, which can only be printed binary. To simulate the continuous tone, halftones are used. If this is done in an appropriate way, the eyes of the viewer are fooled to see the desired continuous tone colour. Thus halftoning is the process of approximating real-valued pixels by binary pixels.

For each process colour one binary halftone image, called *printing separation*, is generated from the original image. Then the separations are printed on top of each other. In this way different colours in the original image can be simulated.

About the Terminology

In current literature there is a slight confusion of notions between the words *dithering*, *screening* and *halftoning*. Some authors treat them as equals, while others give the words slightly different meanings. In this thesis the words have the following meanings:

- *Halftoning*—the process of approximating real-valued pixels by binary pixels.
- *Halftone or halftone image*—an image which has been produced by the halftoning process.

- *Screen*—the actual regular pattern of small dots, which constitute the halftone image.
- *Inverse halftoning*—the process of reconstructing the continuous tone image from its halftone image. Since halftoning is information lossy, this reconstruction has to be an approximation.
- *Dithering and screening*—are not used in this thesis.

I would like to point out that the concepts *halftone image* in English and *halvtonsbild* in Swedish does *not* mean the same thing. Halftone image is translated to "rasterbild", while "halvtonsbild" means a continuous tone image. Why this is the case, I do not know.

2.3 METHODS FOR HALFTONING

There are four main classes of halftoning methods:

1. Halftones constructed of symmetrical dots ordered in regular patterns. This is usually the case for *look-up table* (LUT) *halftoning*.
2. Halftones constructed of non-symmetrical dots ordered in regular patterns. This is the case for *threshold halftoning*.
3. Halftones constructed of symmetrical dots in non-regular patterns. This is the case for *frequency modulated halftoning* (FM) such as *error diffusion*.
4. The fourth class is non-symmetrical dots in non-regular patterns. No well-known halftoning method uses this hybrid technique.

The first two classes are good at reproducing smooth areas with slowly varying colours. The third class is better for detail sharpness. In paper printing, the class two halftones are dominant, while class three is dominant when halftoning for a computer display. The graphic industry is struggling to find good ways to go towards class three halftone also in paper printing. This demands better accuracy in several steps of the printing process.

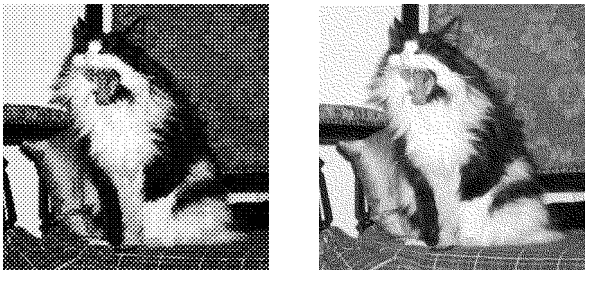


Figure 2.1 *Class 2 (left) and class 3 (right) halftoning. The right image may appear darker due to dot gain (see section 2.4).*

This thesis is concerned with the first two classes, where the halftone dots are ordered in regular patterns. These are currently the most common methods for paper production halftoning, and are based on the traditional photographic halftoning methods. They are expected to be dominant in large edition printing (such as newspapers), for several years yet.

Clustered Dot and Dispersed Dot

The first two method classes could each be divided into two subclasses: halftoning with *clustered dot* and *dispersed dot* (see figure 2.2). Both dot types are created by several very small micro dots (figure 2.3).

The dispersed dot preserves edges and high frequency textures well. Halftoning with dispersed dot is rarely used in paper printing, but is sometimes used in monochrome digital displays. The reason why the dispersed dot is not used when printing on paper, is the difficulty of preserving the shape of the micro dots. This difficulty becomes very visible in figure 2.2. The image at the bottom is generally too dark, and shows a gray level step at about 50 percent gray (that is: in the middle of the image). For darker tones than this there are no unprinted micro dots next to each other, and because they are surrounded by ink, the unprinted micro dots do not appear light enough. This is why the clustered dot is the traditional solution. With a larger clustered dot, the relative deformation effect of the dot shape, is reduced.

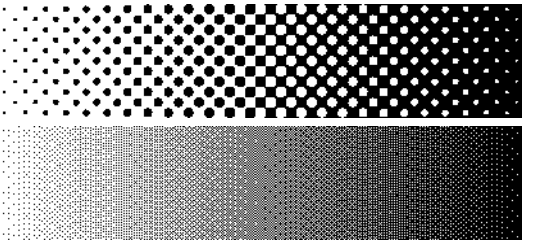


Figure 2.2 *One clustered dot halftone (uppermost), and one dispersed dot halftone, both at the same screen frequency. Threshold matrices according to [Ulich87].*

One of the main advantages of the dispersed dot is that the screen pattern becomes less visible. As mentioned earlier the disadvantage is that when the microdots are made as small as desired their shape is damaged in the printing process.

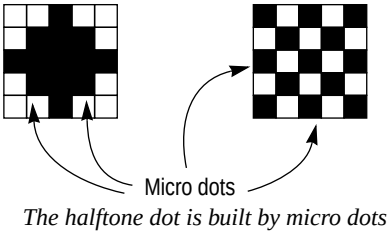


Figure 2.3 *The halftone dot is built by micro dots in the screen cell. To the left: clustered dot, and to the right: dispersed dot. Both at 52% gray.*

As this thesis is concerned with *printed* images it will from now on only focus on clustered dot halftoning. Let us begin by studying the available parameters for controlling the halftoning process.

Frequency and Angle

Each halftone dot belongs to a *screen cell*. The two (commercially) existing cell shapes are rectangular cells and hexagonal cells. Of these two the rectangular one is the most common, and is focused on in this thesis. The shape of the cells is completely defined by three parameter vectors. Two *screen vectors* and one *offset vector*. If the cell shape is rectangular the two screen vectors will be at right angles. If these two vectors also are of the same length, the screen cell will be a square. Square screen cells are the most common ones, when halftone images are produced. But in the printing and scanning processes, skew and quantization error might be introduced.

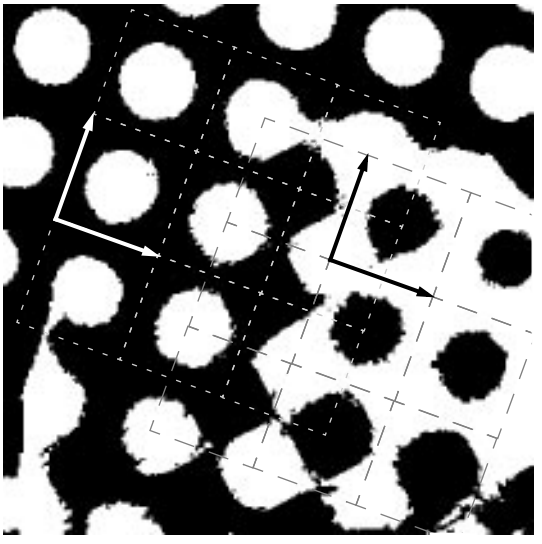


Figure 2.4 *Enlargement of a halftone image. Some of the white screen cells are dotted, while some of the black screen cells are dashed.*

As seen in figure 2.4 there are actually two grids of screen cells. One with black dots in the centre, and one with white dots. These are usually referred to as black screen cells and white screen cells, respectively. Note that the two cell grids are out of phase. It is more natural to talk about black screen cells (dots) in highlighted areas and white cells (dots) in shadowed areas. The arrows in the figure are the screen vectors in two cells. As they are at right angles and of equal length, the screen cells are squares.

When the cell shape is square, the screen vectors usually are replaced by the notions *screen angle* and *frequency*. The first parameter is measured in degrees and the second in *lpi* (lines per inch). This means the number of screen cells that are held in one inch of the image, in the screen vector direction — or in other words: the inverse of the screen vector length measured in inches. In figure 2.4 the screen angle is about 70°.

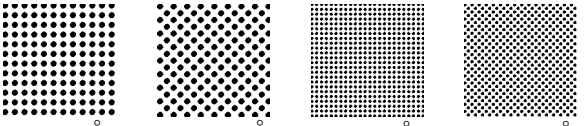


Figure 2.5 *The effects of the screen angle and frequency in a 30% gray image.*

In monochromatic printing the screen angle usually is set to 45°, since this angle has proved to hide the screen most effectively. When more than one process colour is used, each halftone image has to use an angle that differ as much as possible from the screen angles of the other colours. Otherwise the moire effect will produce artefacts in the printed image. This will be more carefully discussed in section 2.5.

The screen frequency parameter is chosen according to printing technique and paper quality. For daily newspapers and ordinary laser printing devices, 60 to 133 *lpi* is commonly used. For magazines 120 to 200 *lpi* is the standard. With modern printing techniques, it is possible to print with as much as above 400 *lpi*, but this is very rare.

LUT and Threshold Halftoning

The LUT halftoning is probably the simplest method to understand. For every real-valued pixel, a look-up table gives the design of the corresponding screen cell. If a larger table is acceptable, the design of the screen cell may depend on several adjacent real-valued pixels. This gives a better halftone image, but a slower processing time.

Threshold screening is the dominant method for paper halftoning. The design of the screen cells is determined from a threshold function, with the continuous tone image as input.

$$h(x,y) = T[f(x,y), c(x,y)] \qquad \textbf{[Eq 2.1]}$$

$$T[u,v] = \begin{cases} 1, & \text{if } u \geq v \\ 0, & \text{else} \end{cases} \qquad \textbf{[Eq 2.2]}$$

In equation 2.1, c is the continuous tone image, h is the binary halftone image, and f is a reference function. This function is often implemented as a *threshold matrix*. An illustration of threshold halftoning process is shown in figure 2.6. Note that the centres of the halftone dots are not necessarily at the same position as the peaks of the reference function. They depend rather of the local gradient of the image function.

A practical example of threshold halftoning is shown in figure 2.4. If LUT had been used, the dot shapes would have been more symmetrical.

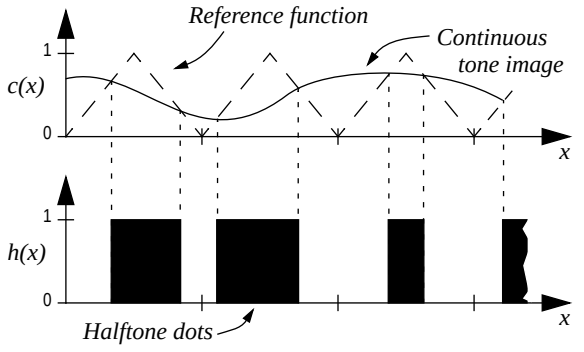


Figure 2.6 *One-dimensional illustration of threshold halftoning.*

Both methods can either use clustered, or dispersed halftone dots in the screen cells. The design of the look-up table or the threshold function determines which kind of halftone dot will be used.

Dot Shape

The design of the look-up table, or the threshold function also defines the dot shape of the clustered halftone dot. The usual shapes are *round*, *elliptical*, *square*, and *line* dots. Also combinations of these are used, so called *transforming* dot shapes. Which shape yielding the best result varies, depending on the characteristics of the picture. In general the transforming elliptical dot is considered the best. When the screen frequency is high, it is hard for the eye to discern which dot shape has been used. This is a desired property, since the intention (in most cases) is that the actual halftone dots should not be seen at all.

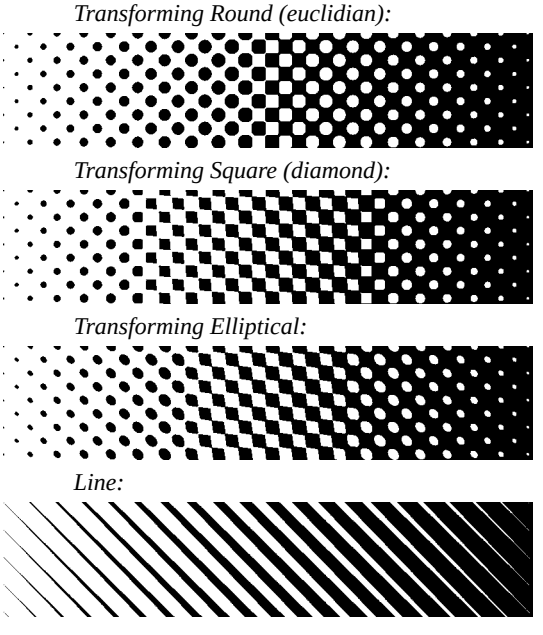


Figure 2.7 *Four different dot shapes at the same screen frequency (10 lpi) and angle (45°).*

Relations between Frequency and Resolution

The number of available gray levels when printing with a certain screen frequency, is depending on the resolution of the printer or the setter. The resolution is measured in *dpi* (dots per inch), which should not be mixed up with *lpi*, mentioned earlier.

The resolution gives the number of positions in the screen cell threshold matrix. The maximum number of gray levels is:

$$n = \left(\frac{\text{Number of dpi}}{\text{Number of lpi}}\right)^2 + 1 \qquad \textbf{[EQ 2.3]}$$

which is the number of micro dots in the screen cell plus one.

Due to quantization error, the actual number in equation 2.3 can be less for some screen angels. The function above is plotted in figure 2.8. When studying this graph, one should keep in mind that the PostScript language does not handle more than 256 levels of gray. When increasing the screen frequency, the number of printable tones is strongly reduced. When the reduction of tone levels becomes visible to the eye, as steps in smooth areas, the phenomenon is called *posterization*, illustrated in figure 2.9.

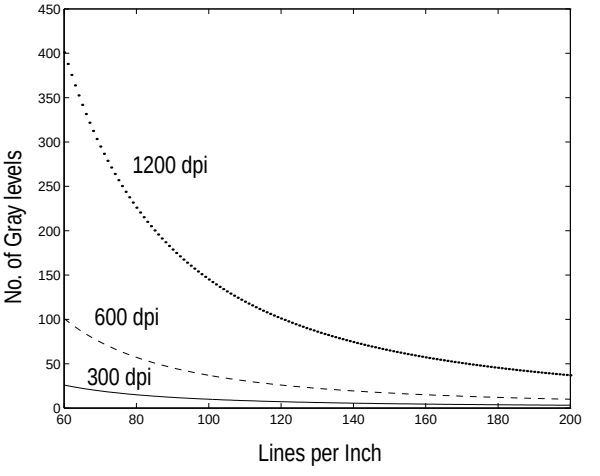


Figure 2.8 *The relation between dpi, lpi, and the number of gray levels. One hundred gray levels, or more, is desirable in high quality printing.*

The eye is normally able to distinguish about 64 gray levels, if they are apportioned over the whole light-to-dark range. Due to disturbances in the printing process at least 100 gray levels are necessary for good results. This means that when printing on a 600 dpi laser printer, no more than 60 lpi should be used, for full colour range. On the other hand; a coarser screen is more visible to the eye, so the choice of screen frequency has to be a trade-off between these two aspects.

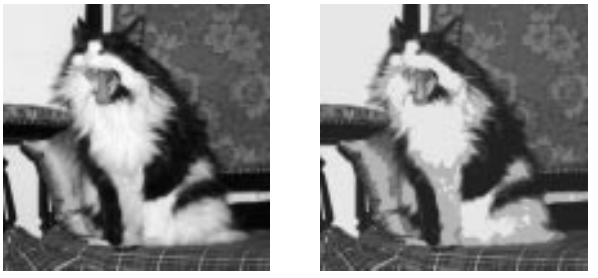


Figure 2.9 *Original image (left) and posterized version (right).*

There is also a relation between the desired number of *lpi* and the required number of *spi* (samples per inch) of the digital (scanned) version of the continuous tone image. To preserve edges and image details more than one continuous tone sample is required per halftone cell. The usual recommendation is at least four samples. This results in the following relation between the two quantities:

$$\textit{Number of spi} \geq 2 \cdot \textit{Number of lpi} \qquad \textbf{[EQ 2.4]}$$

The reason why the number only is doubled, and not quadrupled, is that when measuring the number of samples per area unit, both sides of equation 2.4 will be squared.

2.4 DOT GAIN

As mentioned earlier, the shape of the halftone dots are slightly deformed in the printing process. Usually the dots is percepted a bit larger, which, if not accounted for, yields a darker image. The phenomenon is called dot gain, and the dot gain can be separated into two different types.

Mechanical Dot Gain

The *mechanical dot gain* is a physical enlargement of the halftone dot during the printing process. The term *physical dot gain* is also used. The growth is the result of several interacting factors. For example, the viscosity of the ink, or the adjustment of the pressure from the impression cylinder of the press.

Optical Dot Gain

There are two types of *optical dot gain*. One type occurs in the human visual system, and is dependent of the frequency response of the eye. These limitations of the eye are in fact what makes halftones work, as simulation of continuous tone images. Other aspects of the limitations yields unwanted effects (see figure 2.10).

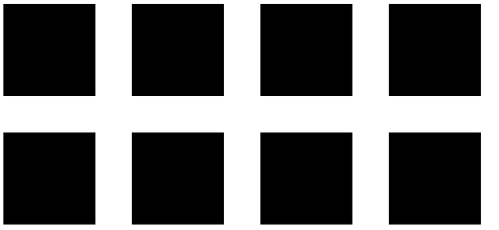


Figure 2.10 *Illustration of optical dot gain. Due to the frequency limitation of the eye, phantom gray areas appears in the white areas between the squares.*

The other type of optical dot gain is caused by the light scattering which occurs due reflecting properties of the paper. Only about 5% of the incoming light is directly reflected by the paper. The rest of the light scatters and later emerges through top and bottom surfaces.

2.5 COLOUR SEPARATION

When colour tones are to be reproduced, usually a number of basis colours are used. For example, the picture on a TV screen is formed by dots of the basis colours *red*, *green*, and *blue*. When all the three colours are present, white colour is percepted, and when none of them are present, black is percepted. This is called *additive* colour mixing and is illustrated in figure 2.11. The spectral distributions of the lights are added.

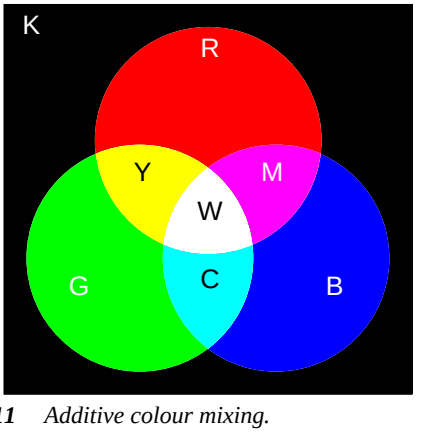


Figure 2.11 *Additive colour mixing.*

Unlike the TV, a piece of paper does not add energy to the illuminating light. The printed colours then could be seen as filters which absorb some of the incoming light before reflecting it. The more colours printed on the same spot, the more filters will subtract energy at certain wavelengths from the illumination. This is called *subtractive* colour mixing and is illustrated in figure 2.12.

In subtractive colour applications, the basis colours *cyan* (*C*), *magenta* (*M*), and *yellow* (*Y*), are usually used. When all the three colours are present, most of the light is absorbed and black is percepted.

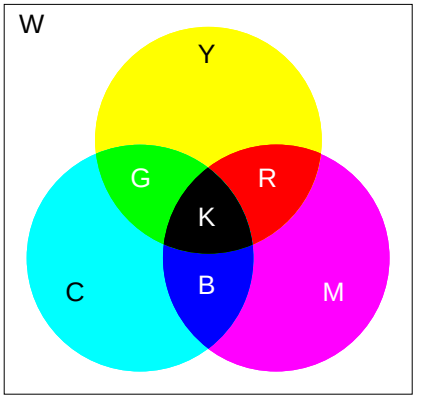


Figure 2.12 *Subtractive colour mixing.*

When printing colour images, four colours are usually used. The three basis colours above (*CMY*), and pure *black* ink (*K*). The reason for using black, despite the fact that it can be produced by the other three colours, is practical. There are three main reasons:

- It is cheaper to print large black areas with black ink than with both cyan, magenta, and yellow ink.
- There is a large risk that small black texts etc. will be a bit blurred if printed in three colours, due to misregistration between the printing plates of the three process colours.
- Introducing the fourth colour, the shadowed areas of images can be printed with more colour depth, and a richer blackness.

CMYK and RGB Colour Modes

When a colour image is scanned and processed in a computer, *RGB* mode is usually used. The standard format for this is 24 bits — eight for each colour. This means that the level of each colour can be adjusted in 256 steps. When such an image is to be reproduced it has to be converted to (32 bits) *CMYK* mode. The transformation is device dependent. Several choices have to be made. The PostScript language defines the standard transformation as follows:

$$\begin{cases} C = min(1.0,\ max(0.0,\ c - UCR(k))) \\ M = min(1.0,\ max(0.0,\ m - UCR(k))) \\ Y = min(1.0,\ max(0.0,\ y - UCR(k))) \\ K = min(1.0,\ max(0.0,\ BG(k))) \end{cases} \quad \textbf{[Eq 2.5]}$$

BG(k) and *UCR(k)* are invocations of the *Black colour Generation* and *Under Colour Removal* functions respectively. They decide how the pure black colour is to be used, compared to the other three printing colours. The temporary variables *c*, *m*, *y*, and *k* are defined as:

$$\begin{cases} c = 1.0 - R \\ m = 1.0 - G \\ y = 1.0 - B \\ k = min(c,\ m,\ y) \end{cases} \quad \textbf{[Eq 2.6]}$$

In the PostScript case the definitions of *UCR* and *BG* are device dependent. When the colour mode conversion is done manually by for example the Adobe PhotoShop software, the function definitions for these two functions are affectable. This gives a great deal of freedom for the user, but it is also a large source of errors, since the colours of the printed image could look different than desired.

Halftones of Colour Images

When the continuous tone CMYK image is to be printed it has to be halftoned. The four colour channels are then separated from each other and individually halftoned. To avoid moire effects (or better: to reduce them) the four halftones use different screen angles. The darkest colour (black) is thereby placed at the “best” angle, which is 45°. The other colours are then traditionally located at 75° (cyan), 15° (magenta) and 0° (yellow). The angles for cyan and magenta is sometimes switched. When halftoning is performed on a computer, these angles does not work optimally due to quantization effects. The angles, as well as the screen frequencies, in some cases have to be slightly mutually adjusted to prevent the moire. The moire effect is illustrated in figure 2.13.

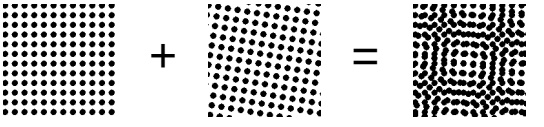


Figure 2.13 *The moire effect yields unwanted patterns.*

When the four halftone screens are correctly adjusted, the only moire effects left are small *rosettes* with open centres (see figure 2.14). That is: there is no halftone dot of any colour in the centre of the rosette. This gives a robustness, because it resists colour shifts in the image even when slight misregistration occurs.

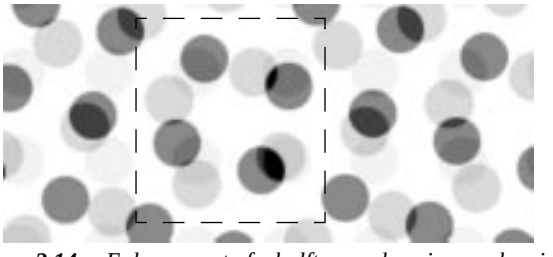


Figure 2.14 *Enlargement of a halftone colour image showing a slightly distorted rosette. Process colours are simulated by grays.*

2.6 THE COLOUR SCANNING PROCESS

There are two main methods for image scanning, *photo multiplicator tubes* (PMT) and *charge-coupled devices* (CCD). Both transforms light levels to continuously varying levels of voltage, which are sampled through an A/D-device. For flatbed scanning devices, CCD is used. This technique is described more thoroughly below.

Charge-Coupled Devices

Flatbed scanners are provided with a linear CCD-matrix with thousands of CCD-elements on a single silicon chip. To distinguish colours, each CCD-element is provided with one out of three different colour filters. One filter type each for red, green, and blue. The incoming light gives rise to a proportional electronic charge in the elements. The analogous charge is systematically transferred along chains of CCD-elements to A/D-converters where the signal is sampled to digital RGB form. A good introduction to the scanning process is given in [Agfa94].

2.7 WHY INVERSE HALFTONING?

There are several situations, where there is need for inverse halftoning:

- When an already halftoned and printed image is to be used in digital publishing systems.
- For data compression of scanned halftone images.
- When converting offset printing plates to gravure plates.
- To achieve an image that is mathematically comparable to the original image, when measuring SNR or RMS[1] of printed images.

When traditional image processing, such as sharpening and tone control, is adapted to an already halftoned image, the result is often poor and unpredictable. Also, when a halftone image is scanned and then reproduced once more, moire effects and colour shifts are likely to appear. Therefore it is desirable to reconstruct an image which is similar to the original continuous tone image. As mentioned earlier an exact inverse to the halftoning process is not possible. The task is thus to reconstruct the continuous tone image, with as little loss of information as possible.

The simplest methods are either defocusing the scanner optics, or to digitally low pass filter the halftone image, to smear the halftone dots over the whole screen cell. The second of these methods is often performed by using a gauss kernel, to avoid artefacts from the kernel shape in the resulting continuous tone image. This works well on images with low frequency contents, but makes details unsharp. Lots of different methods have been proposed to solve this problem (see chapter 3, “Methods for Inverse Halftoning”). However, none of these reports deals with colour images—and this is the motive for this thesis.

1. SNR is short for *signal to noise ratio*, and RMS is short for *root mean square error*. Both are distortion measures.

Chapter 3.

METHODS FOR INVERSE HALFTONING

This chapter summarizes the different approaches for inverse halftoning found in the current literature. No one has to my knowledge published any reports on inverse halftoning of colour images, so the methods below treat gray level images.

3.1 INVERSE HALFTONING INVOLVES LOW PASS FILTERING

Since traditional threshold halftoning includes a threshold operation, it will add high frequency energy to the image. The inverse process then must have some kind of low pass filtering effect. The halftone dots have to be smeared out over the area around them. The process can be made in several linear or non-linear ways, but the fact remains—inverse halftoning involves some kind of low pass filtering. Thus, we are bound to loose information from the original image if it is not frequency limited.

3.2 CLASSIFICATION OF METHODS FOR INVERSE HALFTONING

It is important to classify the inverse halftoning methods. Thereby the choice of a method suited for the existing input data, is simplified.

Linear Methods

A method that only applies linear operations to the image during the reconstruction process is called a *linear method*. Examples of linear operations are: spatial convolution, resampling, and interpolation. The linear methods are often simple to implement, and have nice mathematical properties.

Non-linear Methods

All methods that are not linear are characterized as *non-linear methods*. These methods could for example utilize neural networks, statistical methods, or advanced parameter estimation.

Orthographic Methods

A halftone image is *orthographic* if the coordinates in the halftone directly corresponds to exact coordinates in the original continuous tone image. This means that the halftone image has not been rotated, scaled, skewed, or exposed to noise. It implies that the halftone image never has left the computer—otherwise these properties can not be fulfilled. A method for inverse halftoning that requires orthographic input images is said to be an *orthographic method*. Because of this, no orthographic methods can be used within the scope of this thesis, since the target input image is printed and then scanned.

More or less all orthographic methods are also non-linear. If they were linear, there probably would not be any need for orthographic input. The orthographic methods often involves very precise estimation of screen parameters, such as the thresholding matrix. If the matrix also is known, the reconstruction can be made even more accurate.

Non-orthographic Methods

The non-orthographic methods does not require orthographic input images. Thus, these are the methods that are applicable to scanned halftones.

Presumed Halftoning Methods

An important aspect when classifying the methods for inverse halftoning is what kinds of halftoning algorithms they recover from. There is large field of methods specially designed for FM halftones. Especially recon-

struction from error diffused halftones is often discussed. These are however not interesting within the scope of this thesis.

3.3 REPORTED NON-ORTHOGRAPHIC METHODS

This thesis focus on scanned input images. This implies that the input data will not be orthographic—and that decreases the number of available methods.

DGP based Inverse Halftoning

Perhaps the most promising method for non-orthographic images is developed by Forchhammer et al. at the Denmark Technical University. It is a linear method designed for gray level images, but can also handle colour images if the printing colours can be separated. This is easy if the printing films can be scanned separately, but is a harder task when the printing colours already are printed on top of each other. Eskofot DGS has developed commercial software and hardware that implements these methods, but no such equipment was available for reference during the writing of this thesis. It is very expensive and is sold in very exclusive quantities.

The method will be described in chapter 4, "DGP-based Inverse Halftoning". If there is a possibility to separate the printing colours, then this method, or a modification there of, is probably superior to any other method when it comes to reconstruct continuous tone from scanned clustered dot halftones.

Plain Linear Filtering

The design of filter kernels for linear filtering is discussed in [Joh96]. In that thesis four different spatial kernel types are compared:

- *High resolution cubic spline.* Approximation of a two-dimensional *sinc* function, covering 4×4 screen cells.
- *Cubic spline.* Approximation of the central lobe of the sinc function, covering 2×2 screen cells.
- *Square.* Flat averaging kernel covering one screen cell.
- *Gaussian.* Smooth gauss-shaped kernel, covering 2×2 screen cells.

The best signal to noise ratio was achieved using the square kernel and the cubic spline kernel. The square kernel produced sharper images than the cubic spline, but the sharp edges of the kernel produces some blocky artefacts in the reconstructed image. Since the square kernel is four times smaller than the cubic spline kernel, it will be much faster to apply.

The linear filtering can also be performed in the Fourier domain. This is examined in chapter 5, "Halftones in the Fourier Domain".

Chapter

4. DGP-BASED INVERSE HALFTONING

This chapter is a brief compilation of the results produced at Denmark Technical University, by Søren Forchhammer et al. (see [Forch91], [Forch94], and their other reports, listed in section D.2). In this chapter the halftone image is assumed to be halftoned by clustered dots, ordered in regular patterns. This means halftones of type one and two, as described in section 2.3. The input image is assumed to be binary. This induce a loss of information, if the image scan is made in gray level mode. Therefore, it becomes very important that it is possible to reconstruct an accurate binary image from the dot gained gray level image. This problem is treated in chapter 6, "Modelling of the Dot Gain". The DGP-based inverse halftoning is not used in my proposed method (see chapter 8, "the Proposed Method"), due to problems with the colour classification. Though the DGP-based methods is very interesting for further developments of the method proposed in chapter 8.

There are several similar methods proposed, based on DGP. They all have in common that the screen parameters are estimated, and the halftone image is then filtered *in phase* with the screen pattern. The accuracy of the phase is so important, that a polynomial lattice is used (described in section 4.2).

The target with DGP-based methods is to achieve a sample resolution of more than one sample per screen cell. This gives a better sharpness in detailed areas of the image.

4.1 THE SCREEN LATTICE

The dots in a halftone image are ordered in a lattice called *screen*. Each black halftone dot belongs to one *black screen cell*. The white areas between the dots are sometimes referred to as white dots. Their *white screen cells* are displaced by half a cell compared to the black ones (see figure 4.3). The size and shape of the cell is defined by two *screen vectors*. The relation between the coloured part of the cell (the area of the dot) and the total area of the screen cell is called the *blackness* of the cell.

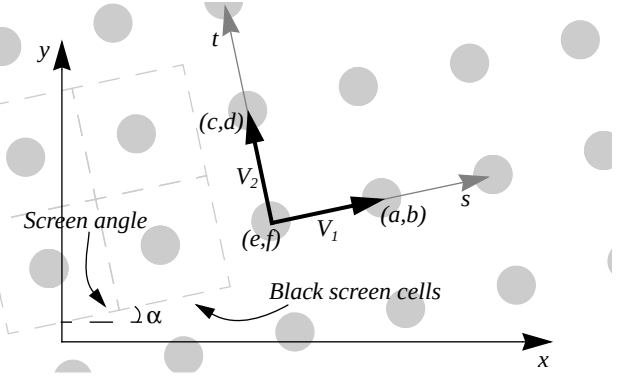


Figure 4.1 *Illustration of some basic concepts. Please note that the coordinates a, b, c, d, e, and f are given in the xy-system.*

If the lattice is linear, the centres of the screen cells are located at integer values for s and t, in the equation below:

$$\begin{aligned} x(s,t) &= s \cdot V_{1x} + t \cdot V_{2x} + e \\ y(s,t) &= s \cdot V_{1y} + t \cdot V_{2y} + f \end{aligned} \qquad \textbf{[EQ 4.1]}$$

where $(e, f) \in \Re^2$ is the offset vector, and the screen vectors are defined as:

$$\bar{V}_1 = \begin{pmatrix} a-e \\ b-f \end{pmatrix}, \quad \bar{V}_2 = \begin{pmatrix} c-e \\ d-f \end{pmatrix} \qquad \textbf{[EQ 4.2]}$$

If $V_{1x}=V_{2y}$ and $V_{1y}=-V_{2x}$, then the screen cells have square shapes. All this is illustrated in figure 4.1.

When the screen vectors have non-integer length there will be numerical rounding problems. Then it is a non-trivial task to tell which of two screen cells a pixel belongs to.

The vectors may be estimated with good accuracy from the Fourier spectrum of the halftone image—see section 5.5.

4.2 THE POLYNOMIAL LATTICE

When scanning printed halftones, a slight distortion of the screen is unavoidable. This makes the linear screen model inadequate, according to Forchhammer. A better approximation is to use a second degree polynomial approximation, between four known values located in the corners of the considered image block.

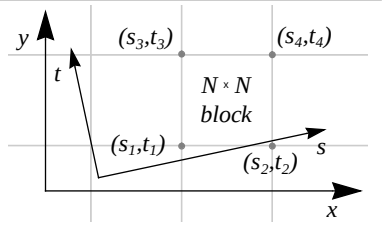


Figure 4.2 *With polynomial approximation the screen cell centres are interpolated from four known values. Please note that the coordinates are given in the st-system.*

With terms as in figure 4.2, the interpolation of more accurate *st*-coordinates can be written as:

$$s(\tilde{x},\tilde{y}) = s_1 + (s_2 - s_1)\frac{\tilde{x}}{N} + (s_3 - s_1)\frac{\tilde{y}}{N} + \\ + (s_4 - s_3 - s_2 + s_1)\frac{\tilde{x}\cdot\tilde{y}}{N^2} \qquad \textbf{[EQ 4.3]}$$

$$t(\tilde{x},\tilde{y}) = t_1 + (t_2 - t_1)\frac{\tilde{x}}{N} + (t_3 - t_1)\frac{\tilde{y}}{N} + \\ + (t_4 - t_3 - t_2 + t_1)\frac{\tilde{x}\cdot\tilde{y}}{N^2} \qquad \textbf{[EQ 4.4]}$$

In the equations above, $\tilde{x}$ and $\tilde{y}$ are the *x* and *y* coordinates when the origin of the *xy*-system has been moved to the lower left corner of the block (the corner indexed by number one in figure 4.2).

If the blocks are small enough (about 256×256 pixels), it is possible to achieve an accurate, phase continuous description of the screen lattice. The adjacent blocks have to use the same coordinates for the shared corner points, to achieve phase continuity.

4.3 CELLS AND TRIANGLES

To achieve a higher-than-screen resolution the screen cells are partitioned into four triangles (illustrated in figure 4.3). In this way we can utilize possible asymmetries of the halftone dot to detect image details. Since every pixel belongs to two screen cells—one black and one white—which can both be partitioned, it is possible to get eight gray values from each halftone dot. This method, among others, is proposed in [Forch91], but due to the quantization errors discussed in section 4.4, this approach gets very sensitive to skew and noise. If eight samples per cell are used, some of the samples will be bad.

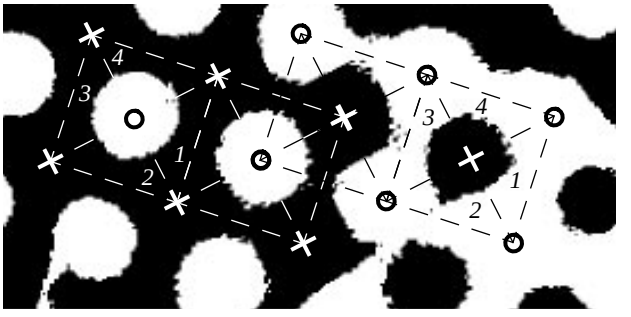


Figure 4.3 *Black and white screen cells partitioned into four triangles each. White screen cell centres are marked by rings, while black cell centres are marked by crosses.*

4.4 DIGITAL GRID POINTS

A digital representation of the screen cell centres will differ slightly from their actual location, due to quantization errors. This is exemplified in figure 4.4. In practice, however, the relationship between the sample intensity and the distance of adjacent micro dots is better than in this example. This means that the dotted grid is finer, yielding smaller differences between the black and white circles, symbolizing the halftone microdots and their sampled versions.

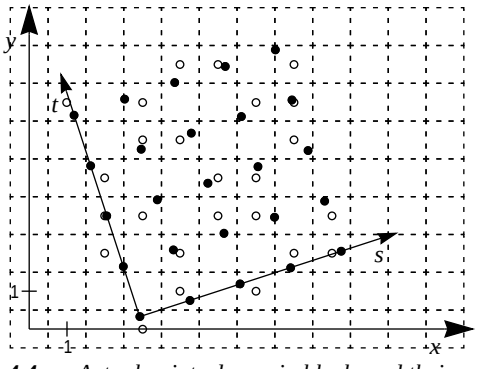


Figure 4.4 *Actual points drawn in black, and their corresponding digital versions drawn in white. The quantization errors are obvious. To lower this error the sample intensity is increased.*

The actual definition of a digital grid point segment is formulated by Forchhammer as:

> ***Definition.*** *A set of lattice points, S in two dimensions, is a digital grid point segment (DGPS) over G (the st-system above), if there is a set of grid points, A in G, whose digital image D(A) is S. If A has integer distances between adjacent grid points, S is a digital regular grid point segment (DRGPS).*

This mathematical definition is fundamental for the DGP-based inverse halftoning. The scanned version of the screen cell is considered a DGPS. This is illustrated

in figure 4.5. In practice a finer sample grid is used, yielding more triangular shapes of the four cell partitions in figure 4.5c.

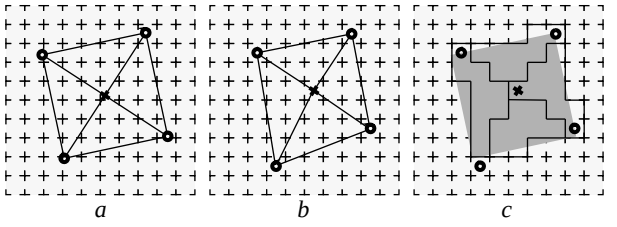


Figure 4.5 *A screen cell (a) is sampled. The centre and corner points are then quantized (b), which distorts the quadratic shape of the cell. The final digitalization of the square and the triangles (c) is drawn over a shadowed version of the original cell.*

Observe that this digital version of the cell neither correspond to a digitalization of the bounding lines of the continuous triangles, nor do they have uniformity in area. However the variation is limited to one pixel, since the digital triangles are balanced. In figure 4.5c two of the distorted triangles have 8 pixels, while the other two have 9.

The impact of the quantization error is decreased by restricting the use of white cells to dominantly dark areas, and black cells to light areas. In this way, most of the quantized boundaries are located in the dominant colour. Thus the number of pixels of non-dominant colour is less affected by the quantization. This error reduction is done on the expense of the algorithm complexity.

4.5 HIGHLIGHTS AND SHADOWS

As stated in the previous sections, the black and white screen cells are partitioned into four triangles each. But to reduce the noise, only triangle values from white cells are supposed to be used in shadowed areas, and black cells in highlighted areas. In areas with little or no details higher gray value accuracy may be achieved by averaging over the whole cell, instead of over each of the four triangles. The question is now how to decide which triangles to use, in a simple manner. There are several ways proposed, on how to use the triangles (see [Forch91]), but here I have here chosen to briefly present the algorithm presented in [Forch94].

The first step is to traverse all the black screen cells of the image block, and through averaging decide if the cells are located in shadowed or highlighted areas. When this is done, the table in figure 4.6 tells us which triangles or cells to use. The table is designed to tell which triangles are good and which are bad—and also when to use complete cells, instead of triangles. In smooth areas a complete cell gives a more accurate value than each triangle, because the averaging is performed over a larger area.

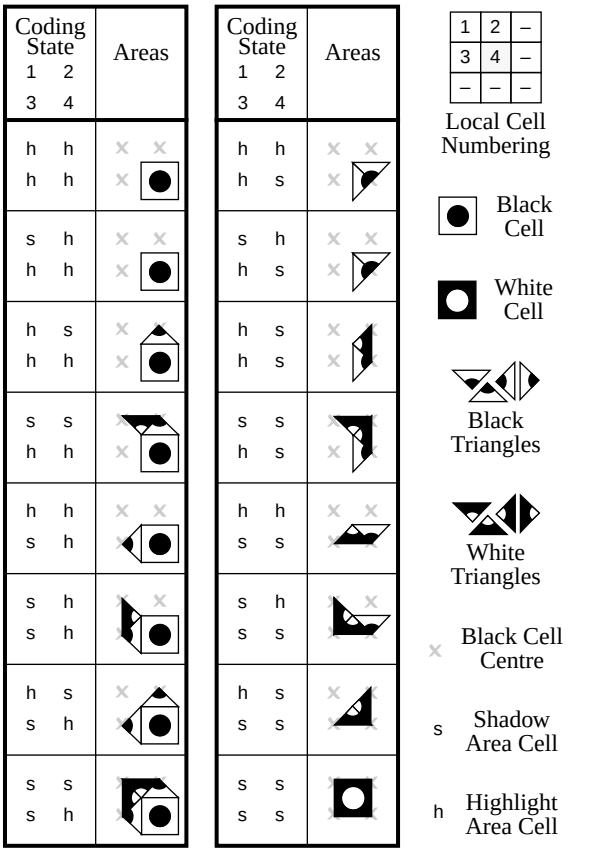


Figure 4.6 *State table specifying which kinds of areas that are supposed to be used for the averaging process, when calculating the gray level.*

What is not so easy to see from the table, is that, by this coding, there will be no overlap of black and white triangles. But this is the case. The partitioning of the image is thereby unique. Proof of this theorem is found in [Forch94].

An application of the table in figure 4.6 is shown in figure 4.7. When coding an image block it is sometimes necessary to have knowledge of the adjacent image blocks' edge cells. In figure 4.7c, some of the border cells are coded with such knowledge.

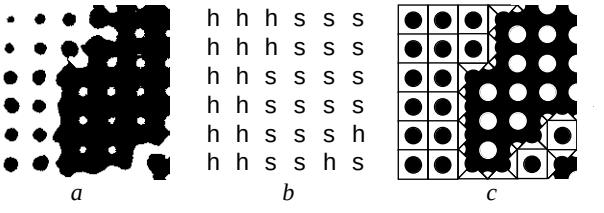


Figure 4.7 *A halftone image is coded by the table in figure 4.6. Some of the edge cells are coded by knowledge of the adjacent image boxes' edge cells.*

4.6 CALCULATION OF BLACKNESS

The blackness of the triangles and cells is calculated as the quotient between the number of black pixels, and the total number of pixels in the triangle or cell. The achieved blackness is then assigned to all gray level pix-

els corresponding to the triangle or cell. A better image can be achieved if the calculated gray levels are assigned to the central coordinate of the triangle or cell, and the gray level pixels are interpolated from the blackness assigned to the few closest of these central coordinates.

4.7 ENHANCEMENT BY USE OF GRADIENTS

It is possible to enhance the reconstruction even further by estimating the local derivatives. Let A_1, A_2, A_3, and A_4 be the averaged gray levels, in the four triangles in a screen cell. The triangle numbering is according to figure 4.3. Then the partial derivatives may be estimated as:

$$f_s' = \frac{\alpha(A_1 - A_3)}{\sqrt{A_1 + A_2 + A_3 + A_4}} \qquad \textbf{[EQ 4.5]}$$

$$f_t' = \frac{\alpha(A_4 - A_2)}{\sqrt{A_1 + A_2 + A_3 + A_4}} \qquad \textbf{[EQ 4.6]}$$

$$A_i \in [0, 1] \qquad \textbf{[EQ 4.7]}$$

The constant α in these equations is depending on the dot shape. If the dot shape is round, then $\alpha = 2\sqrt{\pi}$, and if the dot shape is diamond, then $\alpha = 2\sqrt{2}$. Now both samples of gradient and amplitude can be used to achieve an even better reconstruction. More about how this is done can be read in the articles and reports by Forchhammer et al. listed in section D.2.

Chapter

5. HALFTONES IN THE FOURIER DOMAIN

Very little of the literature found on halftoning regard the behaviour in the Fourier domain. However, a spectral point of view helps the understanding of several topics, concerning the ordered clustered dot halftoning process. This knowledge is very helpful when searching for an appropriate inverse process.

5.1 THE FAST FOURIER TRANSFORM

When a discrete signal, such as a sampled image, is to be Fourier transformed, the *discrete Fourier transform* (DFT) is used. In its two-dimensional form, with an image of $N \times M$ pixels it is defined as:

$$X_{N,M}[k,l] = \frac{1}{NM} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x[n,m] \cdot e^{-2\pi i \left(\frac{kn}{N} + \frac{lm}{M}\right)} \quad \textbf{[EQ 5.1]}$$

and the inverse formula is defined as:

$$x[n,m] = \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} X_{N,M}[k,l] \cdot e^{2\pi i \left(\frac{kn}{N} + \frac{lm}{M}\right)} \quad \textbf{[EQ 5.2]}$$

where $x[n,m]$ is the image.

This transformation is computationally heavy, so to speed up computations, an algorithm called the *fast Fourier transform* (FFT) is used. This enhances the algorithm complexity from $O(N^2M^2)$ to $O(NM\log NM)$. The algorithm uses the fact that the same multiplication is required, for the calculation of the value in more than one Fourier domain coordinate. For more information on the Fourier transform, please refer to [Brac86].

5.2 THE SIMPLIFIED HALFTONING MODEL

As stated in section 2.3, the functions defining the ordered threshold halftone image are:

$$h(x,y) = T[f(x,y), c(x,y)] \quad \textbf{[EQ 5.3]}$$

$$T[u,v] = \begin{cases} 1, & \text{if } u \geq v \\ 0, & \text{else} \end{cases} \quad \textbf{[EQ 5.4]}$$

The function f is a periodic reference function. This function is usually approximately a two-dimensional cosine function—possibly rotated and phase shifted:

$$f(x,y) = \frac{\cos(\omega(x\cos\varphi + y\sin\varphi) + \theta_1)}{4} + \frac{\cos(\omega(y\cos\varphi - x\sin\varphi) + \theta_2) + 2}{4} \quad \textbf{[EQ 5.5]}$$

The principle look of the Fourier transform of the function defined in equation 5.5 is shown in figure 5.1.

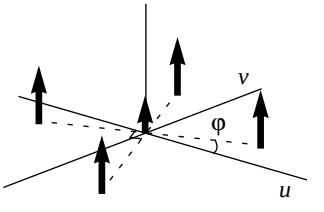


Figure 5.1 *The principle look of the Fourier spectra of a two-dimensional cosine function.*

To simplify equation 5.5 we assign θ_1 and θ_2 to zero, and the angle φ to 45°, which gives:

$$f(x,y) = \frac{1}{2} \cdot \cos\frac{\omega x}{\sqrt{2}} \cdot \cos\frac{\omega y}{\sqrt{2}} + \frac{1}{2} \quad \textbf{[EQ 5.6]}$$

By using equation 5.4, equation 5.3 can be rewritten as:

$$h(x,y) = T[0, c(x,y) - f(x,y)] \quad \textbf{[EQ 5.7]}$$

When Fourier transforming $h(x,y)$, we are thus interested in how the non-linear thresholding function $T[0, a(x,y)]$, is affecting $a(x,y)$ in the Fourier domain.

Let us begin by studying how T handles f alone, approximated by the two-dimensional cosine function, described above.

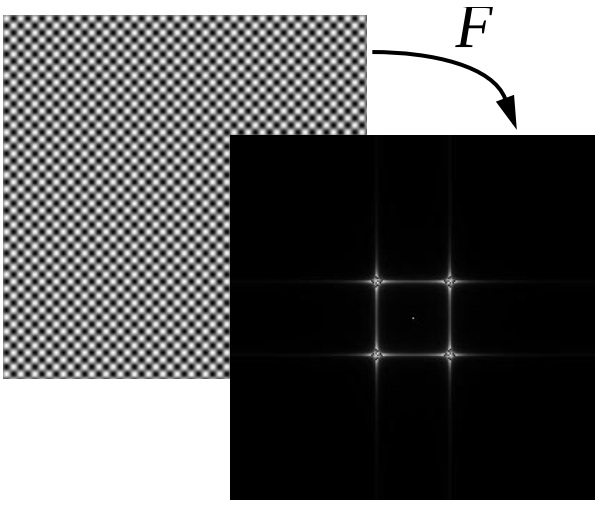


Figure 5.2 *The Fourier transform[1] of the reference function simulated by a two-dimensional cosine function rotated by 45°. The peak in the origin is due to the fact that the signal is strictly positive (it might not be visible in your copy but it is there).*

One might think that the Fourier transform of this f only should be five dirac impulses. However, as seen in figure 5.2, this is not the case. The ideal cosine function is spatially multiplied by a rectangular window at the size of the image. In the Fourier domain this is equal to convolution with a two-dimensional sinc function[2], since the Fourier transform of a rectangular window is a sinc—and spatial multiplication corresponds to a spectral convolution. This is why the four peaks are a little smeared.

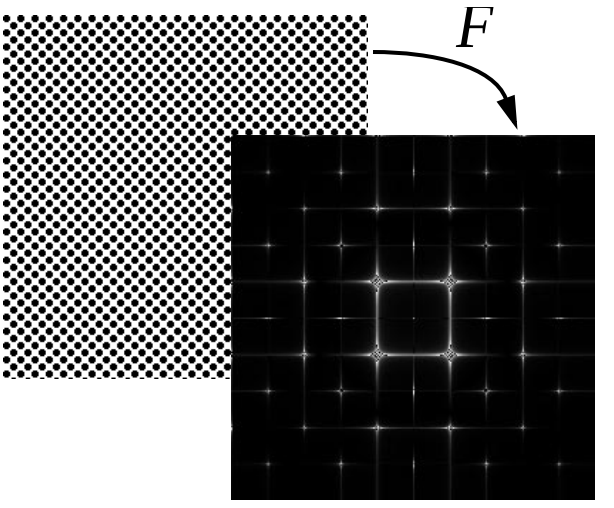


Figure 5.3 *The Fourier transform of the thresholded version of the reference signal. High frequency energy has been introduced as multiples of the original frequency parts.*

When the reference signal is thresholded, high frequency energy is introduced. As seen in figure 5.3, this high frequency energy is introduced in the form of exact integer multiples of the original frequency components.

Figure 5.4 *Original image and its Fourier transform.*

5.3 FOURIER TRANSFORMING A HALFTONE IMAGE

Now, let us consider a real image (figure 5.4) and its halftone (figure 5.5). The Fourier plot in figure 5.5 shows how the threshold operator handles the original image and the reference signal. Since the input image—whether it is the continuous tone image or the halftone image—only contains (positive) real values, the Fourier spectrum will be symmetric around the origin. That is: $F(u, v) = F(-u, -v)$ for all u and v.

---

1. Here, as in the following Fourier domain plots, it is actually the absolute value of the Fourier transform that is plotted. In all Fourier domain plots, the origin is in the centre of the intensity plot. (When performing the transformation in Matlab by the *fft2* function, the origin will be in the upper left corner.)

2. $sinc\ x = \frac{\sin \pi x}{\pi x}$. The two-dimensional version is:

   $F(u, v) = sinc\ u \cdot sinc\ v$

Figure 5.5 *Halftone image and its Fourier transform. Notice that the peak pattern from the reference signal and the low frequency parts of the original image still are present.*

Figure 5.6 gives important information when trying to find a the inverse function. The left the two plots, shows the frequencies that differ the most, when comparing the halftone image with the original image. The right plot shows the transfer function of the ideal filter that would map the halftone image back on the original image, in this particular case. It gives us a good hint why low pass filtering—and especially smooth gaussian low pass filtering—gives so good results when used for inverse halftoning. For low frequencies the gaussian low pass filter is a fair approximation of the ideal filter[3].

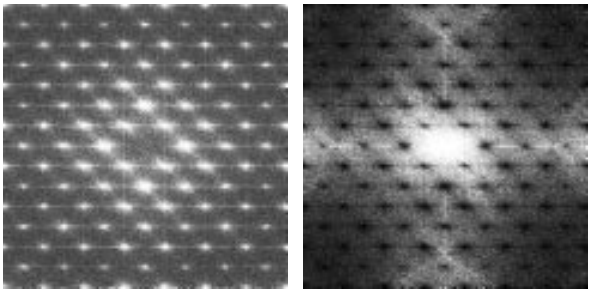


Figure 5.6 *To the left: the absolute difference between the spectra in figures 5.4 and 5.5. To the right: the transfer function of the linear filter that would map the halftone image back on the original image, in this particular case.*

For higher frequencies the ideal filter indicates that it is important to mask out the peaks derived from the threshold reference function. We would like to get a filter with gaussian-like behaviour for low frequencies and zero crossings at the halftone's peak frequencies.

3. Actually: this far we only know that the gaussian filter transfer function approximates the *absolute value* of the ideal transfer function—we know nothing about the *phase*, which also is an important aspect.

Inverse Halftoning using a Sinc Function

There is a simple solution with the characteristics described above: the two-dimensional sinc function, rotated to the right angle. Multiplying in the Fourier domain with a sinc function, is actually equal to spatial convolution with a rectangular and flat averaging kernel, at the same size and angle as the screen cells of the halftone image. The resulting image quality is quite good, although it is a bit blurred, and a bit blocky because of the sharp edges of the spatial kernel.

Figure 5.7 *Comparison of original image (left) and reconstruction (right) from the clustered dot halftone image in figure 5.5.*

The result of this filtering can be seen in figure 5.7. Except for a little lack of detail compared to the original image it is a fairly good reconstruction from the halftone image in figure 5.5. To give the sinc function in the Fourier domain the desired characteristics (zero crossings at reference function peaks), it is essential that the spatial averaging kernel has exactly the right size and angle. Otherwise the zero crossings will be misplaced. It is easy to realise that a spatially too small kernel does not remove the screen pattern, but due to the properties of the sinc function, a kernel that is bigger than necessary might work the same way. The result is shown in figure 5.8. Of course the reconstructed images are of continuous tone but are here halftoned again for reproduction. This might decrease the readers ability to spot the differences between the images. Most of the images in this chapter are also found in appendix A.

Figure 5.8 *On the left the same kernel size as in figure 5.7 is used, but at the wrong angle. On the right a too large kernel is used. The image is more blurred than necessary, but still shows remains of the screen pattern.*

Inverse Halftoning using a Gaussian Low Pass Filter

From the discussion above, it is understood that the sinc approach is quite sensitive to errors in the estimation of the screen parameters. By using a gaussian low pass filter with sufficiently low limit frequency, all the unwanted peaks are filtered. Of course also more other information, possibly deriving from the original image, is filtered. On the other hand we do not have to worry about having a spatially too large kernel, or a kernel rotated into the wrong angle. Because of this insensitivity, this method is commonly used in image processing toolkits like Adobe's Photoshop.

Then, what happens if we instead of the soft gaussian filter use a sharp, close to ideal, low pass filter, with limit frequency just inside the first halftone peaks? The result is shockwaves near edges, and is shown in figure 5.9, although the reproduction halftoning here hides these obviously negative effects. For a closer look: please refer to appendix A.

Figure 5.9 *On the left: gaussian low pass filtering with the highest possible limit frequency without any sign of the screen pattern. On the right: the effect of using a ideal low pass filter that cuts of the spectrum just before the first peaks.*

What about the Phase?

We have now seen that low pass filtering works quite well when performing inverse halftoning. But what about the phase? How does the halftoning operation change the phase in the Fourier domain?

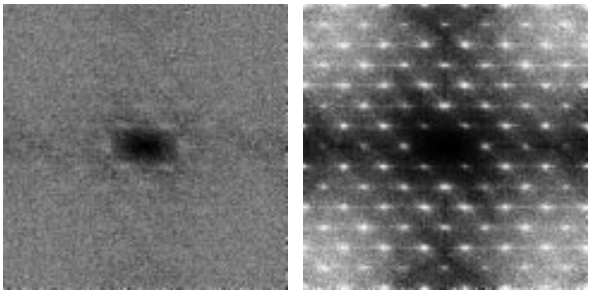


Figure 5.10 *On the left we see the absolute difference of the phase angles between the halftone image and the original image. On the right: the absolute relative error compared to the original image.*

As shown in figure 5.10 the error in phase as well as the relative amplitude error is very low for low frequencies. This is the reason why the phase preserving filters, tried earlier, have worked. The figure also indicates that it will be tough to construct a good filter that only masks out the peaks. Not only is the needed phase correction between the peaks, for high frequencies unpredictable —for higher frequencies the relative amplitude error is so large, that almost no information of the original spectrum remains.

5.4 FOURIER TRANSFORMS OF COLOUR HALFTONES

When trying to perform inverse halftoning on *colour* halftones more difficulties are introduced. The image is printed in the four primary colours cyan, magenta, yellow, and black. The scanner on the other hand usually uses red, green and blue. It is a very difficult problem to fully extract the four printing colours from the scanned *RGB* image. The reason for this is the problem with black colour. With four primary colours it is in micro perspective possible to create 16 different colours—but of these, 9 are more or less black! One of the black colours is generated by *C*, *M*, and *Y*, while the other 8 are combinations of the primary colours containing *K*. No one has to my knowledge yet succeeded in distinguishing between all 9 variants of black.

But there are more difficulties. Depending of the frequency characteristics of the filter in the scanner, there might be hard to distinguish between some of the other seven colours. The scanner used for this project (AGFA Arcus II) has great difficulties distinguishing printed magenta and red (which consists magenta and yellow).

The discussion above indicates that the inverse halftoning probably will have to be performed on colour channel images that have more than one screen pattern. This implies a new problem. The two screens will interact and create moire patterns (see section 2.5). And this proves to be a new difficulty when performing inverse halftoning.

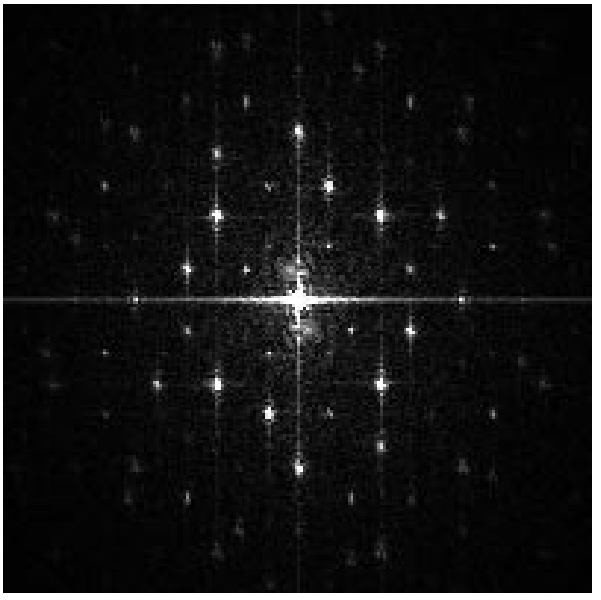


Figure 5.11 *The R (red) channel of a scanned colour halftone image. Traces of C, M, and K are clearly present.*

Figure 5.11 shows the Fourier transform of the red channel of a scanned colour halftone image. At a first look, the peaks does not seem to be reproduced at integer multiples of the lowest frequency peaks any more. But this is actually still the case, although the pattern is more complicated.

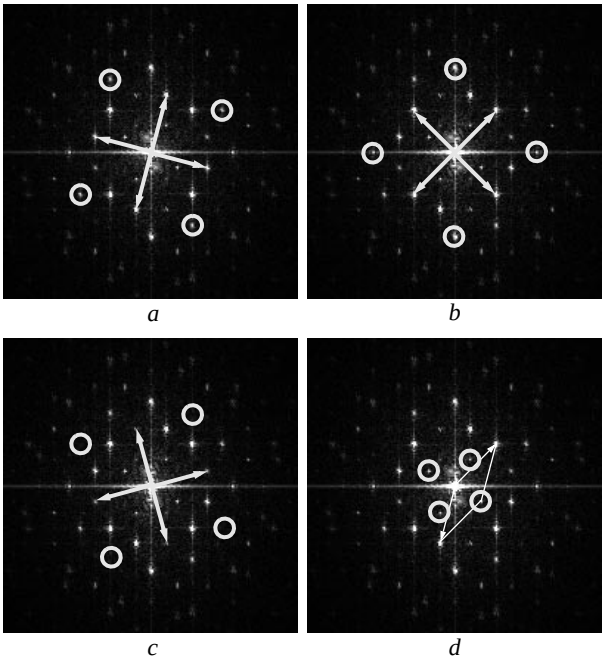
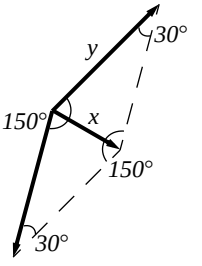


Figure 5.12 *The explanation to the different peaks found in figure 5.11.*

Image *a* in figure 5.12 shows the base peaks originating from the cyan colour halftone (marked by arrows), and their first integer multiples (circles). Images *b* and *c* show the same for black and magenta (magenta is weaker than the other two). Image *d* marks a few of the *moire peaks*. By this is meant peaks originating from integer multiple combinations from peaks of different colours. There are lots of them, but only a few are so strong that they are visible. In this case both cyan and black are strong so several combinations of these peaks are visible. Four of them (marked with circles in image *d*) gives sharp peaks for frequencies *lower* than the base peaks.

This moire effect yields that the lowest frequency component (peak) added to the original image is lower than the lowest peak of each primary colour screen. In this case we loose a factor of 1.93 (that is: the screen frequency has to be increased, with a factor 1.93 of the primary halftone image's, to have an equally invisible screen pattern):



$$\frac{y}{x} = \frac{1}{2 \cdot \sin\frac{30°}{2}} \approx 1.93$$

This calculation also implies why the moire effect is lowest[4] when the screen angles differ with 45°. This yields only a degeneration of a factor 1.31. When the difference between the screen angles get smaller the factor increases. This is pessimistic: we print with four colour, which gives us an average angle difference of 18° and a frequency degeneration factor of 3.20. But this is not the whole truth. Since yellow is lighter than the other three printing colours, the moire pattern involving yellow is less visible and can therefore be tolerated, to get less moire involving the other colours. The factor in our example of 1.93 is quite typical.

The factor above implies that we have to set the frequency limit to half of what we should need to, when applying the low pass filtering techniques. The only way to completely remove this unnecessary need for low pass filtering, is to completely extract the four halftone images (one for each printing colour) from the scanned version of the printed colour halftone image. And, as stated earlier, this is very hard to achieve.

5.5 ESTIMATING FREQUENCY AND ANGLE

It is possible to estimate both screen frequency and angle from the Fourier spectrum with high accuracy. There is a tight connection between the positions of the

4. This is true when we do not have full control of the spatial position of the screen pattern. Otherwise we could achieve no moire at all by giving all printing colours the same angle.

peaks in the spectrum and halftone dot periodicity. In the Fourier domain we define the vectors to the base peaks as follows:

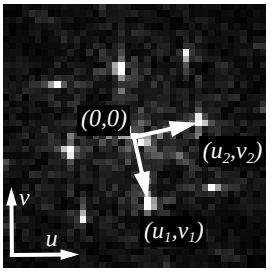


Figure 5.13 *Naming convention for vector variables. The Fourier plane has here been cropped. Only a partition around the origin is shown.*

$$u_1 > 0; \quad v_1 \le 0; \quad u_2 \ge 0; \quad v_2 > 0 \qquad \textbf{[EQ 5.8]}$$

Then, according to [Joh96] (the equations have here been slightly adjusted), the connection between the vectors in the Fourier domain and the screen vectors is:

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \frac{1}{u_1 v_2 - u_2 v_1} \begin{pmatrix} 0 & -W \\ H & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} \qquad \textbf{[EQ 5.9]}$$

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \frac{1}{u_1 v_2 - u_2 v_1} \begin{pmatrix} 0 & W \\ -H & 0 \end{pmatrix} \begin{pmatrix} u_2 \\ v_2 \end{pmatrix} \qquad \textbf{[EQ 5.10]}$$

where W and H are the width and height of the Fourier transformed image. For traditional clustered dot halftoning the two vectors above should be at right angles and have the same length. This will be the case for most of our scanned input images. To fully estimate the approximate centres of the halftone dots we have to know not only the distance between the dot—we have to know an offset vector too. This offset vector can be approximated from the argument of the Fourier spectrum in the points (u_1, v_1) and (u_1, v_1):

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \frac{-\arg F(u_1, v_1)}{2\pi} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} + \frac{-\arg F(u_2, v_2)}{2\pi} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \qquad \textbf{[EQ 5.11]}$$

No we can easily calculate the screen cell centres for the white cells:

$$\begin{pmatrix} x_{cw} \\ y_{cw} \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + k \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + l \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \qquad \textbf{[EQ 5.12]}$$

or the black cells:

$$\begin{pmatrix} x_{cb} \\ y_{cb} \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \left(k + \frac{1}{2}\right) \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \left(l + \frac{1}{2}\right) \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \qquad \textbf{[EQ 5.13]}$$

where k and l are integers. If the vectors $(x_1, y_1)^t$ and $(x_2, y_2)^t$ are at right angels and have the same length we can calculate the screen angle and cell size by:

$$\theta \approx \operatorname{atan}\frac{y_1}{x_1} \approx 90° + \operatorname{atan}\frac{y_2}{x_2} \qquad \textbf{[EQ 5.14]}$$

$$l \approx \sqrt{x_1^2 + y_1^2} \approx \sqrt{x_2^2 + y_2^2} \qquad \textbf{[EQ 5.15]}$$

But how do we extract the Fourier coordinates of the peaks from the spectrum? As we can see in figure 5.14, which is a 3D plot of the spectrum of a scanned authentic halftone image, the task will not be very hard. Except from the origin or points very close to the origin, the four base peaks are *much* higher than any other values in the spectrum. Extracting their positions will be quite easy.

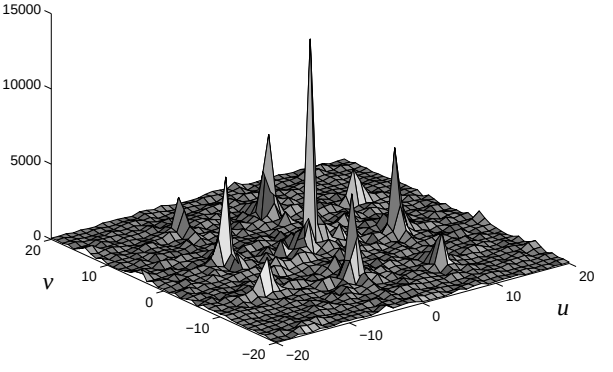


Figure 5.14 *Plot of the absolute value around the origin, of the Fourier transform of a one-colour halftone image (compare to figure 5.13).*

But the resolution of the DFT is quite coarse. There should be some way to interpolate a better coordinate for the peak than just an integer. This is done with help from the surrounding points. The method used is called *local gauss interpolation*. In both u and v directions we interpolate a more accurate peak coordinate, by fitting a gauss curve to the local neighbourhood. This is illustrated in figure 5.15.

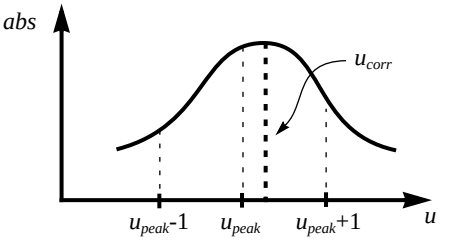


Figure 5.15 *Local gauss interpolation in one dimension. The integer peak value is corrected from the two adjacent values.*

The formulas to interpolate the new peaks is:

$$\begin{pmatrix} u_{corr} \\ v_{corr} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \dfrac{q_{u2} - q_{u1}}{q_{u2} + q_{u1}} \\ \dfrac{q_{v2} - q_{v1}}{q_{v2} + q_{v1}} \end{pmatrix} \qquad \textbf{[EQ 5.16]}$$

where

$$q_{u1} = \ln\left(\frac{abs(F(u,v))}{abs(F(u+1,v))}\right) \quad \textbf{[EQ 5.17]}$$

$$q_{u2} = \ln\left(\frac{abs(F(u,v))}{abs(F(u-1,v))}\right) \quad \textbf{[EQ 5.18]}$$

$$q_{v1} = \ln\left(\frac{abs(F(u,v))}{abs(F(u,v+1))}\right) \quad \textbf{[EQ 5.19]}$$

$$q_{v2} = \ln\left(\frac{abs(F(u,v))}{abs(F(u,v-1))}\right) \quad \textbf{[EQ 5.20]}$$

for $u = u_{peak}$ and $v = v_{peak}$.

The correctness of this estimation is analysed in more thoroughly in [Joh96], and is here illustrated in figure 5.16.

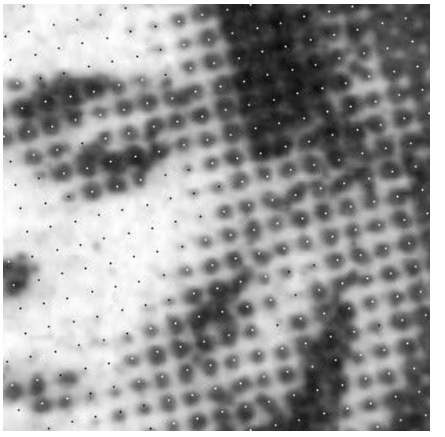


Figure 5.16 *The estimated screen cell centres marked by points on a scanned halftone image.*

Another way to increase the accuracy of the coordinates is to use higher order peaks than the base peaks, and then divide the estimated coordinates with their multiplicity. But since printing and scanning has a low pass filtering effect on the halftone image, the higher order peaks generally will be very suppressed and thus very uncertain. Also, when working with colour images, the peak pattern gets so complex (see figure 5.11), that everything above second order peaks is very uncertain.

Chapter

6. MODELLING OF THE DOT GAIN

This chapter is a brief compilation of the results produced at the Image Processing Laboratory, Linköping University, by Mikael Wedin [Wed95], Stefan Gustavson [Gust95], and Björn Kruse. One method each for estimating the mechanical and optical dot gain, respectively, is suggested: the transition area method and the colour shift method. The transition area method is essentially a colour classification method (compare to the methods described in the next chapter).

6.1 WHAT IS DOT GAIN?

When printing halftone dots, they hardly ever come out the same size as they had on the original film, that was used to create the printing plate. Usually the dots grow larger than on the original printing film, which results in a too dark image. Therefore, when making the film, this is (hopefully) taken into consideration.

The change of size of the halftone dots is not constant. It varies depending on ink quality, paper type, printing technique, and viewing conditions. This is a problem, and the distortion of the halftone dot's shape is referred to as *dot gain*. There are two main types of dot gain:

- *Mechanical dot gain*, which is the physical enlargement of the halftone dot.
- *Optical dot gain*, which is a visual enlargement of the halftone dot. Partly due to interaction between the incoming light and the paper substratum, in the vicinity of the dot—and partly due to the limitations of the eye's frequency response.

Traditionally the optical dot gain is ignored, or not separated from the mechanical dot gain. But since the two different types of dot gain have such different sources, the model may be improved by separating the two concepts.

6.2 MEASURING THE DOT AREA

When measuring dot gain the traditional way, the area actually covered with ink has to be measured. For this purpose, usually either a *planimeter* or a *densitometer* is used. With these methods, most of the smooth transition area surrounding the printed dot (see figure 6.1), is classified as ink covered. The reported ink covered area is thus a bit too large. This is why the transition area method, briefly described in section 6.4, is a better choice.

As the available scanners become better and better, the expensive planimeter may be exchanged for a high resolution scanner. Commercial scanners unfortunately (in this case) often include a compensation filtering step. The image is probably filtered automatically, which is bad when a raw data image is desired.

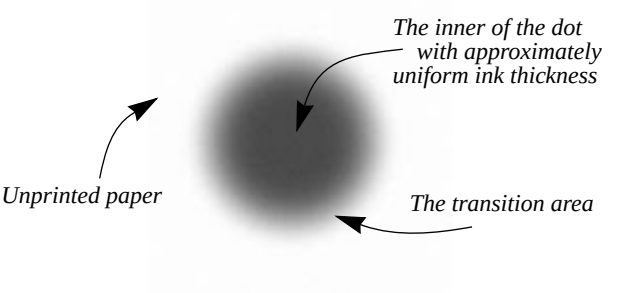


Figure 6.1 *A printed halftone dot with its surrounding shadow, the transition area, that yields a "darker" print (higher density).*

6.3 COLOUR DISTRIBUTIONS

If we do not consider the transition areas surrounding the halftone dots, there are only four classes of colours, when a halftone image is examined closely.

- The colour of the paper on which the halftone image is printed. Presumably some kind of white.
- The primary printing colours—usually cyan, magenta, yellow, and black—as they look when printed at this particular paper type. They are here referred to as *type I* colours.
- The secondary colours of two overlapping type *I* colours. With primary colours as above they will be red, green, blue, and black[1]. They are here referred to as *type II* colours.
- The overlapping of three or four different type *I* colours. They are all black. They are here referred to as *type III* colours.

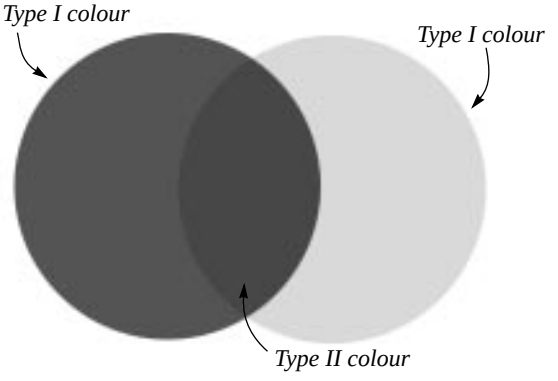


Figure 6.2 *The transition between two type I colours, goes via a type II colour.*

If two halftone dots are printed overlapping each other a type *II* is produced. There will be a border between each of the two type *I* colours and the type *II* colour, but no border between the two type *I* colours. In fact: there can be no colourimetric transition area between any two type *I*'s, or any two type *II*'s, according to the model proposed by Wedin. The transition between two primary colours always goes via a secondary colour. This is illustrated in figure 6.2.

In three colour printing we have eight clusters, as shown in figure 6.3, with twelve connecting lines, symbolizing the possible transitions between the clusters. Each of these transitions is approximately one-dimensional. This approximation can reduce the problem complexity a lot—instead of a three-dimensional problem, we can have twelve one-dimensional problems to deal with. It is definitely easier to decide the one-dimensional threshold values than the three-dimensional ones.

1. The fact that black can be both a type *I*, *II*, and *III* colour, is due to that there are nine kinds of black, as stated earlier.

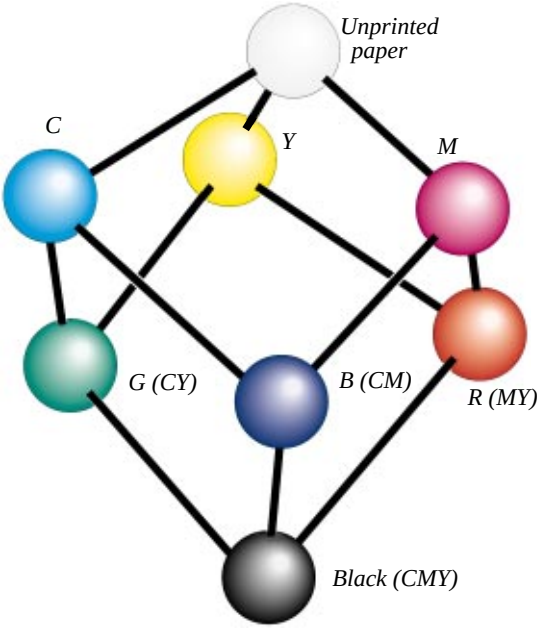


Figure 6.3 *The eight colour clusters (in three colour printing), and the twelve possible transitions between them.*

The necessary steps to convert the three-dimensional problem to the one-dimensional ones are:

- Calculate the centre of gravity, for each of the eight clusters in the 3D histogram. To do this we have to find the averages of the most frequent colour's coordinates in some colour space (see chapter 7, "Colour Classification" for explanation of colour space). Preferably, the CIELAB colour space is used.
- For each pixel: project the colour coordinate, assigned to the pixel, orthogonally to the colourimetrically closest, of the vectors between the centres of gravity.
- Colours close to a centre of gravity is projected on all three of the closest vectors.
- The one-dimensional colour distributions are the histograms over the projections on each vector. For simplicity, the lengths of the vectors between the clusters are normalized to one, prior to the calculation of the histograms.

Comment on this Colour Model

Since almost all colour printing is made in CMYK, and not in CMY, the discussion above is a grave simplification of the reality. Also, since the target input images in this thesis are printed in a newspaper or similar, there is a great deal of noise added. The nice cubic shape from figure 6.3 is then quite distorted.

6.4 ESTIMATION OF THE MECHANICAL DOT GAIN

The transition area function, $T(d)$, for a specific printing setup, gives the probable number of pixels, belonging to the transition areas of the halftone dots (see figure 6.1). T is a function depending on the tone, or gray level in the monochromic case, here denoted d. Let $C_d(x)$ be the one-dimensional colour distribution (histogram), for the transition in point. Let:

$$A(x_0, x_1) = \int_{x_0}^{x_1} C_d(x)(x)dx \qquad \textbf{[EQ 6.1]}$$

and then maximize:

$$\max_{x_1 - x_0} A(x_0, x_1) \leq T(d) \qquad \textbf{[EQ 6.2]}$$

Thus, we find the largest possible interval $[x_0, x_1]$ in the distribution between the clusters, holding the expected number of transition pixels—we locate the pixels where the tone variation is largest. The pixels yielding values between x_0 and x_1 in the colour distribution, is said to belong to the transition area. The actual edge is assumed to be located in the middle of this interval:

$$x_{edge} = \frac{x_1 - x_0}{2} \qquad \textbf{[EQ 6.3]}$$

Then, pixels yielding values in the distribution less than x_{edge} belong to the halftone dot, while those yielding larger values do not. The mechanical dot gain can be determined as:

$$dot\ gain = \frac{A_{gain}}{A_{desired}} = \frac{\int_{-\infty}^{x_{edge}} C_d(x)dx - A_{desired}}{A_{desired}} \qquad \textbf{[EQ 6.4]}$$

where $A_{desired}$ is the number of pixels belonging to halftone dots prior to the printing.

I do not understand how the author of [Wed95] decides the proper transition area function, to use in the calculations above. Therefore I refer to that report, in this matter. The same applies to the question of whether the transition area method can be used on images, other than test images with a single tone level.

6.5 ESTIMATION OF OPTICAL DOT GAIN

A large portion of the light reaching the paper surface, is not directly reflected. Instead it penetrates the paper and is scattered (see figure 6.4). Then it emerges from the top and bottom surfaces of the paper. The light scattering can be modelled by *point spread functions*.

The estimation of the optical dot gain, based on colour shifts, is far too complex to be described here.

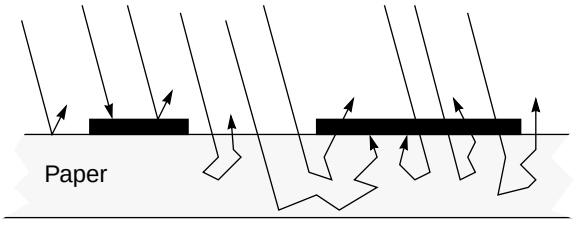


Figure 6.4 *Some possible paths for the photons. Some are directly reflected. Others are scattered in the paper. Some of the light is absorbed by the ink.*

6.6 THE USE OF THE DOT GAIN THEORY IN THIS THESIS

Since the target input images in this project are printed in four colour printing, and are quite noisy, there is a great difficulty in the reseparation step. Because of this, and shortness of time, it was hard to adopt any of the dot gain theories within the scope of this thesis. I have chosen to although include this chapter to point out the importance of the dot gain concept, and to inspire the use of this knowledge in future works on inverse halftoning.

Chapter

7. COLOUR CLASSIFICATION

This chapter describes a few methods for classification of the 16 possible colours in colour halftones. The methods are: global thresholding, local thresholdning, and learning vector quantization. None of the proposed methods were capable of distinguishing the nine possible black colours. This is why the DGP-based methods were not used in the method proposed in chapter 8.

7.1 COLOUR SPACES

When discussing colours we have to relate to one of several colour spaces. Usually one out of four common spaces is used:

- *RGB*—colourimetric tristimulus values with a tight connection to the physical aspects of colour and light.
- *XYZ*—colourimetric tristimulus values derived from *RGB*.
- *CIE 1976 ($L^*a^*b^*$ or $L^*u^*v^*$)*—device independent colour spaces created for good perceptual uniformity. That is, colours at equal distance in any direction in the coordinate system, will be percepted as equally different by the human eye.
- *CMYK*—device dependent format which is used in colour printers and the printing industry. The formats above are converted in a device dependent way to tones of the primary colours cyan, magenta, yellow, and black.

When the image is printed it has been halftoned from a *CMYK* image, but when the printed image is scanned, the scanner will most likely produce an *RGB* image. Apart from the fact that the conversion between these spaces can be done in several ways, the printing and scanning processes have most likely introduced errors. This is why the reseparation process is so difficult.

Let us start by looking a little closer on the four colour spaces mentioned above. For even more information, please refer to [Hunt91], or [Poyn95].

RGB and $\tilde{R}\tilde{G}\tilde{B}$

A fundamental fact in colour theory is that all visible colours can be expressed as linear combinations of three basis colours. Red, green, and blue are often practically chosen as basis colours. To determine the blending proportions of these three colours for all visible colours, an experiment was made (see figure 7.1). The CIE[1] Standard Observer is the average human observer, of the population having normal colour vision. The experiment was performed by letting the observer blend the red, green, and blue light, in order to achieve the same colour sensation as a reference light of each particular wavelength. The result is shown in figure 7.2.

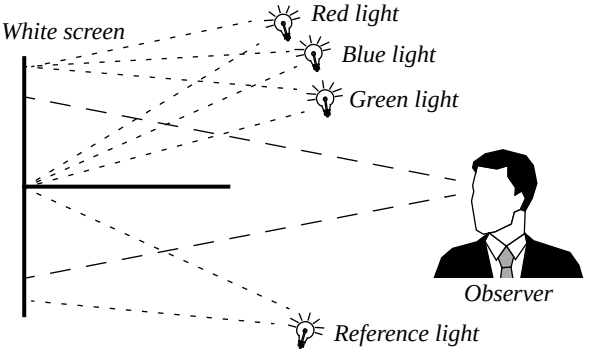


Figure 7.1 *Simplified illustration of the experimental setup, when deciding the colour matching functions.*

1. CIE is short for *Commission Internationale de l'Éclairage*.

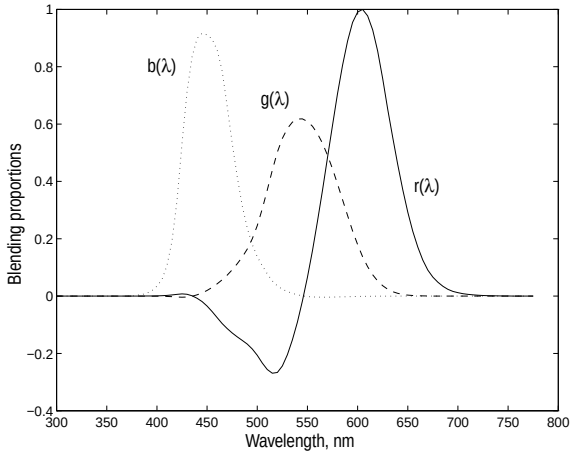


Figure 7.2 *The rgb colour matching functions.*

For some wavelengths some of the basis colours have negative coefficients. This strange phenomenon arises when it is not possible to achieve the reference colour with the red, green, and blue light. Then, one or several of the lights are moved over to the reference side, and thereby the same colour sensation can achieved on both sides. The wavelengths of the red, green, and blue lights used in this experiment are very well defined: red is $700nm$, green $546.1nm$, and blue is $435.8nm$.

By using the colour matching functions, it is possible to achieve so called tristimulus values R, G, and B for a certain coloured object:

$$R = \int_{\lambda} I(\lambda)r(\lambda)d\lambda \qquad \textbf{[EQ 7.1]}$$

$$G = \int_{\lambda} I(\lambda)g(\lambda)d\lambda \qquad \textbf{[EQ 7.2]}$$

$$B = \int_{\lambda} I(\lambda)b(\lambda)d\lambda \qquad \textbf{[EQ 7.3]}$$

where $I(\lambda) = F(\lambda)P(\lambda)$. P is the photon distribution illuminating the object, and F is the objects influence on incoming light (reflectance function). The percepted colour of an object is dependent on the light source illuminating the object—and two objects with different F's may be percepted as having the same colour when viewed under different light sources. This phenomenon is called *metamerism*. Therefore, it is important to specify the used illuminating light source, when comparing colours.

When scanning an image the *RGB* values received will not be depending on the Standard Observer's colour matching functions—they will be depending on the filter responses of the three filters, separating the colour components in the scanner. Let us call these functions $\tilde{r}(\lambda)$, $\tilde{g}(\lambda)$, and $\tilde{b}(\lambda)$. When the tristimulus values $\tilde{R}$, $\tilde{G}$, and $\tilde{B}$ are calculated in the same manner as above, these will most likely differ from R, G, and B. This implies that different transformations are required when converting the two different *RGB* formats to other colour spaces.

In computer graphics it is usually the $\tilde{R}\tilde{G}\tilde{B}$ values that are referred to, when using the term *RGB mode*.

XYZ

The unpleasant property of $r(\lambda)$, as an example, to be negative for some λ's creates a demand for other colour matching functions. They are standardized to:

$$\begin{pmatrix} x(\lambda) \\ y(\lambda) \\ z(\lambda) \end{pmatrix} = \begin{pmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0 & 0.01 & 0.99 \end{pmatrix} \begin{pmatrix} r(\lambda) \\ g(\lambda) \\ b(\lambda) \end{pmatrix} \qquad \textbf{[EQ 7.4]}$$

These transformed functions can be seen in figure 7.3. The transformation matrix may seem strange—why more decimals in the middle row? The answer is that the $y(\lambda)$ component is a sort of luminance or lightness indicator. Since the human eye is extra sensitive to differences in lightness, it is more important to have extra precision in this component.

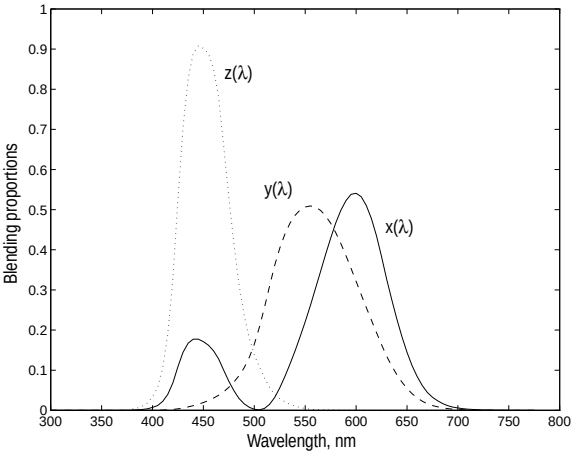


Figure 7.3 *The xyz colour matching functions.*

Now the tristimulus values *X*, *Y*, and *Z*, can be calculated in the same manner as *R*, *G*, and *B*:

$$X = \int_{\lambda} I(\lambda)x(\lambda)d\lambda \qquad \textbf{[EQ 7.5]}$$

$$Y = \int_{\lambda} I(\lambda)y(\lambda)d\lambda \qquad \textbf{[EQ 7.6]}$$

$$Z = \int_{\lambda} I(\lambda)z(\lambda)d\lambda \qquad \textbf{[EQ 7.7]}$$

The same transformation matrix may be used for the tristimulus values as for the colour matching functions. When transforming $\tilde{R}\tilde{G}\tilde{B}$ values to *XYZ* values the transformation matrix will be slightly modified (see equation 7.10, suggested in [Poyn95]).

Since it is custom in Matlab to give *RGB* values in the range of zero to one and *not* zero to one hundred (which is assumed in most literature), it is convenient to multiply the transformation matrix (and divide the inverse matrix) by a factor 100. Otherwise the standard values of X_n, Y_n, and Z_n, discussed in the next section have to be modified (divided by 100).

After applying the modification factor, described above, the transformation matrices are:

$$A_{RGB} = \begin{pmatrix} 49 & 31 & 20 \\ 17.697 & 81.240 & 1.063 \\ 0 & 1 & 99 \end{pmatrix} \quad \textbf{[Eq 7.8]}$$

$$A^{-1}_{RGB} = \begin{pmatrix} 23.646 & -8.9654 & -4.6807 \\ -5.1517 & 14.264 & 0.88758 \\ 0.05204 & -0.14408 & 10.092 \end{pmatrix} \cdot 10^{-3} \quad \textbf{[Eq 7.9]}$$

$$A_{\tilde{R}\tilde{G}\tilde{B}} = \begin{pmatrix} 41.2453 & 35.7580 & 18.0423 \\ 21.2671 & 71.5160 & 7.2169 \\ 1.9334 & 11.9193 & 95.0227 \end{pmatrix} \quad \textbf{[Eq 7.10]}$$

$$A^{-1}_{\tilde{R}\tilde{G}\tilde{B}} = \begin{pmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{pmatrix} \cdot 10^{-2} \quad \textbf{[Eq 7.11]}$$

where *A* is the transformation from *RGB* values to *XYZ* values, and *A inverse* is the opposite transformation.

CIELAB

The CIE 1976 ($L^*a^*b^*$) colour space, usually written CIELAB, is derived from *XYZ* coordinates with aim on perceptual uniformity. Originally it was developed to give the textile industry an accurate way to describe colours. Now it serves as one of the most well known device independent colour spaces, for all kinds of applications.

The transformation between *XYZ* values and CIELAB values is defined by:

$$L^* = \begin{cases} 116 \cdot \left(\frac{Y}{Y_n}\right)^{1/3} - 16, & \left(\frac{Y}{Y_n}\right) > 0.008856 \\ 903.3 \cdot \left(\frac{Y}{Y_n}\right), & \left(\frac{Y}{Y_n}\right) \le 0.008856 \end{cases} \quad \textbf{[Eq 7.12]}$$

$$a^* = 500 \cdot \left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right) \quad \textbf{[Eq 7.13]}$$

$$b^* = 200 \cdot \left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right) \quad \textbf{[Eq 7.14]}$$

$$f(x) = \begin{cases} x^{1/3}, & x > 0.008856 \\ 7.787x + \frac{16}{116}, & x \le 0.008856 \end{cases} \quad \textbf{[Eq 7.15]}$$

The constants X_n, Y_n, and Z_n are the *XYZ* values for the chosen reference white point. When working with colour monitors good choices could be something close to $(X_n, Y_n, Z_n) = (95.04, 100, 108.89)$, according to [Hunt91] and [Poyn95]. That is the reference white point for D65 CIE Standard Illuminant[2]. The equations above are valid for:

$$0 \le X \le X_n; \quad 0 \le Y \le Y_n; \quad 0 \le Z \le Z_n \quad \textbf{[Eq 7.16]}$$

Because of this there is a connection between the values of X_n, Y_n, Z_n, and the matrices in equations 7.8 and 7.10. They should be chosen as the row sums. The chosen values are the row sums of the matrix in 7.10.

How shall the coordinates in CIELAB be interpreted? Figure figure 7.4 gives a hint. L^* is the lightness, while a^* is (approximately) corresponding to the greenness/redness and b^* is (also approximately) the yellowness/blueness.

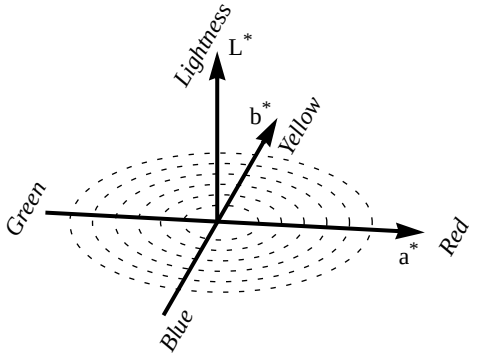


Figure 7.4 *The CIELAB colour space.*

In addition to $L^*a^*b^*$ coordinates there is also another device independent colour space called $L^*u^*v^*$ coordinates, defined by slightly modified equations. CIELAB works best in subtractive colour applications such as paper printing, while CIELUV works better for additive applications such as computer monitors and TVs.

CMYK

The conversion between *RGB* mode and *CMYK* is described in section 2.5. It is device dependent, and subtractive colour space.

2. One of several well defined standard illumination sets.

7.2 GLOBAL THRESHOLDING

The easiest approach to separate the four printing colours in a scanned image, is probably global thresholding. This implies studying of different histograms of the image. The target is to distinguish as many as possible of the 16 possible colours. In an ideal case it will be possible to detect them as clusters in the histograms, illustrated in figure 7.5.

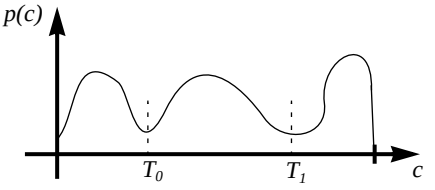


Figure 7.5 *In ideal histograms clusters of more frequent intensities can be spotted.*

It is quite easy, in the ideal case, to guess appropriate threshold values, and there are several statistical methods to improve these guessed values. But for real halftone images this is not the case. It is very rare to see any particular clusters at all (see figure 7.6)—whatever colour space is chosen.

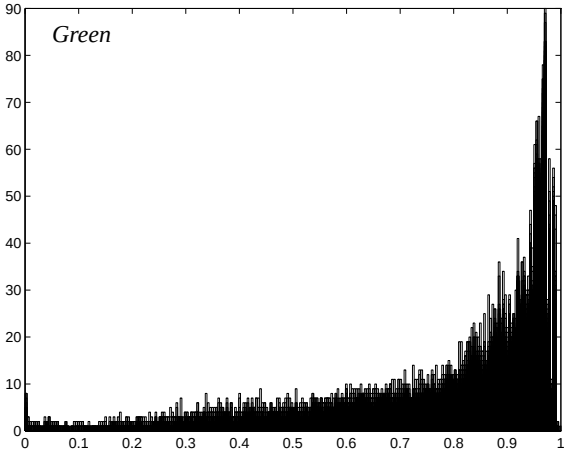


Figure 7.6 *Histogram from the green colour channels in a light RGB image. It is a non-trivial task to decide the best threshold values.*

The situation does not improve very much by switching to a device independent colour space, such as CIELAB. It is possible to distinguish between all colours that are not black, with pretty good accuracy, but impossible to tell one black from another. And it is very hard to automize the thresholding process, because of the fact that the histograms do not have the desired properties.

The ability to detect different kinds of black is very low. The global thresholding only works properly for three colour prints, such as in figure 7.7.

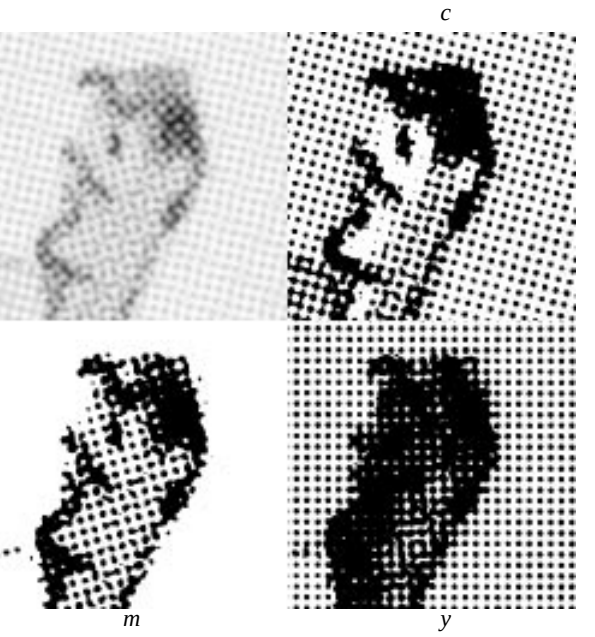


Figure 7.7 *A scanned, three colour printed[3], halftone image (uppermost, left), and globally thresholded estimates of the three printing colours.*

7.3 LOCAL THRESHOLDING

To compensate for shifts of illumination in the scanner, and colour shifts, local thresholding may be applied. Originally this would require the generation of one histogram for the neighbourhood of each pixel. This is much to computationally heavy. Instead of a fixed threshold, the pixel values are compared to an adaptive threshold, dependent of the local average tone:

$$g(x,y) = \begin{cases} 1 & f(x,y) \geq \bar{f}(x,y) + T \\ 0 & otherwise \end{cases} \qquad \textbf{[EQ 7.17]}$$

which is equivalent to:

$$g(x,y) = \begin{cases} 1 & f(x,y) - \bar{f}(x,y) \geq T \\ 0 & otherwise \end{cases} \qquad \textbf{[EQ 7.18]}$$

The signal $f(x,y) - \bar{f}(x,y)$ is achieved by filtering the image with a Laplace operator, exemplified in figure 7.8, of appropriate size. What "appropriate" means in this in this case, is depending on the image characteristics.

3. The image is actually four colour printed, but *K* is used in very few spots.

-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	24	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

Figure 7.8 *Laplace operator of size 5x5.*

This method is quite hard to apply to the halftone image. Since areas can be completely filled with ink, or completely white, the Laplace operator would have to be quite large (the size of several screen cells) to get accurate results. The computation to filter the image will be rather heavy. Even then, there will be problems with large completely filled or white areas. The local thresholding method is based upon the assumption that there are approximately the same proportions between printed and unprinted areas, throughout the image. This is rarely true for halftone images (illustrated in figure 7.9).

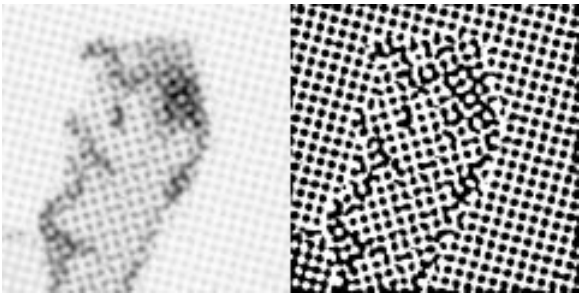


Figure 7.9 *To the left: the green channel of a scanned image, containing primarily the magenta screen. To the right: local thresholding of the image. In solid non-printed areas, the cyan screen is gained (compare to figure 7.7m).*

The summary is that local thresholding will be very time consuming, and maybe produce inaccurate results. Thus it is not very applicable to the existing problem.

7.4 LEARNING VECTOR QUANTIZATION

This next colour classification method is collected from Matlab's Neural Network Toolbox. It is called *Learning vector quantization* (LVQ), and is a method for training competitive layers in a supervised manner. The competitive layers will automatically learn how to classify input vectors.

The resulting classification can be quite good depending on the choice of training vectors and the distance between the vector clusters representing each colour in the chosen colour space. However, it is very difficult choosing training vectors for all the nine blacks. Practically it can not be done, other in special test images. Accordingly, the LVQ will not be able to distinguish them. A negative aspect of the LVQ estimates, compared to the thresholded estimates, is that the shape of the halftone dots have rough edges. This can be seen in figure 7.10.

From some experiments (see figure 7.10), the conclusion was drawn, that CIELAB was better suited for the LVQ, than the *RGB* colour space. The choice of training pixels also affects the estimate quality, so perhaps this comparison should be examined more carefully.

Figure 7.10 *LVQ estimation of the cyan screen from image in figure 7.7. To the left: estimation in RGB. To the right: estimation in CIELAB. Compare these images to figure 7.7c.*

7.5 WHY THE INTEREST IN NINE BLACKS?

Why is it so important to distinguish between nine black colour combinations, that almost entirely look the same? If the eye can not tell the difference, why should we bother to get the computer to do it? A simple solution would be to say that wherever K is present, none of the other three colours are. Or another solution: wherever K is present, then so are all of the other three colours. Both of these solutions would make the black screen pattern present in all the other colour channels. There will be two screens in each channel, and therefore we will still not be able to apply the methods described in chapter 4, "DGP-based Inverse Halftoning".

I do not think that it ever will be possible to distinguish all nine blacks, but perhaps they could at least be separated into four "black classes". Then perhaps the DGP-based methods would be applicable.

7.6 EFFECTS WHEN INCREASING THE COLOUR DEPTH

The scanner that was used during this project had the feature to optionally scan in 48 bit *RGB* (16 per colour channel) instead of the traditional 24 bit *RGB* (8 per channel). What benefit would this give. when trying to reseparate the primary colours from the scanned halftone image? Unfortunately the answer is almost *none*. The presence of noise in the printed newspaper image is larger than the smallest colour unit in 24 bit mode. That is, the new 24 bits will not provide much more valuable information at all.

Chapter

8. THE PROPOSED METHOD

The method I propose when performing inverse halftoning of colour images, is here described step by step. In section 8.5, an example is provided. This example is intended to illustrate the abstract description from the first four sections. Some readers may find it easier to read section 8.5, before reading the other sections.

In chapter 5, "Halftones in the Fourier Domain", it was concluded that if the screen parameters were accurately estimated, then a flat, square, spatial filtering kernel, of the right size and rotation, was a good choice for inverse halftoning of monochrome images. But to remove more than one screen pattern, as in the case of colour halftones, we would have to apply several filter kernels with different rotations (and possibly sizes). Frequency components between the Fourier domain peaks will by this iteration be suppressed several times, which is not good. The moire would force the use of spatially larger kernels, since it adds low frequency peaks to the spectrum. This can be avoided by following the steps described in the next few sections.

8.1 STEP 1: ESTIMATING PARAMETERS

The first step is to estimate the screen angles and frequencies of the different screen patterns in the scanned halftone image. The scanner delivers an *RGB* image, which has three colour channels. If the black printing colour (*K*) is disregarded, the following is known:

- In the *R* channel, the cyan printing colour will be dominant, since red is not additive part of cyan in *RGB*, while green and blue are. Approximately $C \approx 1 - R$.
- In the *G* channel, the magenta printing colour will be dominant for the same reason as above: $M \approx 1 - G$.
- In the *B* channel, the yellow printing colour will be dominant: $Y \approx 1 - B$.

K will be more or less present in all three of the channels. Either it, or *C*, *M*, and *Y*, respectively (according to the statements above), will be dominant in the three *RGB* channels. Then the actually dominant screen is decided for each channel, by searching for the highest peaks in the Fourier spectrum (besides origo). From the coordinates of these peaks, the screen frequencies and angles are calculated. The calculation is performed according to the algorithms described in section 5.5. We completely disregard the other screens and the moire effects, and concentrate on removing the dominant pattern. The moire will not be dominant at this stage; the integer multiple peaks are always lower than the base peaks they descent from.

8.2 STEP 2: PRODUCING FILTER KERNELS

Now, for each colour channel *R*, *G*, and *B*, the screen angle and frequency, for the screen that is dominant in that particular channel, is captured. We now create flat, spatial convolution kernels for the three colour channels. If the lengths of the sides of the screen cell are close to integer, they can be produced in Matlab, in the following way:

```
> A=zeros(s+2,s+2);
> A(2:s+1,2:s+1)=ones(s,s);
> A=imrotate(A,ang,'bilinear');
> A=A/sum(sum(A));
```

where *s* is the (integer) side length of the estimated screen cell, and *ang* is the estimated screen angle. The produced kernel's behaviour in the Fourier domain is

illustrated in figure 8.1. In the same way as above, one kernel each, for the other two colour channels, is created. The convolution is applied by:

```
> r1=conv2(A,r);
```

where *r* is the red channel image. When the convolution is applied, the resulting image will be larger than the original. Therefore, *r1* is cropped to the same size as *r* by choosing a centred partition of *r1*, of the right size.

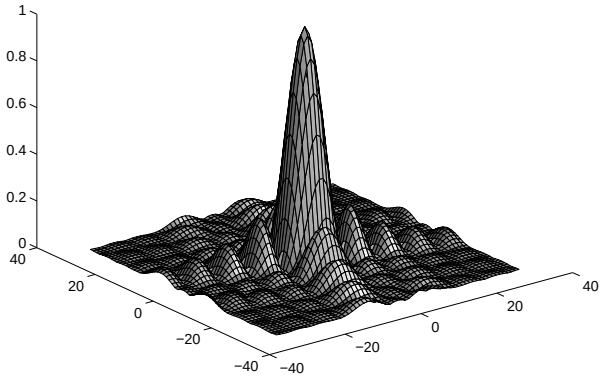


Figure 8.1 *Example of the Fourier transform of a spatial kernel, created as described above.*

If the three resulting images are viewed they will most probably show remains of the non-dominant screen patterns, and the moire effects. But even these peaks in the Fourier spectrum will most likely have been much reduced by this first filtering (compare figure 8.4 to figure 8.3). In fact they are reduced so much that the remaining peaks can be adjusted for, selectively—we do not need another low pass filtering.

8.3 STEP 3: ADJUSTING REMAINING PEAKS

To adjust the remaining peaks, we, for each colour channel, choose a representative part of the image, at the size of, for example, 256×256 pixels. Working with the whole image would be too computationally heavy. Then we Fourier transform the three partitions, and thereby get three Fourier spectra. By visualizing these we can decide which annoying peaks still remains. By studying the neighbourhoods of these peaks, we can determine how a Fourier domain filter should be designed to suppress them.

The image size is increased by adding zeros around the image, to a size where width as well as height are integer multiples of 256 (or another number, if a different size of reference area is chosen above). Then we partition the image into blocks of size 256×256, Fourier transform them, apply the filter as a multiplication in the Fourier domain, and perform inverse transformation of the result. Finally we remove the pixels that we added earlier (as zeros) by cropping the image. The reason why the channel images are Fourier transformed in partitions, rather than in one piece, is that transforming a large image requires enormous amounts of computer memory during the calculation. This is avoided by partitioning the image into smaller blocks. Since the same filter is applied to all blocks in each channel image, the borders between the blocks will not be visible in the output images.

When adding the three colour channels, we will get a relatively sharp continuous tone version of the input halftone image. The reconstruction is completed, and may, if wished, be compared to the original continuous tone image. Then calculations of the image degradation, due to halftoning and printing errors, can be made.

8.4 STEP 4: POST-FILTERING

If the purpose of the inverse halftoning is to rereproduce the image, it will most probably require some kind of post-filtering to reduce the printing noise. This filtering should be highly adaptive to the image characteristics, and knowledge about how the image is supposed to look. The most convenient environment to perform these operations in, is probably programs like Photoshop, or similar.

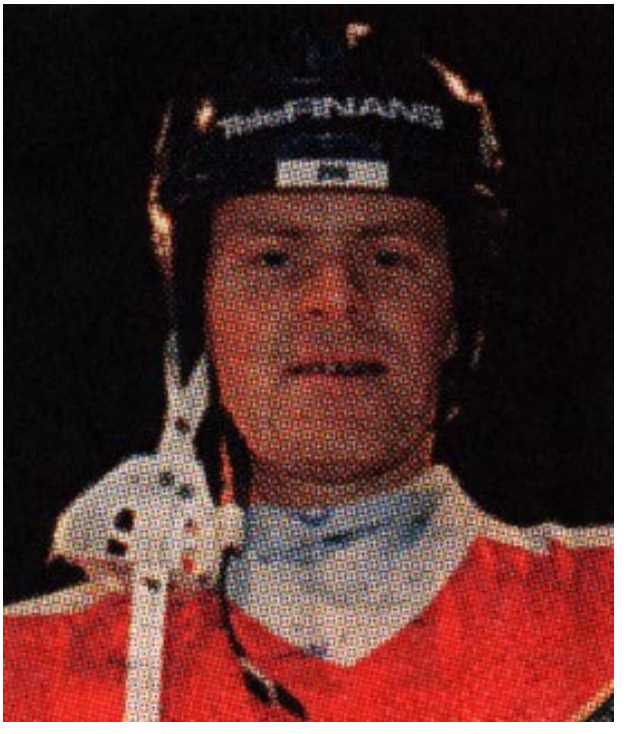


Figure 8.2 *The scanned example image. This image is a printed colour halftone image. When reproduced here it is halftoned a second time, which gives some additional moire.*

8.5 AN EXAMPLE

The easiest way to illustrate the method is to perform an actual image reconstruction on a printed image[1] (see figure 8.2). The image is scanned on an AGFA Arcus II scanner in 600 *dpi*, which is the optical resolution of the scanner, and 24 bits colour depth. For optimal reconstruction quality the image should be scanned at the highest available optical resolution. Interpolated resolution does not add any valuable information.

The dominant screen frequencies and angles for the three colour channels *R*, *G*, and *B*, are automatically determined with the Matlab function *peaks*[2] from a partition of the size 256×256. In figure 8.3, the *G* channel of this partition is shown. The partition should be chosen as a representative area, and it does not have to be the same area for the three channels.

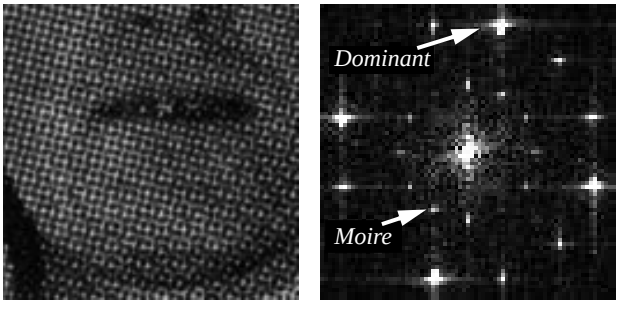


Figure 8.3 *Partition of the G channel with its Fourier transform. Note that in this chapter the plots from the Fourier domain are cropped to only show the most interesting parts of the spectrum around origo.*

The estimated values are:

- In the *R* channel the screen angle is 15.1°, and the cell width 7.54 (pixels).
- In the *G* channel the screen angle is 74.9°, and the cell width 7.54 (pixels).
- In the *B* channel the screen angle is 0.1°, and the cell width 7.99 (pixels).

Since we know that the input image was scanned in 600 *dpi* we can now calculate the screen frequency:

$$\frac{600}{7.54} = 79.58 \qquad \frac{600}{7.99} = 75.09 \qquad \textbf{[Eq 8.1]}$$

This means that the image is printed with cyan and magenta in about 80 *lpi*, while yellow is only printed in about 75 *lpi*. It is quite common to print yellow at a lower frequency, since the images has proven to be more pleasing to the eye this way. Since black was not dominant in any of the channels, we do not know what frequency was used for this colour[3].

Now three flat averaging kernels are constructed, with sizes and rotations as above. Each channel is spatially convolved with its corresponding kernel with the Matlab function *conv2*. A partition of the output image of the G channel is shown in figure 8.4. It is a good first result, but it still has small remains of the original screens.

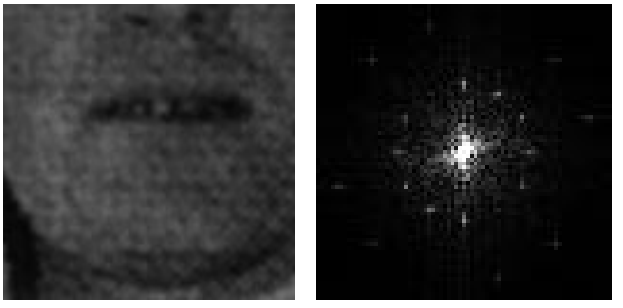


Figure 8.4 *The partition from figure 8.3 filtered with averaging kernel and its Fourier transform.*

From the Fourier plot in figure 8.4, we can see how effectively the dominant peaks have been removed from the spectrum in figure 8.3. But still there are small peaks left from the moire and the non-dominant base peaks. What is most important: the remaining peaks are almost all reduced to one sample point in the spectrum. These points can now be removed properly by a simple Fourier filter. The filter is designed in the following way:

- Again choose a representative area from the output images from the previous step, of for example 256×256 pixels.
- Calculate the Fourier spectrum for the chosen partition, in each of the three colour channels.
- Initiate the Fourier filter, by creating a matrix of the same size as the Fourier spectrum, and assign all positions in the matrix to 1.
- Study the amplitude spectrum in the neighbourhoods of the peaks, and decide how much they have to be suppressed to have about the same amplitude as the environment. Assign the position in the matrix corresponding to the peak coordinate an appropriate value between 0 and 1. This process has not yet been automized, but that could be done quite easily.
- Remember that for the matrix we must have $M(u, v) = M(-u, -v)$. Otherwise the output image will not be real—it will contain complex numbers.

1. I have (randomly) chosen an image from the local newspaper Östgöta Correspondenten. The image shows the local bandy player Roger Carlsson.
2. The function *peaks* is developed for this project and the code is available in appendix B.
3. Black is printed with the same screen frequency as cyan and magenta in this image. This is the standard solution.

An example of the resulting matrix is shown in figure 8.5. Three filter matrices are created—one for each colour channel.

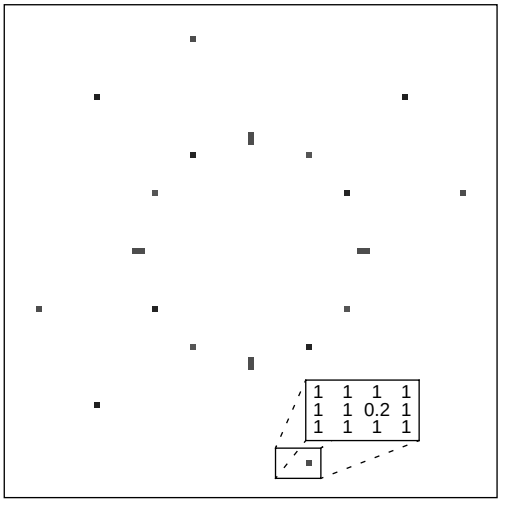


Figure 8.5 *Fourier filter to be multiplied with the spectrum in figure 8.4, in order to get rid of the remaining peaks.*

The image from step 2 is now partitioned into pieces of the same size as the filter matrices (in this case 256×256 pixels). If the image's width and height are not integer multiples of the filter size, zeros are added at the image borders, prior to the partitioning. The output image for each colour channel is received by applying the filter, to the Fourier transforms of all partitions. The whole channel image is filtered with the same filter. This is done in Matlab with the function *fftfilt*[4].

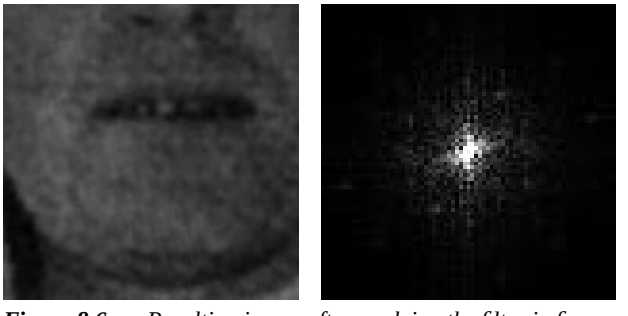


Figure 8.6 *Resulting image after applying the filter in figure 8.5. The output image now lacks almost all remains of the original screen patterns.*

4. Implemented for this project. Source code in appendix B.

The three channels are then added and saved as a standard 24 bit *RGB* image (see figure 8.7). It can then be post-filtered in Photoshop to get an even better output image. A comparison of the scanned image and the reconstruction can be found in appendix A.

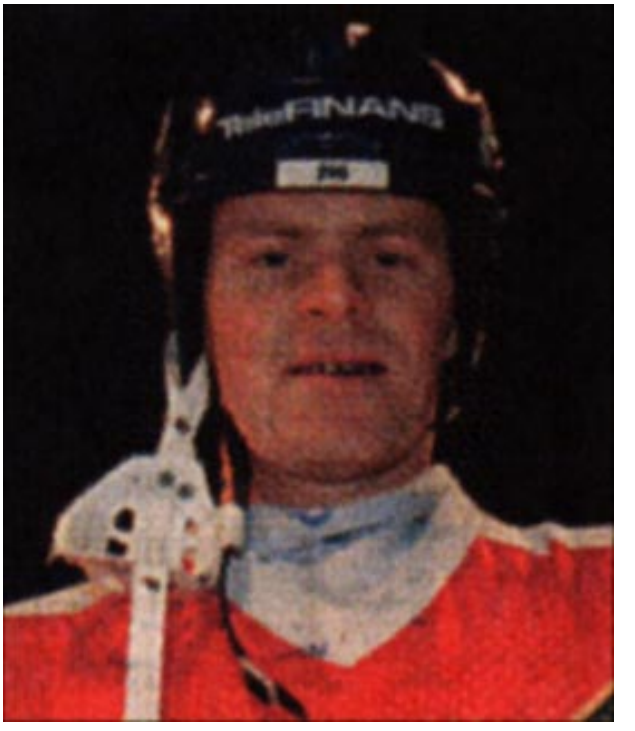


Figure 8.7 *The output image before any further post-filtering. This should be a continuous tone colour image, but it is here halftoned for reproduction.*

8.6 WHAT ARE THE BENEFITS OF THIS TECHNIQUE?

Today, to my knowledge, the only commercial available techniques to perform inverse halftoning of colour images is to use low pass filtering such as gaussian blur in Photoshop, or use scanner producers algorithms, such as AGFA's OpenLook descreening.

The proposed method will yield a sharper image than both of these methods and due to the adaptiveness of step 3, it is guaranteed that no trace of the original screens remains in the reconstructed image.

Chapter

9. CONCLUSIONS AND SUGGESTIONS

The original model from figure 1.1 on page 1 had to be adjusted because of the problem of performing some of the steps. A modified approach (figure 9.1) was taken and the result was pleasing.

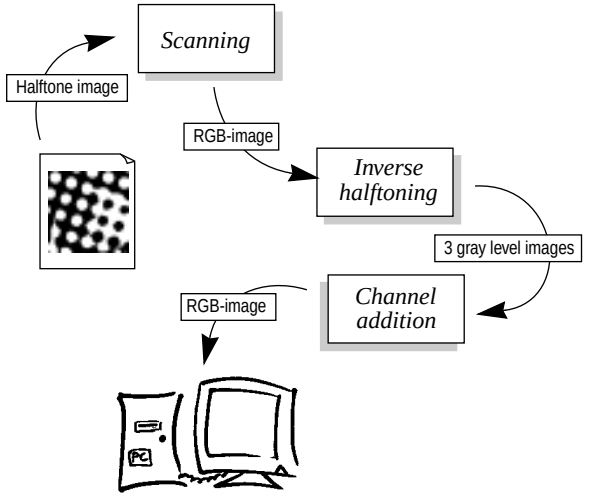


Figure 9.1 *The modified model for inverse halftoning.*

9.1 CONCLUSIONS

A method for inverse halftoning of colour halftone images has been proposed and implemented in Matlab. The resulting output images have no trace of the original screen pattern, and are percepted approximately as sharp as the halftone images by the eye.

If the printed image is scanned on a colour calibrated scanner it may be compared to the original scanned photography, to get a mathematical model of the printing and halftoning processes.

9.2 SUGGESTIONS FOR FURTHER WORK

I suggest that the method is fully automized and implemented as a plug-in filter to Photoshop. That may very well be a commercial product. The graphical industry now lacks a good way to perform inverse halftoning of already printed colour halftones. As colour scanners and printers become cheaper and cheaper, there will be a great need for a product such as this on the private market. The algorithms for inverse halftoning, that the scanner producers deliver today, are not good enough.

Further, I suggest that the method is tested as a tool for evaluating the halftoning and printing processes. In this way, the newspaper producers could achieve a better knowledge of the distortion their colour images will be exposed to. This way a better image quality in print could be achieved.

More effort should be made to separate the four printing colours in the printed image. In this way, the DGP based method for inverse halftoning could be used. That method produces sharper images than the linear filtering step in the my method, but is completely depending on the fact that only one screen pattern is present in each channel image.

Appendix

A. IMAGES

In this appendix some test images are reproduced in dye diffusion technique[1], yielding continuous tone images. The dye diffusion prints are reproduced on special printing films. Because of this, this appendix is only single sided.

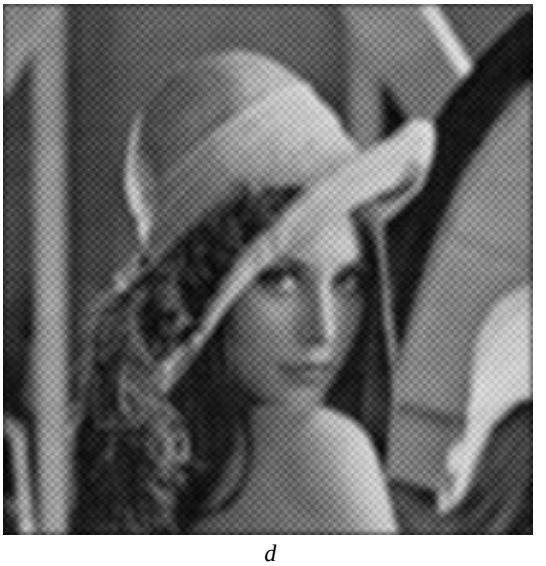
The used printer is a Seiko Professional ColorPoint II, capable of printing 300 *dpi* colour images with dye diffusion. The output quality is almost as good as photographic originals. The copies should not be directly exposed to sunlight.

1. Because it is so expensive printing in dye diffusion, not all copies of this thesis include the colour prints.

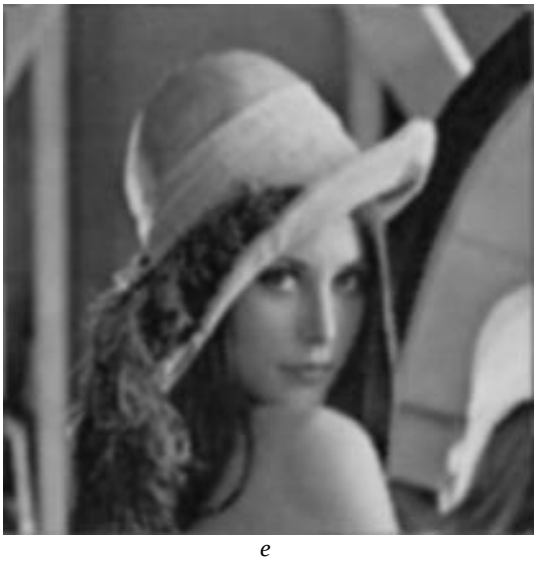
a

b

c



d



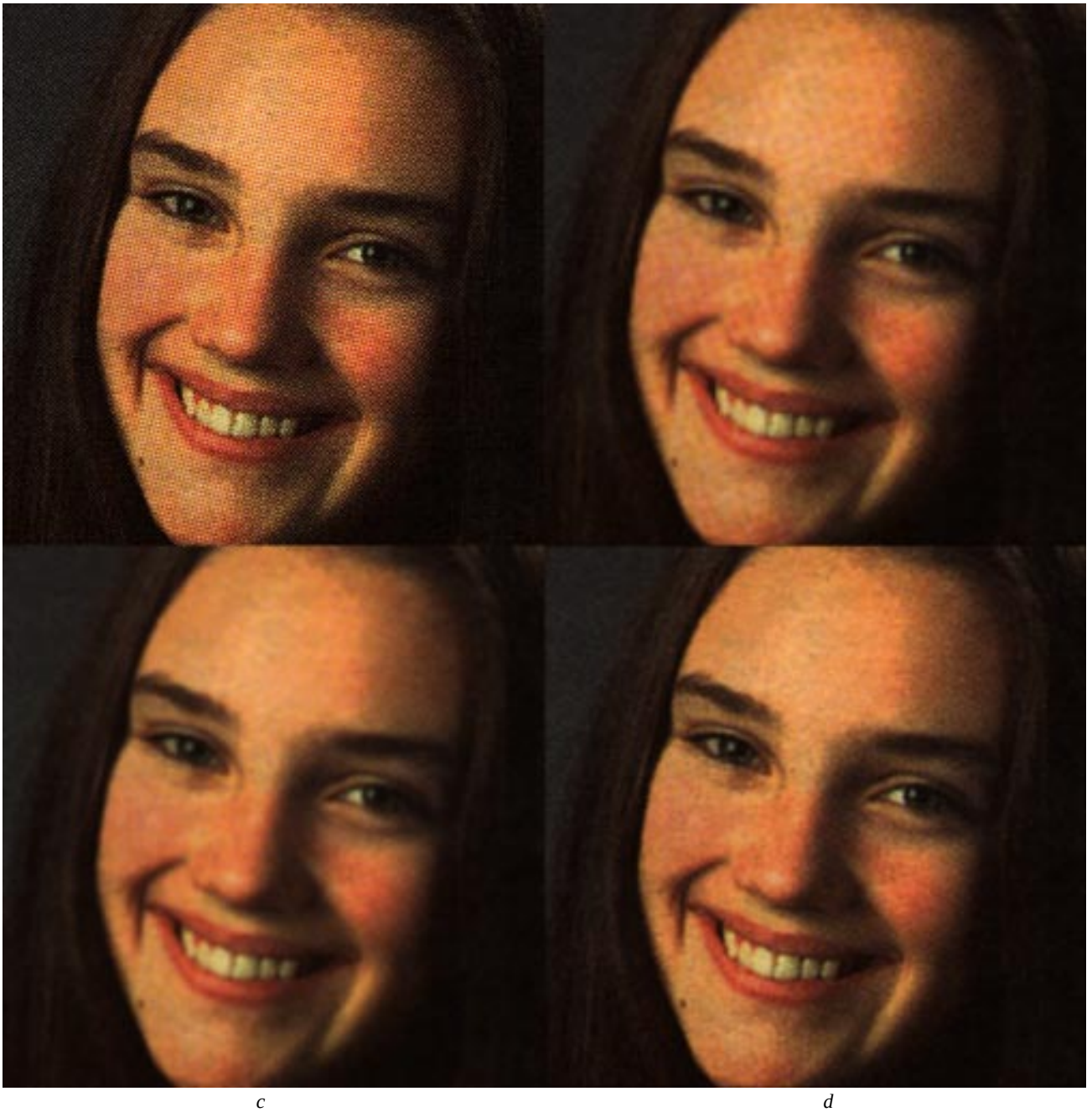
e

f

Figure A.1 *Images from chapter 5. (a) Original continuous tone image. (b) Binary halftone image, halftoned with threshold halftoning. (c) Reconstruction with a proper sinc function. (d) Reconstruction with a sinc function, with too low limit frequency. (e) Reconstruction with a gaussian low pass filter. (f) Reconstruction with a close to ideal low pass filter.*

Figure A.2 *Images from chapter 8. (uppermost) Scanned newspaper image. Here enlarged three times. (lowest) Continuous tone image, reconstructed as described in chapter 8.*

a *b*



c *d*

Figure A.3 *Comparing different methods for inverse halftoning. (a) Scanned from newspaper image. Here enlarged two times. (b) Reconstruction using AGFA's OpenLook descreening. (c) Optimal reconstruction using Photoshop's Gaussian Blur filter. (d) Reconstruction using the method proposed in chapter 8.*

Appendix

B. MATLAB CODE

This chapter contains some of the source code that was implemented in Matlab 4.2c during the project. Matlab Image Toolkit, Neural Network Toolkit, and Signal Toolkit was available during the implementations.

B.1 CONVERTING TOOLS

```
function [x,y,z] = rgb2xyz(r,g,b)
% RGB2XYZ Converting from RGB [0,1], to XYZ
%         tristimulus values.
[n,m] = size(r);
Q = [41.2453 35.7580 18.0423; ...
     21.2671 71.5160  7.2169; ...
      1.9334 11.9193 95.0227];
xyz = Q*[r(:)';g(:)';b(:)'];
x = reshape(xyz(1,:),n,m);
y = reshape(xyz(2,:),n,m);
z = reshape(xyz(3,:),n,m);
```

```
function [r,g,b] = xyz2rgb(x,y,z)
% XYZ2RGB Converting from XYZ, to RGB [0,1]
%         tristimulus values.
[n,m] = size(x);
Q = [0.03240479 -0.01537150 -0.00498535; ...
    -0.00969256  0.01875992  0.00041556; ...
     0.00055648 -0.00204043  0.01057311];
rgb = min(1,max(0,Q*[x(:)';y(:)';z(:)']));
r = reshape(rgb(1,:),n,m);
g = reshape(rgb(2,:),n,m);
b = reshape(rgb(3,:),n,m);
```

```
function [L,a,b]=xyz2lab(x,y,z)
% XYZ2LAB Converting from XYZ tristimulus
%         values, to CIELAB colour space.
Xn = 95.05;
Yn = 100.00;
Zn = 108.89;
ylim = Yn*0.008856;
L = (y>ylim).*(116*(y/Yn).^(1/3)-16) ...
    +(y<=ylim).*(903.3*y/Yn);
ytemp = foo(y/Yn)
a = 500*(foo(x/Xn)-ytemp);
b = 200*(ytemp-foo(z/Zn));
```

```
function [x,y,z]=lab2xyz(L,a,b)
% LAB2XYZ Converting CIELAB colour space,
%         to from XYZ tristimulus values.
Xn = 95.05;
Yn = 100.00;
Zn = 108.89;
y = Yn*((L<8).*L/903.3 ...
       +(L>=8).*((L+16)/116).^3);
ytemp = foo(y/Yn);
x = Xn*(a/500+ytemp).^3;
z = Zn*(ytemp-b/200).^3;
```

```
function y = foo(x)
% FOO Helping function used by XYZ2LAB
%     and LAB2XYZ.
y = (x<=0.008856).*(7.787*x+16/116) ...
    +(x>0.008856).*x.^(1/3);
```

B.2 PARAMETER ESTIMATION

```
function [A,l,fi]=peaks(F)
% PEAKS Calculates the dominant screen
%       vectors (A), the width of the screen
%       cell (l), and the screen angle (fi),
%       from the Fourier spectrum (F),
%       received from the fft2 function.

N = 16;
[m,n] = size(F);
G = abs(F);
[Y,I] = sort(G(:));
B = [];
j = 0;

for i = 1:N
  x = 0;
  y = 0;
  while (abs(x)<5 & abs(y)<5)
    x = floor((I(m*n-j)-1)/m)+1;
    y = I(m*n-j)-m*(x-1);
    temp = gaussint(y,x,G);
    x = temp(2);
    y = temp(1);
    if x<(n/2)
      x = x-1;
    else
      x = x-n-1;
    end
    if y<(m/2)
      y = 1-y;
    else
      y = 1+m-y;
    end
    j = j+1;
  end
  B = [B;[y,x]];
end

i = 1;
uv1 = [];
while (i<=N)
  if (B(i,1)<=0.1) & (B(i,2)>=-0.1)
    uv1 = [uv1;B(i,:)];
  end
  i = i+1;
end

uv2 = [];
j = 1;
Q = [];
while uv2==[] & j<=size(uv1,1)
  i = 1;
  while i<=N & uv2==[]
    if (B(i,1)>=-0.1) & (B(i,2)>=-0.1) & ...
        abs(sum(B(i,:).*uv1(j,:)))< ...
        0.01*sum([B(i,:),uv1(j,:)].^2) & ...
        abs(sum(B(i,:).^2)- ...
        sum(uv1(j,:).^2))< ...
        0.1*sum([B(i,:),uv1(j,:)].^2)
      uv2 = B(i,:);
      uv1 = uv1(j,:);
    end
    i = i+1;
  end
  j = j+1;
end

if size(uv1,1)==1 & size(uv2,1)==1
  d = uv1(2)*uv2(1)-uv1(1)*uv2(2);
  A = [-n*uv1(1) m*uv1(2); ...
       n*uv2(1) -m*uv2(2)]/d;
  l = (sqrt(A(1,1)^2+A(1,2)^2) ...
       +sqrt(A(2,1)^2+A(2,2)^2))/2;
  fi = atan2(A(1,2),A(1,1))*180/pi;
else
  disp('Unable to calculate the parametres.')
end
```

```
function A = gaussint(i,j,fftim)

% GAUSSINT Gaussian interpolation of
%          actual local maximum in fftim,
%          close to coordinate (i,j)

fi = abs(fftim(i,j));

if j==size(fftim,2)
  a = log(fi/abs(fftim(i,1)));
else
  a = log(fi/abs(fftim(i,j+1)));
end

if j==1
  b = log(fi/abs(fftim(i,size(fftim,2))));
else
  b = log(fi/abs(fftim(i,j-1)));
end

if i==size(fftim,1)
  c = log(fi/abs(fftim(1,j)));
else
  c = log(fi/abs(fftim(i+1,j)));
end

if i==1
  d = log(fi/abs(fftim(size(fftim,1),j)));
else
  d = log(fi/abs(fftim(i-1,j)));
end

A = [i+(d-c)/(2*d+2*c),j+(b-a)/(2*b+2*a)];
```

B.3 FILTERING FUNCTIONS

```
function B=fftfilt(A,F)
% FFTFILT Application of the Fourier
%         filter F, on the image A. The
%         width and height of A must be
%         integer multiples of the width
%         and height of F.
[m,n] = size(A);
[k,l] = size(F);
B = [];
for i = 0:(m/k-1)
  line = [];
  for j = 0:(n/l-1)
    line = [line,ifft2(fft2( ...
           A((1+i*k):((i+1)*k), ...
           (1+j*l):((j+1)*l))).*F)];
  end
  B = [B;line];
end
```

Appendix

C. ENGLISH-SWEDISH DICTIONARY

This is appendix is present to increase the understanding of this thesis, for the Swedish reader. Also, it is useful to have Swedish translations of some of the English terms, when discussing the topics of this thesis in Swedish.

B

Base peaks. Första ordningens toppar. De stora topparna i frekvensspektrumet som härrör direkt från rasterfrekvensen.

C

Clustered dot. Samlad rasterpunkt. Trösklingsfunktionen som rastrerar bilden är uppbyggd så att samtliga färgade punkter i varje rastercell hänger samman (gäller i lågfrekventa områden i bilden). Traditionellt använd i trycksaksindustrin.

Continuous tone image. Halvtonsbild (jfr. Halftone image). Inläst eller datorgenererad bild som inte är binär. Närliggande punkter har ofta närliggande färg- eller gråskalevärde – färgen varierar kontinuerligt.

D

Dispersed dot. Spridd eller sprängd rasterpunkt. ”Motsats” till samlad rasterpunkt (Clustered dot). Ofta använd vid rastrering för bildskärmar.

Dot gain. Punktförstoring. Mekanisk eller optisk förvanskning av rasterpunktens form och storlek.

Dots per inch (dpi). Måttenhet för upplösning på skärm eller papper. Antalet minsta byggstenar (punkter) som ryms på en tum.

H

Halftone image. Rastrerad bild eller rasterbild. En binär bild som efterliknar utseendet hos originalbilden. När rasterbilden har tryckts och läses in igen så är den inte längre binär – den har tillförts brus och lågpassfiltrerats.

Halftoning. Rastrering. Processen att framställa en rastrerad bild från en kontinuerlig originalbild – på svenska kallad halvtonsbild.

I

Inverse halftoning. Avrastrering eller derastrering. Process för att avlägsna rastermönstret och ta fram en bild som i någon mening liknar halvtonsbilden som var original till den rastrerade bilden.

L

Lines per inch (lpi). Mått på rastertäthet eller rasterfrekvens. En dagstidning trycker normalt med omkring 85 lpi medan en dyrare tidskrift med bättre papperskvalitet använder mellan 120 och 200 lpi.

Look-up table halftoning (LUT halftoning). Tabellrastrering. Rastreringsmetod där man använder sig av ett förutbestämt alfabet (en tabell) av byggstenar för att skapa sin rasterbild.

M

Misregistration. Misspass eller registerfel. De olika tryckplåtarna är ej korrekt centrerade över varandra. Dagstidningar kan i extrema fall ha över 1 *mm* misspass.

Moire. Moiréeffekt eller interferensmönster. Oönskade regelbundna lågfrekventa mönster som uppstår då flera raster läggs ovanpå varandra.

Moire peaks. Moirétoppar. Toppar i Fourierspektrumet som har lägre frekvens än första ordningens toppar och härrör från moiréeffekten.

O

Orded dither. Rastreringsmetod som producerar regelbundna (periodiska) mönster i rasterbilden.

Orthographic. Ortografisk. En bild som ej är utsatt för skalning, rotation, skevning eller brus.

P

Peaks. Toppar (i Fourierspektrumet). Starka frekvenskomponenter som härrör från rastreringen.

Pixels per inch (ppi). Måttenhet för upplösning i inlästa bilder, det vill säga samplingsfrekvensen hos scannern.

Posterization. Reduktion av antalet färger eller gråskalenivåer som resulterar i att mjuka toningar får tydliga steg (kvantiseringseffekt). En oönskad effekt som ofta uppträder när man använder för hög rastertäthet på en skrivare med för låg upplösning.

S

Screen, or screen pattern. Det regelbundna punktmönster som rasterbilden byggs upp av.

Screen angle. Rastervinkel. Den vinkel som rastermönstret är vridet med.

Screen frequency. Rastertäthet eller rasterfrekvens. Frekvensen av rasterpunkter.

Screen vectors. Rastervektorer. Vektorer som definierar rastertäthet och rastervinkelvinkel i två dimensioner.

T

Threshold halftoning. Tröskelrastrering. Rastrering med hjälp av en trösklingsfunktion eller trösklingsmatris.

Transition area. Övergångsområde. Den mjuka övergången mellan rasterpunkt och omgivningen som bildas på grund av punktförstoringen.

Appendix D. REFERENCES

This appendix is divided into two parts. The first part contains the references that directly are referred to in this report. The second part is a bibliography of all the current literature on inverse halftoning of clustered dot halftone images.

D.1 LITERATURE DIRECTLY REFERRED TO IN THIS REPORT

[Brac86]
"The Fourier Transform and its Applications (second edition)"
by **Ronald N. Bracewell**
McGraw-Hill, 1986. ISBN 0-07-066454-4.

[Ulich87]
"Digital halftoning"
by **Robert Ulichney**
The MIT Press, 1987. ISBN 0-262-21009-6. Based on a Ph.D. thesis at MIT (1986), entitled "Digital halftoning and the physical reconstruction function".

[Hunt91]
"Measuring Colour (second edition)"
by **Robert W.G. Hunt**
Ellis Horwood, 1991. ISBN 0-13-567686-X.

[Forch91]
"Digital Screening and Descreening of Images"
by **Søren Forchhammer, Kim Skovgård Jensen, Barun Mukherjee & H.O. Jørgensen**
Subreport 3a/3 of the project: Digital Image Processing Adapted to Graphic Production. Institute of Circuit Theory and Telecommunication, Technical University of Denmark. July 1991.

[Forch94]
"Data Compression of Scanned Halftone Images"
by **Søren Forchhammer & Kim Skovgård Jensen**
IEEE Transactions on Communications, vol. 42, no. 2/3/4, February/March/April 1994, pp. 1881–1893.

[Agfa94]
"En Introduktion till Digital Bildinläsning (Digital Prepress i Färg volym fyra)"
by **AGFA**
Agfa-Gevaert N.V., 1994. (Swedish)

[Poyn95]
"Frequently Asked Questions about Colour"
by **Charles A. Poynton**
Report from Internet, found at ftp://ftp.inforamp.net/pub/users/poynton/doc/colour/

[Wed95]
"Modelling of Dot Gain in Halftone Colour Prints"
by **Mikael Wedin**
Linköping Studies in Science and Technology, Thesis no. 508, Image Processing Laboratory, Linköping University and Institute of Technology. LIU-TEK-LIC-1995:40. September 1995. ISBN 91-7871-591-1.

[Gust95]
"Modelling of Light Scattering Effects in Print"
by **Stefan Gustavson**
Linköping Studies in Science and Technology, Thesis no. 508, Image Processing Laboratory, Linköping University and Institute of Technology. LIU-TEK-LIC-1995:52. October 1995. ISBN 91-7871-623-3.

[Joh96]
"Descreening of Cluster-dot Screened Images"
by **Daniel Johansson**
Master's thesis, Department of Electrical Engineering, Image Processing Group, Linköping University and Institute of Technology. LiTH-ISY-EX-1488. 18/3/96.

[AG96]
"Color Calibration of Digital Devices"
by **Anders Gustafsson**
Master's thesis, Department of Electrical Engineering, Image Processing Group, Linköping University and Institute of Technology. LiTH-ISY-EX-1682. 20/9/96.

D.2 BIBLIOGRAPHY ON INVERSE HALFTONING

The following sources of information could be of value for future work in this field:

"Skew and Distance of Digitized Grid Points in the Plane"
by **Søren Forchhammer**
SCIA '87, Proceedings of the 5th Scandinavian Conference on Image Analysis, Stockholm, vol. 2, pp. 451–458, 2–5/6/87.

"Algorithms for Coding Scanned Halftone Pictures"
by **Søren Forchhammer & Morten Forchhammer**
IEEE, Proceedings of 9th ICPR, Rome, pp. 297–299, 14–17/11/88.

"Digital Plane and Grid Point Segments"
by **Søren Forchhammer**
Computer Vision, Graphics, and Image Processing, vol. 47, no. 3, September 1989, pp. 373–384.

"Digitale systemer til datakompression af billeder"
by **Kim Skovgård Jensen & Barun Mukherjee**
Master's thesis, Institute of Circuit Theory and Telecommunication, Technical University of Denmark. IT-89-02, ID-E 461. 1/9/89. (Danish)

"Reconstruction from Halftone Images using Gradient Estimates"
by **Søren Forchhammer & Kim Skovgård Jensen**
Lyngby 1990. Technical report from the Institute of Circuit Theory and Telecommunication, Technical University of Denmark.

"Reconstruction Filters using Gradients with application to Halftone Images"
by **Søren Forchhammer & Kim Skovgård Jensen**
Lyngby 1991. Technical report from the Institute of Circuit Theory and Telecommunication, Technical University of Denmark.

"Electronic Screening at Arbitrary Angles and Rulings"
by **Søren Forchhammer & Kim Skovgård Jensen**
Proceedings of TAGA, 1991, pp. 20-35.

"Digital Screening and Descreening of Images"
by **Søren Forchhammer, Kim Skovgård Jensen, Barun Mukherjee & H.O. Jørgensen**
Subreport 3a/3 of the project: Digital Image Processing Adapted to Graphic Production. Institute of Circuit Theory and Telecommunication, Technical University of Denmark. July 1991.

"Quality Evaluation of Descreened/Rescreened Images"
by **Aa. Frøslev-Nielsen, H.O. Jørgensen, M. Klaman, S. Forchhammer & K. Rousing**
Subreport 3b/3 of the project: Digital Image Processing Adapted to Graphic Production. September 1991.

"Reconstruction from Halftone Images using Gradient Estimates"
by **Søren Forchhammer & Kim Skovgård Jensen**
SCIA '91. Proceedings of the 7th Scandinavian Conference on Image Analysis, Aalborg, vol. 1, pp. 249–257. 13–16/8/91.

"Data Compression of Scanned Halftone Images"
by **Søren Forchhammer & Kim Skovgård Jensen**
IEEE Transactions on Communications, vol. 42, no. 2/3/4, February/March/April 1994, pp. 1881–1893.

"Filters Involving Derivatives with Application to Reconstruction from Scanned Halftone Images"
by **Søren Forchhammer & Kim Skovgård Jensen**
IEEE Transactions on Image Processing, vol. 4, no. 4, April 1995, pp. 448–459.

"Om du vill scanna färdiga filmer"
by **Aage Frøslev-Nielsen**
Aktuell Grafisk Information, no. 266, April 1996, pp. 30–31. (Swedish)

"Avrastrering – från forskning till färdig metod"
by **Søren Forchhammer**
Aktuell Grafisk Information, no. 266, April 1996, pp. 31. (Swedish)

"Descreening of Cluster-dot Screened Images"
by **Daniel Johansson**
Master's thesis, Department of Electrical Engineering, Image Processing Group, Linköping University and Institute of Technology. LiTH-ISY-EX-1488. 18/3/96.

"Descreening and Rescreening of Halftoned Images"
by **Stefan Gustavson & Björn Kruse**
Report, Department of Electrical Engineering, Image Processing Group, Linköping University and Institute of Technology. LiTH-ISY-I-1271. 11/10/91.

"Inverse Ordered Dithered Halftoning using Permutation Filters"
by **Yeong-Taeg Kim & Gonzalo R. Arce**
IEEE International Conference on Image Processing, vol. 2, 1994, pp. 1017–1021.

"Inverse Halftoning Using Binary Permutation Filters"
by **Yeong-Taeg Kim, Gonzalo R. Arce & Nikolai Grabowski**
IEEE Transactions on Image Processing, vol. 4, no. 9, September 1995, pp. 1296–1311.

"Inverse Halftoning via MAP Estimation"
by **Robert L. Stevenson**
Submitted to IEEE Transactions on Image Processing, 2/3/94, revised 28/2/95. Report from Internet, found at http://lisa.ee.nd.edu/rls/papers/J15/

"Converting dithered images back to gray scale"
by **Allen Stenger**
Dr. Dobb's Journal, vol. 17, no. 11, November 1992, pp. 64–68.

“Incoherent 2-D spatial filtering of screened images”
by **Elsa N. Hogert & Nestor G. Gaggioli**
Applied Optics, vol. 29, no. 17, June 1990, pp. 2564–2568.

“New Results on Reconstruction of Continuous-tone from halftone”
by **Mostafa Analoui & Jan P. Allebach**
IEEE ICASSP-92, Proceedings of the International Conference on Acoustics, Speech and Signal Processing, vol. 3, pp. 313–316, 23–26/3/92.

“Descreening via linear filtering and iterative techniques”
by **Randall S. Kern, Thomas G. Stockham Jr. & David C. Strong**
Proceedings of the SPIE, Human Vision, Visual Processing and Digital Display IV, 1993, pp. 299–309.

“Inverse Halftoning for Monochrome Pictures”
by **Li-Ming Chen & Hsueh-Ming Hang**
IEEE International Conference on Image Processing, vol. 2, 1994, pp. 1022–1026.

INDEX

M

O

P

R

S

T

U

W

Avdelning, Institution
Division, department

Department of Electrical Engineering
Image Processing Laboratory

Datum
Date

1997-01-31

Språk
Language

☐ Svenska/Swedish
☑ Engelska/English
☐ _____

Rapporttyp
Report: category

☐ Licentiatavhandling
☑ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ _____

ISBN

ISRN

Serietitel och serienummer
Title of series, numbering

ISSN

LiTH-ISY-EX-1713

URL för elektronisk version

Titel
Title

Inverse Halftoning of Scanned Colour Images

(Avrastrering av digitalt inlästa färgbilder)

Författare
Author

Jörgen Rydenius

Sammanfattning
Abstract

This thesis is concerned with inverse halftoning of scanned colour halftone images. Different aspects of the problem are treated thoroughly and the difficulties are pointed out. Halftones, and especially colour halftones, are analysed in the Fourier domain.

A method to perform inverse halftoning of colour halftone images is proposed and implemented in Matlab. The resulting image quality is good and the method may very well be fully automized in the future.

The thesis gives an introduction to the concept of inverse halftoning and could serve as a base for future development of educational material on the topic.

Nyckelord
Keywords

descreening, halftoning, inverse halftoning, undithering, colour scanning, colour classification, image reconstruction